Learning to Optimize: Applications in Physical Designs and Manufacturing

by

Haozhu Wang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical and Computer Engineering)
in The University of Michigan
2022

Haozhu Wang

hzwang@umich.edu

ORCID iD: 0000-0002-9679-0144

To my parents Mr. Ping Wang and Mrs. Zhongmei Li and my wife Dr. Tianyu Liu for their unconditional love.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor Prof. L. Jay Guo. Without the guidance and support from Prof. Guo, this dissertation would not exist. I first met Prof. Guo in April 2016 but did not start in Prof. Guo's research group when I joined the EECS Department in September 2016 as a Ph.D. student. After spending almost three and a half years working on other research directions including organic opto-electronics and machine learning for healthcare, I discovered shared research interests between Prof. Guo and I on applying machine learning to real-world engineering and physical science problems. I started working with Prof. Guo on this fascinating research direction since January 2020. Back then, I could never imagine how much I would manage to learn and achieve in the following two years. Throughout the past two years, Prof. Guo has taught me tremendous amount of important lessons, not only on becoming an independent researcher, but also on being a kind and thoughtful person. Simply put, Prof. Guo is the advisor any Ph.D. student can ever dream to have and I feel extremely blessed being mentored by him, especially during the difficult global pandemic period.

I would also like to thank my committee members Prof. Theodore Norris, Prof. Xiaogan Liang, and Prof. Andrew Owens for providing invaluable feedback and suggestions on my dissertation and research direction.

I am also grateful for being able to work with Prof. Parag Deotare and Prof. Jenna Wiens. I worked with Prof. Deotare on organic opto-electronics and he taught me how to be a detail-oriented researcher. Prof. Wiens introduced me to the field of machine learning for healthcare and I learned how to use machine learning for social good from Prof. Wiens. Through both precious experience, I developed research skills that allowed the completion of this dissertation.

Additionally, I would like to think my amazing collaborators and friends, who have provided me support during my Ph.D. journey. The list goes long (not in any particular order): Prof. Laura Balzano, Prof. Brian Denton, Prof. Harrish Subbaraman, Prof. Chelsea Finn, Prof. Cheng Zhang, Prof. Stephen Salant, Chengang Ji, Zhong Zhang, Jingyang Sui, Zhongjun Jin, Zeyu Zheng, Dejiao Zhang, Hanfa Song, Jeeheh Oh, Jiaxuan Wang, Ian Fox, Rui Liu, Taigao Ma, Erkin Otles, Yongbum Park, Changyeong Jeong, Mustafa Tobah, Anwesha Saha, Mingzhe Yu, Hao Huang, Zhen Xu, Ruikun Luo, Dehui Zhang, Florent Muramutsa, Suyanpeng Zhang, Zhangxing Bian, Haleh Hagn-Shenas, Nick Johnson, Nitsan Ben-Gal, Libby Oliver, and many others who I couldn't list due to the limited space.

Last but not the least, I want to thank my parents and my wife Tianyu for their unwavering support. It's them who allowed me to escape from research to gain energy and confidence when I feel stuck. I'm extremely fortunate to go through the Ph.D. journey together with Tianyu, who is the bravest girl I ever know. I will never forget the countless nights we spent in NCRC where you worked on experiments while I was coding in your office. I'm proud that we both will continue a research career after years of Ph.D. grind.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Engineering and physical science often involve the *design* and *manufacturing* of physical devices. Conventionally, optimizing the physical design and the manufacturing process heavily relies on domain expertise and requires an iterative trial-and-error process conducted by human experts before achieving desired performance. Though numerical optimization methods have been developed for assisting domain experts, they often rely on heuristics that could be sub-optimal for the tasks of interest. Additionally, the performance of conventional optimization methods does not improve as more tasks are solved. This dissertation frames optimization as a learning problem, i.e., learning-to-optimize, where machine learning models are trained to solve optimization problems.

We propose three methods for solving practical optical inverse design and manufacturing problems. Our first proposed method `OML-PPO` treats optical multilayer thin films design tasks as sequence generation problems. Sequence generation networks that can discover optimal designs corresponding to user-specified optical properties are trained by reinforcement learning. The proposed method has been used to design a perfect broadband absorber with reflectance higher than 99%, an incandescent light bulb filter that can enhance the brightness by 16.3 times, and chrome replacement coatings with a close appearance to chrome films. Instead of targeting generic optical design tasks, our second method NEUTRON is a hybrid machine learning and optimization approach for efficiently designing optical multilayer thin films for structural color applications. By modeling the structural color inverse design as a bi-level optimization problem, NEUTRON applies machine learning models for fast, approximate material selection and particle swarm optimization for an exact search of the optimal thickness. We applied NEUTRON to both the chrome replacement coating and image reconstruction tasks. The results show that NEUTRON can achieve more accurate designs than machine learning or optimization alone. Thanks to the high efficiency of NEUTRON, we can reconstruct images with more than 200,000 pixels within a few hours. Our third method $\mathtt{M}^2\mathtt{BOP}$ addresses the costly data collection problem common in manufacturing problems by combining meta-learning and model-based offline reinforcement learning. By learning a meta environment model using offline data collected from relevant tasks, $\mathtt{M}^2\mathtt{BOP}$ can solve new tasks efficiently with a handful of data. On robot locomotion control tasks, $\mathtt{M}^2\mathtt{BOP}$ outperforms baseline approaches, especially on offline datasets that contain sub-optimal demonstrations.

CHAPTER 1

Introduction

Many engineering and physical science problems concern designing and manufacturing physical objects with desired properties. At the core of such problems are inverse design and manufacturing process parameter optimization, both critical for achieving the desired performance. Scientists and engineers often rely on a manual trial-and-error process guided by domain expertise to optimize the designs and the subsequent manufacturing process. However, such an iterative process is slow and could lead to sub-optimal results. Thus, it is highly desirable to develop tools for assisting domain experts with the design and manufacturing process. Fortunately, with the rapid digital revolution of most scientific and engineering fields, it has started to become possible to generate highly accurate simulation data or collect a large amount of real data of the underlying process for many real-world science and engineering problems. Thus, data-driven methods for aiding the design and manufacturing process start to gain increasing interest.

Though many data-driven approaches, especially machine learning methods, for inverse designs and manufacturing process parameter optimization have been proposed, systematic study of the challenges associated with these problems is rare. In this dissertation, we treat the application of machine learning in engineering optimization problems as a learning-to-optimize problem (*Li and Malik*, 2017; *Dai et al.*, 2017). The logged interaction data from human experts or existing solvers for specific optimization problems are treated as datapoints. A machine learning algorithm can extract the relationship between the features of different optimization problem instances and the optimal solutions. Thanks to the strong generalization ability of modern machine learning models, they can accurately predict the solutions for unseen optimization problems similar to the problems used for training the model.

When applying machine learning to solve optimization problems, it can either be used to predict solutions directly or be combined with existing optimization solvers. Both approaches have their advantages and should be appropriately chosen based on the specific optimization problem discussed in the later chapters. Though the algorithm details are different, researchers face the same challenges when applying both learning-to-optimize approaches in real optimization tasks, and it is critical to understand these challenges to allow seamless integration of the learned optimizers with the existing scientific discovery and engineering pipeline. Therefore, we provide a brief introduction to the most critical challenges we identified for the development of learning-to-optimize methods in real-world physical science and engineering problems.

1.1 Challenges and Opportunities

Engineering optimization is often not trivial due to the complexity of the problems and the high cost associated with validating optimized solutions. Specifically, solving engineering optimization problems share three common challenges including *large design space*, *non-uniqueness of global optima*, and *costly data collection*. In addition, extensive domain expertise is indispensable for solving most engineering optimization problems. Instead of focusing on a single domain, we aim to systematically study the common challenges associated with engineering optimization. Thus, domain expertise is not listed as one of the main challenges here.

Large Design Space The first and foremost challenge of real-world engineering optimization is the large design space to explore, as most of the problems are combinatorial. For example, designing optical multilayer thin films can easily require searching from a design space containing more than 10^{20} unique solutions, which makes it impossible to search all possible designs exhaustively. Thus, intelligent algorithms are required to navigate the promising design space efficiently. To this end, we explore *reinforcement learning* and *hybrid machine learning and optimization* approaches.

Non-uniqueness of Global Optima The mapping between the optimization variables and the outcome for many engineering systems does not have a one-to-one correspondence. Many different optimization variable values could correspond to the same desired outcome. Thus, methods that can only return a single solution for an engineering optimization problem may fail to identify potentially promising solutions. Stochastic machine learning models can solve this problem by learning the full distribution of potential solutions conditioned on given target outcomes. Thus, all the methods we develop throughout the thesis are based on stochastic models.

Costly Data Collection Machine learning is prevalent in data-abundant applications, including medical diagnosis, facial recognition, traffic forecasting, etc. However, the expensive collection process often makes datasets that can be collected for many engineering optimization problems small. In addition, it is often not possible to actively collect more data when optimizing an engineering task because the data collection process could be dangerous or disruptive for a mission-critical process. Both challenges make it hard to apply machine learning to engineering optimization. Fortunately, many optimization tasks are fundamentally related due to the shared underlying physical process. Moreover, offline data that have been previously collected by human experts or existing solvers are also often available. Thus, one natural question is whether it is possible to leverage the related offline small datasets to extract the shared commonalities of similar optimization tasks to solve them more efficiently without requiring a large amount of data collected from each task. A generic *offline meta-reinforcement learning* algorithm is developed with real-world manufacturing optimization tasks in mind to allow sample-efficient learning for learning-to-optimize tasks.

1.2 Summary of Contributions

This dissertation proposes learning-to-optimize methods that tackle the challenges mentioned above. The organization of this dissertation is as follows:

1. In Chapter 3, we discuss an automatic optical design algorithm OML-PPO (*Wang et al.*, 2021) that can efficiently explore the multilayer optical thin film design space through a novel sequence generation network trained by reinforcement learning.

2. In Chapter 4, we present a hybrid machine learning and optimization method NEUTRON (*Wang and Guo*, 2022) that treats the optical inverse design problem as a bi-level optimization task to explore the design space efficiently. We show that NEUTRON can efficiently identify highly accurate designs on structural color design tasks.

3. In Chapter 5, we develop an offline meta-reinforcement learning algorithm that can optimize the manufacturing process. On a standard robot locomotion simulation environment, we show that the proposed algorithm can efficiently learn from related tasks and perform well after adaptation on little experience.

4. In Chapter 6, we discuss promising future directions for advancing learning-to-optimize in real science and engineering problems.

CHAPTER 2

Background

Most engineering and physical problems concern the *design* and *manufacturing* of objects with specific properties, e.g., optical devices (*Molesky et al.*, 2018), integrated circuits (*Zheng and Louri*, 2019), protein structures (*Angermueller et al.*, 2020a), and mechanical parts (*Guo et al.*, 2021), to name a few. The optimization of both the design and manufacturing process are critical to scientific discoveries and engineering breakthroughs. However, despite the recent development of computer-aided tools for speeding up the design and manufacturing optimization process, both still heavily rely on human expertise and require a time-consuming trial-and-error process. This slow and potentially sub-optimal design and manufacturing optimization process strongly limits the innovation pace. Here, we formalize the problem of inverse design and manufacturing optimization. In addition, we provide a review of existing methods for tackling both problems. At the end, we briefly introduce machine learning tools studied in this dissertation for developing learning-to-optimize methods for inverse design and manufacturing process parameter optimization.

2.1 Inverse Design

Inverse design is an optimization problem that aims to find a system's design parameters x that leads to a system property y closest to the target property y^*. For example, a common inverse design task in optics is to find the materials and thickness for optical multilayer thin films that have a specific reflection spectrum.

Throughout this chapter, we assume the underlying relationship between the design parameters x and the property y is determined by a mapping $g(\cdot) : x \rightarrow y$ endowed by specific underlying physical processes. Inverse design aims to identify an inverse mapping or operation $f(\cdot) : y \rightarrow \mathcal{X}$ that can match a desired property target to corresponding design parameters. Due to the complex underlying processes and the potential one-to-many mapping between the target and the design parameters, it is often challenging to manually identify the appropriate design for a user-specified target property. Thus, both optimization-based methods and machine learning-based methods have been developed for assisting human experts with inverse design.

Optimization-Based Methods Optimization-based methods are iterative approaches that update the design parameters x to minimize a merit function $h(x)$ that measures the performance of the design. Depending on whether the merit function has an explicit analytical form based on the design parameter x, optimization-based inverse design methods can be categorized into gradient-based methods and black-box methods.

Common gradient-based methods include first-order approaches such as steepest descent and conjugate gradient methods, and second-order approaches including Newton's method and L-BFGS-B (*Boyd et al.*, 2004). All these methods follow the same high-level update procedure

$$x_{k+1} = x_k + \alpha_k p_k, \tag{2.1}$$

where α_k is the step size at the kth step, and p_k is the update direction based on the gradient or the Hessian information. Such gradient-based methods could be computationally expensive when the design involve high-dimensional parameters x. For example, when optimizing the topology of a structure, the gradient needs to be calculated for every point in the space. To address this issue, adjoint methods that can efficiently compute the gradient for the entire space have been applied for many real-world inverse design problems. To obtain the adjoint gradient for updating the entire topology, one only needs to perform a forward simulation and an adjoint simulation. However, the partial differential equation describing the property-of-interest needs to be known to computing the adjoint gradient (*Lalau-Keraly et al.*, 2013).

Compared to gradient-based methods, black-box optimization methods is applicable to a wider range of problems because no explicit analytical form describing the underlying system is required to be known (*Audet and Hare*, 2017). Genetic algorithms (*Deb et al.*, 2002), particle swarm optimization (*Kennedy and Eberhart*, 1995), and Bayesian optimization (*Pelikan et al.*, 1999) are widely used black-box optimization algorithms. Genetic algorithms and particle swarm optimization are both population-based heuristic search algorithms that maintain and iteratively update a group of solutions until the algorithm converge. Bayesian optimization relies on an iteratively updated probabilistic surrogate model that can be used for deriving an acquisition function. The design space is sequentially explored based on the acquisition function that is being updated simultaneously with the design space exploration process. For all three methods, the merit function needs to be evaluated either through simulation or experimentation before each update iteration. However, compared to heuristic-based methods, Bayesian optimization often requires fewer number of iterations and is used more widely in problems where the merit function is expensive to evaluate (*Snoek et al.*, 2012).

Machine Learning-Based Methods Unlike optimization-based methods for inverse design, machine learning-based inverse design approaches do not rely on the iterative process required by all optimization-based approaches. Instead, a model $f(\cdot) : y \to x$ that directly maps the desired target properties to designs is trained on a labeled dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=0}^{N-1}$ containing pairs of design parameters and design properties. Depending on whether the model is deterministic or stochastic, a given target property could be mapped to a single design y or a set of designs $\mathcal{Y}$. Common machine learning-based inverse design approaches include tandem networks (*Liu et al.*, 2018a), variational autoencoders (*Kingma and Welling*, 2013; *Ma et al.*, 2019), generative adversarial networks (*Goodfellow et al.*, 2014; *Liu et al.*, 2018b), invertible neural networks (*Ardizzone et al.*, 2019), mixture density networks (*Bishop*, 1994; *Unni et al.*, 2021), etc. Tandem networks and variational autoencoders are both based on autoencoders. However, the decoder and the encoder in a tandem network is trained sequentially while they are trained together in a variational autoencoder. Moreover, tandem networks can only map a desired target to a single design while variational autoencoders can maps the desired target to a distribution of designs. Generative adversarial networks are similar to variational autoencoders in the sense that they both implicitly learn the design distribution corresponding to the target design. However, generative adversarial networks can often better capture the multi-modality in the design distribution while being significantly harder to train than variational autoencoders. Invertible neural networks improves variational autoencoders by incorporating invertible neural network architectures to allow exact likelihood inference, which is useful for uncertainty estimation. Finally, mixture density networks model the target-design mapping with a mixture of gaussians and is the simplest stochastic machine learning model that can explicitly model the one-to-many mapping.

In our recent work on benchmarking machine learning-based inverse design models (*Ma et al.*, 2022), we systematically compared the relative performance of the most widely used inverse design methods in terms of their accuracy, design diversity, and robustness to fabrication errors on nanophotonic inverse design problems (Table 2.1).

Table 2.1: Comparison of common machine learning-based inverse design methods on nanophotonic inverse design problems.

Method	Accuracy	Diversity	Robustness
Tandem networks	⋆⋆⋆	⋆	⋆⋆
Variational autoencoders	⋆⋆⋆	⋆⋆	⋆⋆⋆
Generative adversarial networks	⋆⋆	⋆⋆⋆	⋆⋆

The biggest difference between optimization-based methods and machine learning-based methods is whether a dataset needs to be pre-collected. On the one hand, optimization-based methods can start without any pre-collected data points but require the data to be continuously generated through the iterative evaluation process. On the other hand, machine learning-based methods require a large dataset to be collect through either simulation or experiments. The dataset collection process could be costly but it is a one-time investment. If the inverse design tasks will be performed many times with different property targets, machine learning-based approaches is preferred because the model prediction is much faster than optimization-based approaches.

2.2 Manufacturing Process Parameter Optimization

After the initial design phase, the designed object needs to be manufactured. Most advanced manufacturing processes including semicondutor device fabrication, additive manufacturing, and electric vehicles production involve high-dimensional process parameters that need to be optimized (*Mahadevan and Theocharous*, 1998; *Dimopoulos and Zalzala*, 2000; *Cook et al.*, 2000; *Köksal et al.*, 2011; *Spielberg et al.*, 2017; *Pfrommer et al.*, 2018; *Nian et al.*, 2020). Unlike most inverse design problems, where accurate simulators can be build to predict the properties of the designs, it is often much more challenging to build accurate simulators for complex manufacturing processes and optimizing the manufacturing process parameters largely relies on the data measured from the real manufacturing process. Thus, optimization-based approaches that require iterative evaluations of the manufacturing quality are often intractable due to the cost of measuring real manufacturing data.

Surrogate-model based approaches that combines a trained machine learning model and an optimization procedure is most widely to tackle the above mentioned problem (*Hutter et al.*, 2013). Here, a small amount of data are initially collected through design of experiment (*Pukelsheim*, 2006) and used for training a machine learning model such as support vector machines or deep neural networks. The trained machine learning model serves as a surrogate to the underlying manufacturing process and can be used to predict the manufacturing quality given specific process parameters. Later, an optimization procedure is applied to query the trained model until convergence criterion is met. Due to the small size of the initial dataset used for training the surrogate model, model bias is often unavoidable, which could lead to invalid solutions. The most common strategy for addressing model bias is to iteratively update the surrogate model with labeled query data (*Hutter et al.*, 2011).

2.3 Machine Learning Methods for Learning to Optimize

Engineering optimization including inverse design and manufacturing optimization share same challenges including 1) large design space, 2) non-uniqueness of global optima, and 3) costly data collection. In this dissertation, we propose machine learning approaches to tackle these challenges. Specifically, we propose to use *reinforcement learning* and *hybrid machine learning and optimization methods* for efficiently explore the design space and obtain a set of optimal solutions. For addressing the expensive data collection issue, we propose to combine *offline learning* and *meta-learning* to leverage logged observational data from multiple related tasks to circumvent the expensive dataset collection process.

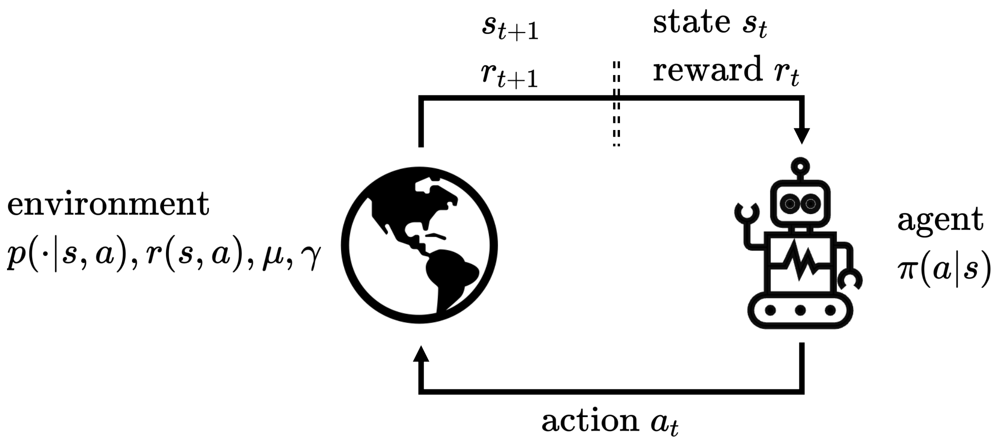


Figure 2.1: Reinforcement learning paradigm. An agent learns from interventional interactions with the environmental.

2.3.1 Reinforcement Learning

Reinforcement learning is a machine learning paradigm that concerns sequential decision making under uncertainty (*Sutton and Barto*, 2018). The sequential decision making problem is often formulated as a Markov Decision-Making Process (MDP) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, r, p, \mu, \gamma\}$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r(s,a)$ is the reward function, $p(\cdot|s,a)$ is the transition probability function (or transition dynamics), μ is the initial state distribution, and γ is the discount factor (Figure 2.1).

Reinforcement learning aims to maximize the discounted accumulated rewards for the entire task horizon by learning a mapping $\pi_\theta(a|s)$ from the observed states s to optimal actions a, i.e.,

$$\max_\theta J(\pi_\theta) = \mathbb{E}_{s_t \sim p(\cdot|s_{t-1}, a_t), a_t \sim \pi_\theta(\cdot|s_t)} \left[\sum_{t=1}^{H} \gamma^{t-1} r(s_t, a_t) \right] \tag{2.2}$$

Because of the flexibility of reward function definition, researchers can reflect specific optimization targets through carefully designed reward functions and engineering optimization problem is appropriately solved when the total return is maximized. Thus, reinforcement learning can be readily applied to inverse design and manufacturing optimization problems. Four families of approaches including approximate dynamic programming (*Mnih et al.*, 2013; *Van Hasselt et al.*, 2016), policy gradient (*Lillicrap et al.*, 2015; *Schulman et al.*, 2017), actor-critic (*Mnih et al.*, 2016; *Haarnoja et al.*, 2018a), and model-based reinforcement learning (*Deisenroth and Rasmussen*, 2011; *Chua et al.*, 2018; *Janner et al.*, 2019) have been developed to solve the above maximization problem. Approximate dynamic programming learns a parameterized state-action value function $Q(s,a)$ that measures the expected total future rewards when starting from a specific state-action pair (s,a). The state-action value function can later be used to obtain the policy $\pi(s) = \arg\max_{a\sim\mathcal{A}} Q(s,a)$. Policy gradient method directly learns a parameterized policy, often in the form of neural networks, through the policy gradient

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim p_{\pi_\theta}(\tau)}\left[\sum_{t=0}^{H}\gamma^t\nabla_\theta\log\pi_\theta(a_t \mid s_t)\underbrace{\left(\sum_{t'=t}^{H}\gamma^{t'-t}r(s_{t'},a_{t'}) - b(s_t)\right)}_{\text{advantage estimate }\hat{A}(s_t,a_t)}\right], \tag{2.3}$$

where $\tau = (s_0, a_0, s_1, a_1, \ldots, s_H)$, $p_{\pi_\theta}(\tau)$ is the trajectory endowed by the policy π_θ and the environment dynamics $p(\cdot|s,a)$, $b(s_t)$ is the baseline variable often computed as the average reward measured in state s_t. Actor-critic methods are improves upon both approximate dynamic programming and policy gradient methods by estimating the advantage using learned value functions instead of

Monte Carlo returns and update the policy through the policy gradient. Actor-critic methods often achieve state-of-art performance on standard benchmark tasks due to improved stability through the learning process when compared to approximate dynamic programming or policy gradient methods (*Haarnoja et al.*, 2018b). Finally, model-based reinforcement learning trains a dynamics model in the form of Gaussian Processes or probabilistic neural networks and generate rollouts from the learned dynamics model to train a policy. It can often be combined with the previously mentioned three families of methods when learning the policy. Planning can also be combined with the learned dynamics model to directly optimize the action sequence without learning a policy explicitly (*Clavera et al.*, 2019).

2.3.2 Hybrid Machine Learning and Optimization Methods

Hybrid machine learning and optimization methods initialize the optimization procedure with predictions from machine learning models (*Salcedo-Sanz et al.*, 2003; *Kuo et al.*, 2006). Such methods are well suited for solving bi-level optimization problems, where the upper level optimization is challenging but does not require very high accuracy. For example, when designing optical devices, both the materials and the structural parameters need to be optimized. The optical design problem can be considered as a bi-level optimization problem where one can treat the material selection as the upper level optimization task and the structural parameter designs conditioned on the material selection as the lower level optimization. Since the material selection task is often a combinatorial optimization task, it is often challenging to optimize with conventional optimization solvers. In this case, machine learning models can be applied to predict the most promising material selections to improve the overall optimization efficiency.

The bi-level optimization problem can be formulated as

$$\begin{aligned} \max_{x \in \mathcal{X}, z} \quad & F(x,z) \\ \text{s.t.} \quad & G(x,z) \geq 0 \\ & z \in S(x), \end{aligned} \tag{2.4}$$

where $S(x)$ is the set of optimal solutions for the following lower level problem

$$\begin{aligned} \max_{z \in \mathcal{Z}} \quad & f(x,z) \\ \text{s.t.} \quad & g(x,z) \geq 0. \end{aligned} \tag{2.5}$$

The function $G(x,z)$ in the upper level problem and the function $g(x,z)$ in the lower level problem represent the constraints of the optimization problem. For optical device optimization, lower level variable z is the structural parameters while upper level variable x is the material selection. Thus, the bi-level optimization problem aims to identify the best structural parameter and material selection combination.

Solving the bi-level optimization problem is often NP-hard. Hybrid machine learning and optimization methods provide an efficient approximate solution by using machine learning models to predict the promising upper level variable range to reduce the complexity of the problem (*Bagloee et al.*, 2018).

2.3.3 Offline Reinforcement Learning

Most reinforcement learning and optimization algorithms require direct feedback from the environment in the form of reward or merit function values. However, for many real-world problems that rely on real measurements, obtaining a large amount of direct feedback while learning or optimizing the task could be extremely expensive or even dangerous. Examples include learning treatments for patients (*Gottesman et al.*, 2019) , material designs (*Zhou et al.*, 2019), and autonomous driving (*Sallab et al.*, 2017), etc. Offline reinforcement learning solves this problem by learning a policy purely based on logged observational data previously collected by human experts or other existing working approaches (*Levine et al.*, 2020). The observational dataset contains transition tuples $\mathcal{D} = \{(s_t^i, a_t^i, s_{t+1}^i, r_t^i)\}$ that provide information on the transition dynamics and reward function of the environment. More importantly, state-action pairs that lead to high future total rewards can inform the learning of optimal policies. Essentially, offline reinforcement learning is a form a causal learning that requires a good understanding the underlying "mechanics" of the environment (*Pearl*, 2009).

A simple strategy to learn from the observational dataset is *imitation learning* (*Ho and Ermon*, 2016), which directly trains a probabilistic regression model or a generative model that maps the state to the action following the same behavior that generates transition tuples in the observational dataset. However, imitation learning can at most achieve the same performance as approaches used for collecting the observational data. On the contrary, offline reinforcement learning aims to extrapolate over the decisions collected from existing solutions to achieve better performance, which is often done through off-policy reinforcement learning approaches that are able to learn a state-action value function and a policy using experience generated from a different policy. Unlike conventional off-policy reinforcement learning algorithms, which still require sporadic online data collection from the environment, offline reinforcement learning is constrained to the setting where online data collection is not possible (*Kumar et al.*, 2020). Though conventional off-policy reinforcement learning methods such as deep Q-learning could perform well on some toy benchmark environments in the offline setting (*Agarwal et al.*, 2020), extrapolation with a value-function learned on a static real-life dataset often leads to *distributional shift* (*Fujimoto et al.*, 2019), which can be easily mitigated by collecting additional data as common in off-policy reinforcement learning but has been the major challenge of offline reinforcement learning due to the inability to train a more accurate function approximate with additionally collected state-action pairs that are rare in the initial training dataset. Due to the unavoidable estimation bias when learning from static data, distributional shift can lead to optimized policy that turns out to perform poorly when deployed in the testing environment. To address this issue, existing approaches can be categorized into three families 1) introduce policy constraints to force the learned offline policy to stay close to the behavior policy (*Kumar et al.*, 2020), 2) penalize the learned value function with estimated uncertainty (*Kumar et al.*, 2019), or 3) learning a pessimistic model or uncertainty-aware model for model-based learning or planning (*Yu et al.*, 2020; *Kidambi et al.*, 2020).

Policy Constraint-based Offline RL The first category for offline reinforcement learning is based on constraining the learned policy to the behavior policy to reduce extrapolation error. Instead of policy gradient methods, these methods are all based on approximate dynamic programming approaches because their off-policy is more amenable to the offline learning setting. The general form of policy-constrained offline reinforcement learning procedure can be written as:

$$\begin{aligned}
&\hat{Q}^{\pi}_{\theta'} \leftarrow \\
&\quad \arg\min_{\theta} \mathbb{E}_{(s,a,s')\sim\mathcal{D}}\left[\left(\hat{Q}_{\theta}(s,a)^{\pi} - \left(r(s,a) + \gamma \mathbb{E}_{s'\sim\pi_{\theta}}\left[\hat{Q}^{\pi}_{\theta}\left(s',a'\right)\right]\right)\right)^2\right] \qquad (2.6) \\
&\pi' \leftarrow \arg\max_{\pi} \mathbb{E}_{s\sim\mathcal{D}}\left[\mathbb{E}_{a\sim\pi_{\theta'}(a|s)}\left[\hat{Q}^{\pi}_{\theta'}(s,a)\right]\right] \text{ s.t. } D\left(\pi,\pi_{\beta}\right) \leq \epsilon.
\end{aligned}$$

where π_β is the behavior policy collecting the offline dataset, $D(\pi, \pi_\beta)$ is a *divergence measure* that quantifies the distance between the learned policy $\pi(a|s)$ and the behavior policy $\pi_\beta(a|s)$. By introducing the constraint that the learned policy *must not* deviate more than ϵ from the behavior policy, the overall extrapolation error can be bounded. In practice, divergence measures including KL-divergence (*Kumar et al.*, 2020), maximum mean divergence (*Kumar et al.*, 2019), and Fisher-divergence (*Kostrikov et al.*, 2021) have all been used for measuring the policy distance.

Uncertainty-Penalized Offline RL Uncertainty-penalized methods are inspired by the fact that *epistemic* uncertainty for rare or unseen state-action pairs when estimating the Q-value should be much higher than in-distribution pairs. Thus, one can estimate the epistemic uncertainty and adjust the estimated Q-value accordingly. The uncertainty-penalized Q-value learning procedure can be written as:

$$\pi' \leftarrow \arg\max_{\pi} \mathbb{E}_{s\sim\mathcal{D}} \left[\mathbb{E}_{a\sim\pi_{\theta'}(a|s)} \left[\hat{Q}^{\pi}_{\theta'}(s,a) - \zeta \cdot b_{\hat{Q}}(s',a')\right]\right] \tag{2.7}$$

where $b_{\hat{Q}}(s',a')$ is the estimated uncertainty of the Q-value function $\hat{Q}^{\pi}_{\theta}$ for the state-action pair (s',a'), which is adjusted by the hyperparameter ζ that controls how conservative the learned policy is. The penalty term can prevent over-optimistic actions due to the erroneous Q-value estimation on out-of-distribution state-action pairs. In practice, bootstrap ensembles (*Osband et al.*, 2016) or probabilistic neural networks (*O'Donoghue et al.*, 2018) have been applied to estimate the Q-value uncertainty.

Model-based Offline Reinforcement Learning In addition to previous two model-free approaches, learning a dynamics model and the reward function of the environment also allows robust offline policy learning (*Yu et al.*, 2020; *Kidambi et al.*, 2020; *Argenson and Dulac-Arnold*, 2021). When directly learning a policy from the rollouts generated by the learned model, uncertainty-based penalty can be added to the predicted reward to prevent learning over-optimistic actions (*Yu et al.*, 2020, 2021b). The learned model can also be combined with planning for direct trajectory optimization without explicitly learning a policy, as usually done in model-predictive control (*Argenson and Dulac-Arnold*, 2021; *Zhan et al.*, 2021).

2.3.4 Meta-Learning

Meta-learning is the problem of learning-to-learn, which is highly related to the field of learning-to-optimize as both can be viewed as meta-algorithms that learn a lower-level algorithm. The goal is to learn from a distribution of tasks so that a new task sampled from the same distribution can be quickly solved with a small amount of data. This idea has been proposed in 90's and recently attracted revived interests due to the advancement in deep learning and reinforcement learning (*Schmidhuber*, 1994). Meta-learning has been successfully applied to image classification (*Ren et al.*, 2018), domain adapataion (*Li et al.*, 2018a), continuous control (*Yu et al.*, 2019), and disease prediction (*Zhang et al.*, 2019b) but hasn't seen extensive adoption in physical science and engineering fields. However, inverse design and manufacturing optimization are both applications where similar tasks are being solved repetitively, which are amenable to meta-learning. In addition, the fact that collecting data is expensive for real-world physical science and engineering applications makes meta-learning an important research direction that could reduce the sample complexity of learning-to-optimize methods to allow a wider range of applications. Existing meta-learning approaches can be mainly grouped into two categories including gradient-methods and contextual meta-learning.

Gradient-based Meta-Learning Model-agnostic meta-learning (MAML) (*Finn et al.*, 2017) is the first proposed gradient-based meta-learning approach that learns a neural network initialization that is amenable to fast adaptation. By sampling disjoint support set and query set from the same task, MAML first updates the neural network initialization parameters with the support set through the inner update, then update the neural network initialization parameters with the meta-gradient

$$\theta' \leftarrow \theta - \beta \overbrace{\nabla_\theta \sum_{\tau_i \sim p(\tau)} \mathcal{L}_{\tau_i}^{qry} \left(f_{\underbrace{\theta - \alpha \nabla_\theta \mathcal{L}_{\tau_i}^{spt}(f_\theta)}_{\text{adaptation}}} \right)}^{\text{meta-gradient}} . \tag{2.8}$$

Since the computation of meta-gradient involves higher-order derivatives, which could be computationally expensive. First-order approaches (*Nichol and Schulman*, 2018) and latent-embedding approaches (*Rusu et al.*, 2018) have been proposed to improve the computation efficiency of gradient-based meta-learning methods. The biggest advantage of such methods is that they do not depend on specific neural network architectures, thus can be readily to many different application domains that require specialized neural network models such as LSTMs, CNNs, and Transformers, etc.

Contextual Meta-Learning Contextual meta-learning has been a popular approach for meta-reinforcement learning (*Duan et al.*, 2016; *Vuorio et al.*, 2019; *Li et al.*, 2021). The idea is that one can extract the contextual information from experience collected in a specific environment. Later, a prediction model or a reinforcement learning policy can be conditioned on the extracted contextual information to give adaptive outputs. In practice, LSTMs have been used for extracting the contextual information from experience and multiple different conditioning approaches including concatenation (*Zintgraf et al.*, 2019), multiplication (*Wang et al.*, 2014), and FiLM (*Perez et al.*, 2018) have been explored.

Contextual meta-learning often outperforms gradient-based meta-learning in terms of few-shot performance. However, one disadvantage of contextual meta-learning is that they require careful designs of the context-extraction model and the conditioning mechanism based on the application domains.

CHAPTER 3

Optical Multilayer Thin Film Design with Deep Reinforcement Learning

3.1 Introduction

Optical multi-layer films have been widely used in many applications, such as broadband filtering (*Yang et al.*, 2016), photovoltaics (*Agrawal and Peumans*, 2008), radiative cooling (*Raman et al.*, 2014), and structural colors (*Li et al.*, 2018b). The design of optical multi-layer films is a combinatorial optimization problem that requires one to choose the best combination of materials and layer thicknesses to form a multi-layer structure. Researchers and engineers often make such designs based on their physical intuition. However, a completely human-based design process is slow and often leads to sub-optimal designs, especially when the design space is enormous. Thus, computational methods for designing optical multi-layer structures, including evolutionary algorithms (*Schubert et al.*, 2008; *Shi et al.*, 2017; *You et al.*, 2020), needle optimization (*Tikhonravov et al.*, 1996), and particle swarm optimization (*Rabady and Ababneh*, 2014), have been proposed to tackle this problem. All of these previous methods frame the optical design task as an optimization problem and aim to synthesize a structure that meets user-specified design criteria. However, these methods for optical design are based entirely on heuristic search, i.e., they do not learn a model to solve the design problems. When the heuristic approach is sub-optimal for a task, the search process may fail to identify a high-performance design.

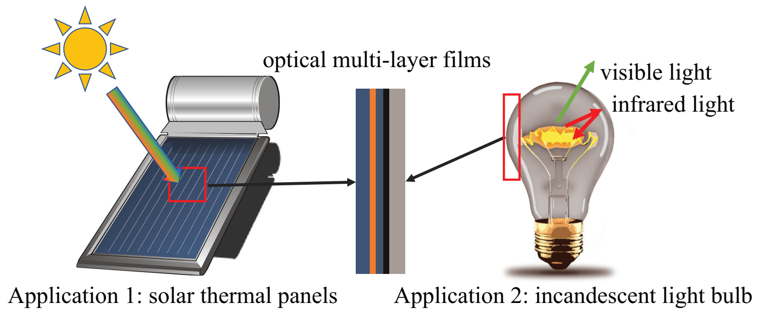

optical multi-layer films
visible light
infrared light
Application 1: solar thermal panels
Application 2: incandescent light bulb

Figure 3.1: Two energy applications of optical multi-layer films. For solar thermal panels, we can use multi-layer films as ultra-wideband absorbers to enhance light absorption efficiency. For incandescent light bulbs, we can coat multi-layer films on them to improve luminous efficiency by reflecting infrared light while transmitting visible light.

In contrast, deep reinforcement learning (DRL) is a learning framework that learns to solve complex tasks through an trial-and-error process. It is proven to be highly scalable for solving large-scale and complicated tasks (*Silver et al.*, 2017; *Vinyals et al.*, 2019). Researchers have successfully applied DRL to various combinatorial optimization problems (*Bello et al.*, 2016; *Khalil et al.*, 2017; *Mirhoseini et al.*, 2017, 2020). Unlike heuristic-based search, reinforcement learning methods learn a model using the reward signal (*Sutton and Barto*, 2018) and do not depend on hand-crafted heuristics. On some combinatorial optimization tasks, DRL has been shown to outperform classic heuristic search methods (*Lu et al.*, 2020). Recently, researchers applied DRL on designing optical devices with a structure template (*Sajedian et al.*, 2019a,b), where the number of layers is fixed. However, when designing the optical multi-layer films, we often do not know the optimal structure template. Thus, the previous DRL approaches are not suitable for multi-layer designs. In addition to DRL, deep learning-enabled inverse design methods have seen great development in recent years (*Ma et al.*, 2018; *Liu et al.*, 2018a,b). These inverse design models learn a mapping between design targets and design parameters using a static training set, which allows users to efficiently retrieve designs that match design targets. However, if a design target does not lie within the training datasets used for training the inverse design model, we will not be able to obtain the corresponding design using the inverse design model. For our performance optimization task, the optimal design is often not covered by a static training dataset. Otherwise, it would mean that the optimization task has already been solved through the training dataset collection process. Thus, reinforcement learning is more suitable than deep-learning-based inverse design methods when users want to optimize the design performance.

Because the multi-layer optical design task is equivalent to a sequence generation problem, we propose a DRL method called Optical Multi-layer Proximal Policy Optimization (OML-PPO) that can generate near-optimal multi-layer structures. The proposed method uses a state-of-the-art DRL algorithm PPO to train a deep recurrent neural network that outputs near-optimal optical designs. We introduce two novel designs for the deep recurrent neural network to allow it to efficiently explore the design space. With an ablation study, we show that the proposed neural network architecture enables the RL agent to explore the design space efficiently.

We applied the proposed method to two optical design tasks that are relevant to energy applications (Figure 3.1): 1) ultra-wideband absorbers that can enhance light-harvesting efficiency, e.g. for thermal photovoltaics and photothermal energy conversion 2) incandescent light bulb filters that can improve light bulb efficiency in emitting visible light. On the task of designing ultra-wideband absorbers, we show that OML-PPO can reliably discover high-performance designs. A 5-layer structure with 97.64% average absorption over the wavelength range [400, 2000] nm is discovered by OML-PPO, outperforming a previously reported structure using the same number of layers with 95.37% average absorption. When applied to generate absorbers with more layers, OML-PPO discovers a 14-layer structure that achieves near-perfect 99.24% average absorption. We also applied our method to design a 42-layer incandescent light bulb filter and achieved an enhancement factor of 16.60, which is 8.5% higher than a 41-layer structure designed by a state-of-the-art memetic algorithm. Our results demonstrate that the proposed algorithm is efficient at discovering near-optimal designs and is scalable to complicated design problems. We summarize our contributions:

1. We frame the multi-layer optical design task as a sequence generation problem and develop a DRL method (OML-PPO) for solving this task.

2. We propose a novel deep sequence generation network that allows efficient exploration of the optical design space.

3. On two optical design tasks, we demonstrate that our method is effective in discovering near-optimal solutions for complicated design tasks.

3.2 Related Work

Researchers have developed reinforcement learning methods for solving various combinatorial optimization problems. In (*Bello et al.*, 2016), the authors trained a Pointer Network (*Vinyals et al.*, 2015) to solve the Traveling Salesman Problem (TSP). Khalil et al. (*Khalil et al.*, 2017) combined graph embedding and RL for solving a diverse set of combinatorial optimization problems including the Minimum Vertex Cover, Maximum Cut, and TSP. Chen and Tian (*Chen and Tian*, 2019) proposed a method to learn policies that can rewrite the heuristics in existing solvers for combinatorial optimization problems. Lu et al. (*Lu et al.*, 2020) showed that RL-based method could outperform a classic operation research algorithm in terms of both average cost and time efficiency.

Many real-life applications can be formalized as sequence generation problems (*Li et al.*, 2016; *Popova et al.*, 2018; *Angermueller et al.*, 2020b; *Mirhoseini et al.*, 2020). In (*Li et al.*, 2016), the authors integrated RL and seq2seq to automatically generate a response by simulating the dialogue between two agents. In (*Angermueller et al.*, 2020b), the authors proposed a model-based variant of PPO to deal with the large-batch, low round setting for biological sequence design (*Angermueller et al.*, 2020b). Mirhoseini et al. (*Mirhoseini et al.*, 2020) combined graph neural networks with RL for sequentially placing devices on a chip. These previous works all trained sequence generation models using policy gradient algorithms. In this work, we introduced a sequence generation network architecture tailored to the optical design task. Additionally, we combined local search with DRL for finetuning the thicknesses of the generated layers.

Deep-learning-based inverse design (*Ma et al.*, 2018; *Liu et al.*, 2018a,b) has been gaining popularity in recent years. In (*Ma et al.*, 2018), the authors trained convolutional neural networks to directly predict design parameters using the design target as the input to the network. Liu et al. (*Liu et al.*, 2018b) trained a generative adversarial network (GAN) to inversely design optical devices by generating 2D shapes of the optical structure. However, these approaches all rely on a curated training set that contains diverse examples. When our goal is to push the performance limit of certain devices, the near-optimal structures is unlikely to be within the training data distribution. Thus, these static methods are not appropriate for optimizing design performances. Our proposed method tackles this problem by actively searching the design space to generate high-performance designs via reinforcement learning. In (*Jiang et al.*, 2019), the authors also developed an active search process by adding additional high-quality data to augment the initial training set. However, their approach requires the users to retrain the neural network with the augmented dataset while our RL-based method accomplishes the design task within one training process.

3.3 Methods

Multi-layer films can be treated as sequences. Each layer is represented as $s_l = (m_l, d_l)$. We can represent such a structure with N layers as $\mathcal{S} = \{(m_0, d_0), (m_1, d_1), (m_2, d_2), \cdots, (m_{N-1}, d_{N-1})\}$, where m_l and d_l denote the material and the thickness of the l-th layer (counting from the top), respectively. When designing optical multi-layer films, we hope to synthesize a sequence that has the desired target spectral response $\tilde{T}$. Thus, the design task is equivalent to a sequence generation problem, where we generate m and d in each step. Generation tasks such as dialogue generation (*Li et al.*, 2016), molecule generation (*Popova et al.*, 2018), and biological sequence generation (*Angermueller et al.*, 2020b) have been widely studied by machine learning researchers. In these works, researchers train a neural network as a generator for synthesizing sequences. Because we do not have ground-truth data for optimal design tasks, we apply reinforcement learning (*Sutton and Barto*, 2018) to train the sequence generator.

3.3.1 Sequence Generation Network

To generate the optical layer sequences, we use a recurrent neural network (RNN) (*Hochreiter and Schmidhuber*, 1997). Unlike simple feed-forward neural networks, RNNs maintain a hidden state h that contains useful information from the history of the sequence. Thus, RNNs are suitable for tasks that require memorizing history and have been widely used in sequence generation tasks (*Graves*, 2013). Gated recurrent units (GRUs) (*Chung et al.*, 2014) and long short-term memory networks (LSTMs) (*Hochreiter and Schmidhuber*, 1997) are two popular variants of RNNs. Researchers have previously found that the empirical performance of GRUs and LSTMs is similar. Because GRUs have a simpler structure than LSTMs and require fewer parameters to train, we choose to use a GRU for generating the optical multi-layer structures. Similar to sampling words from a dictionary when generating a sentence, we sample the material m_l from a fixed set of materials $\mathcal{M}$ for each layer. Though the thickness d_l is intrinsically a continuous variable, we choose to sample the thickness from a set of discrete values $\mathcal{D}$ to reduce the size of the exploration space. Later, we apply quasi-Newton methods (*Zhu et al.*, 1997) to finetune the layer thicknesses of the generated structure for further performance improvement.

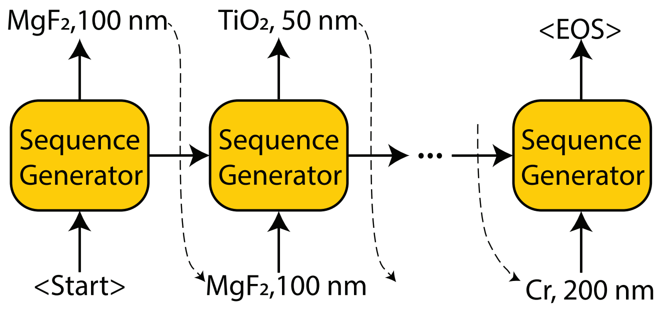


Figure 3.2: Optical multi-layer design as sequence generation. The generation process will stop when either the EOS token is sampled, or the length of the sequence reaches the maximum allowed length L.

Our optical multi-layer sequence generation network consists of a GRU and two multi-layer perceptrons (MLPs) (*Goodfellow et al.*, 2016). Gated Recurrent Unit (GRU) is a variant of recurrent neural network that has been widely adopted in many applications *Chung et al.* (2014). The operation of GRU can be summarized as below:

$$\begin{aligned} z_t &= \sigma\left(W_z \cdot [h_{t-1}, x_t]\right) \\ r_t &= \sigma\left(W_r \cdot [h_{t-1}, x_t]\right) \\ \tilde{h}_t &= \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right) \\ h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t, \end{aligned} \tag{3.1}$$

where x_t is the input at step t.

At generation step l, the GRU takes its own output from the previous step $s_{l-1} = (m_{l-1}, d_{l-1})$ and the previous hidden state h_l as the inputs to compute the hidden state h_l. This auto-regressive generation process allows the GRU to remember what has been generated so far. To generate the material and thickness for layer l, the hidden state h_l of the GRU is inputted to two MLPs. One of the MLPs outputs logits vector $\sigma_{m_l} \in \mathbb{R}^{|\mathcal{M}|+1}$ corresponding to all possible materials and an end-of-sequence token (EOS). The other MLP outputs a thickness logits vector $\sigma_{d_l} \in \mathbb{R}^{|\mathcal{D}|}$ corresponding to all allowable thicknesses in the set $\mathcal{D}$. Then, we transform these logits vectors with the *softmax* function to obtain proper probability distributions. Finally, the material and thickness are sampled from their corresponding distributions. The generation process will stop either when the length reaches the maximum length L set by the user or when the EOS token is sampled. Thus, the number of layers N of a generated structure is always lower than or equal to the maximum sequence length L. The process for generating a sequence is illustrated in Figure. 3.2.

3.3.1.1 Non-repetitive gating

The aforementioned material sampling procedure does not prevent the situation where the same material is sampled for adjacent layers. However, such consecutive layers of the same material are equivalent to a single thicker layer. Thus, allowing the sequence generator to generate the same material for adjacent layers leads to redundant computation. Moreover, doing so increases the exploration space size and makes the search problem harder. Thus, we introduce a non-repetitive gating function that removes the logit element corresponding to the most recently sampled material to prevent the sequence

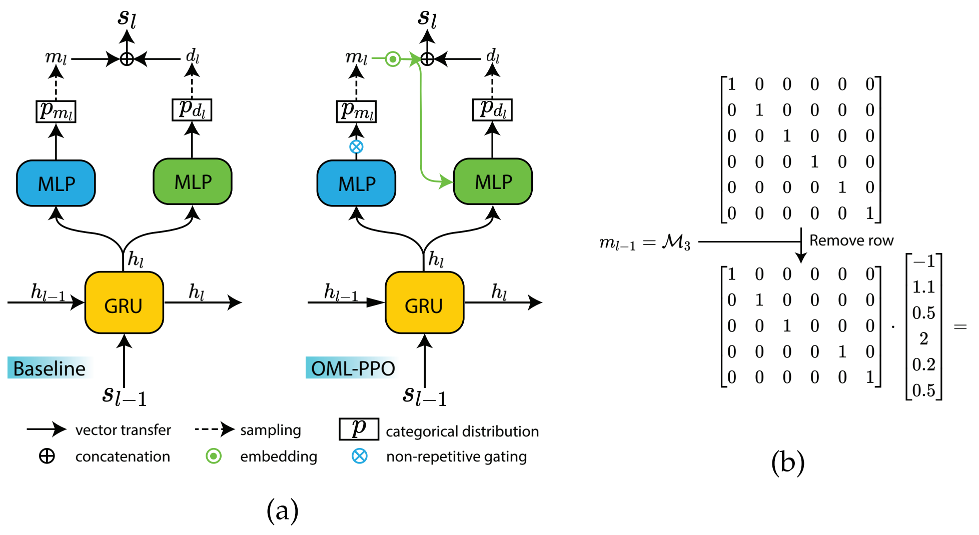


Figure 3.3: Neural network architectures for generating optical multi-layer films. (a) We show one generation step in the plot. The hidden state h_l of the GRU is passed to two MLPs to output material and thickness probabilities, respectively. The actual material and thickness for layer l are sampled from categorical distributions parametrized by p_{m_l} and p_{d_l}. Built-upon the baseline architecture, our proposed model adds a non-repetitive gating function and auto-regressive connection between the sampled material and the thickness MLP. (b) Illustration of how the non-repetitive gating works. Here we suppose there are a total of 5 materials. Thus, the gating matrix is of dimension 5×6.

generator from generating the same materials in a row. This gating function is a matrix $I_{NR} \in \mathbb{R}^{|\mathcal{M}|\times(|\mathcal{M}|+1)}$ formed by removing the row corresponding to the most recently sampled material from an identity matrix. When multiplied with the logits vector σ_{m_l}, the element corresponding to that material will be removed, i.e., $\sigma'_{m_l} = I_{NR} \cdot \sigma_{m_l} \in \mathbb{R}^{|\mathcal{M}|}$. Then, we pass the transformed logit vector σ'_{m_l} to the softmax layer to obtain the sampling probability. By doing so, we set the sampling probability for the recurring material to 0. With the non-repetitive gating, the generated material sequence is guaranteed to have different materials for adjacent layers. Note that, we do not apply the gating function for the first generation step because there is no previously sampled material.

3.3.1.2 Auto-regressive generation of material and thickness

Because the proper thickness of a layer should depend on the material, we input the sampled material m_l to the thickness MLP in addition to the hidden state h_l. A similar approach has been applied in RL problems where the actions are dependent on each other (*Vinyals et al.*, 2019). Instead of using a one-hot vector to represent the material, we train a material embedding matrix $emb \in \mathbb{R}^{|\mathcal{M}| \times d}$ together with the sequence generator network. Each row $emb_m \in \mathbb{R}^d$ of the embedding matrix is a continuous representation of one material, where d is the embedding size. Using an embedding allows us to use a large number of materials without significantly increasing the dimensionality of the material representation. The material embedding vector for the sampled material emb_{m_l} is concatenated with the hidden state h_l to form the input $[emb_{m_l}, h_l]$ to the material MLP.

The full sequence generator architecture is plotted in Figure. 3.3a. To understand the effect of non-repetitive gating and modeling the dependency between the material and the thickness, we compare the proposed OML-PPO architecture against a baseline architecture Experiment section.

3.3.2 Training Sequence Generation Network with PPO

We train the sequence generation network with reinforcement learning. The goal of reinforcement learning is to maximize expected cumulative rewards $G = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$ by learning a policy $\pi_\theta(a|s)$ that can map a state s to an action a. Here, γ is the discount factor that penalizes future rewards and r_t is the reward at step t. The sequence generation network described above serves as the policy.

We represent the state at the l-th generation step as the concatenation of the last layer information and the GRU hidden state, i.e., $s_l = [(m_{l-1}, d_{l-1}), h_l]$. The actions a_l correspond to the material and thickness (m_l, d_l) of the current layer. We set the reward to be 0 for all generation steps except the final step. At the final step (i.e., the structure $\mathcal{S}$ has been completely generated), we compute the spectrum of the generated structure with an optical spectrum calculation package TMM (*Byrnes*, 2016) and assign the final reward based on how well the structure spectrum matches with the target spectrum. We also tried to calculate the spectrum following every generation step and assign intermediate rewards. However, this dense-reward approach is slow and does not lead to improved performance. Thus, we only report the final-only approach here. We set the discount factor $\gamma = 1$. Thus, the cumulative reward G for the generated sequence $\mathcal{S}$ is simply the reward at the final step, which is defined as one minus the mean absolute error between the spectrum of the generated structure and the target spectrum:

$$G(\mathcal{S}) = 1 - \frac{1}{K} \sum_{k=0} \frac{1}{J} \sum_{j=0}^{J-1} |T^{\mathcal{S}}(\lambda_j, \delta_k) - \tilde{T}(\lambda_j, \delta_k)| \tag{3.2}$$

where $T^{\mathcal{S}}(\lambda_j, \delta_k)$ is the spectrum of the generated structure $\mathcal{S}$ at wavelength λ_j under incidence angle δ_k. Because $T \in [0, 1]$, the cumulative reward is always non-negative. The reward value will become higher as the spectrum $T^{\mathcal{S}}$ gets closer to the target spectrum $\tilde{T}$ until it reaches 1 when the structure spectrum perfectly matches with the target spectrum.

During training, the sequence generator π_θ actively generates new structures and receive rewards. Our goal is to maximize the expected rewards for structures sampled from the sequence generation network:

$$J(\theta) = \mathbb{E}_{\mathcal{S} \sim \pi_\theta}[G(\mathcal{S})]. \tag{3.3}$$

Based on the calculated rewards for generated sequences, the agent adjusts its parameters θ with gradient ascent so that future rewards can be improved. Here, we use a policy gradient algorithm to compute the gradient $\nabla_\theta J(\theta)$ for updating the sequence generator π_θ. From the policy gradient theorem (*Sutton and Barto*, 2018; *Schulman et al.*, 2017), we have

$$g = \nabla_\theta J(\theta) = \mathbb{E}_{\mathcal{S} \sim \pi_\theta}\left[A(\mathcal{S}) \nabla_\theta \log P_\theta(\mathcal{S})\right], \tag{3.4}$$

where $P_\theta(\mathcal{S}) = \prod_{l=0}^{N-1} p_\theta(m_l | s_{l-1}, h_{l-1}) \cdot p_\theta(d_l | m_l, s_{l-1}, h_{l-1})$ is the probability of sampling a structure $\mathcal{S}$ from the generator network π_θ and $A(\mathcal{S})$ is the estimated advantage function (*Schulman et al.*, 2015), which measures the performance of the generated sequence $\mathcal{S}$ compared against the average performance of structures sampled from π_θ.

Instead of directly updating the sequence generator using Eqn.3.4, we use a state-of-the-art policy gradient algorithm *Proximal Policy Optimization* (PPO) (*Schulman et al.*, 2017) to compute the policy gradient from a surrogate objective function:

$$g = \nabla_\theta \mathbb{E}_{\mathcal{S} \sim \pi_\theta}\left[\min\left(r(\theta) A_{\theta_v}(\mathcal{S}), \text{clip}\left(r(\theta), 1-\epsilon, 1+\epsilon\right) A_{\theta_v}(\mathcal{S})\right)\right], \tag{3.5}$$

where $r(\theta) = \frac{P_\theta(\mathcal{S})}{P_{\theta_{\text{old}}}(\mathcal{S})}$ is the importance weight that measures the distance between the policies before and after the gradient update. The clip function disincentivizes large update steps to the policy, where ϵ is a hyperparameter that affects the actual update size. Here, the advantage A_{θ_v} is estimated by *Generalized Advantage Estimation* (GAE) (*Schulman et al.*, 2015), which achieves a good balance between bias and variance of the estimated gradients. θ_v is the model parameters for a critic network that is trained together with the sequence generator. Compared to the vanilla policy gradient and actor-critic algorithms, PPO is more sample-efficient because it allows multi-step updates using the same batch of trajectories. Previous results show that PPO can achieve state-of-the-art performance on many tasks (*Schulman et al.*, 2017). With the computed policy gradient, the sequence generator model parameters are updated using the Adam optimizer (*Kingma and Ba*, 2014). The model training process is summarized in Figure. 3.4. Similar to the *active search* approach in Bello et al. (*Bello et al.*, 2016), we output the best structure discovered throughout the entire training process as the final design. The pseudocode that summarizes our design generation process is given in Algorithm **??**.

Our model is implemented using PyTorch (*Paszke et al.*, 2019) and Spinning Up (*Achiam*, 2018). The data used in this study and our code are publicly available[1].

[1]https://github.com/hammer-wang/oml-ppo

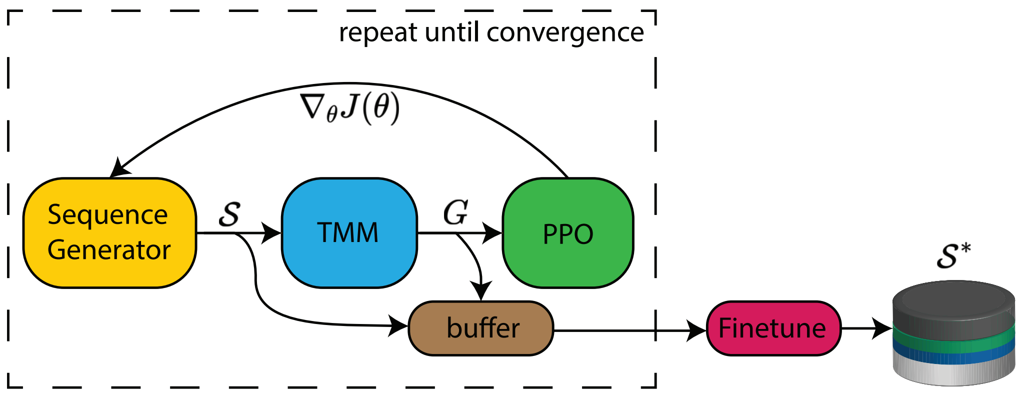


Figure 3.4: Pipeline of the sequence generator training process. We first generate multi-layer structures using the sequence generator π_θ. The spectrum of the generated structures are simulated by the TMM module. Next, PPO algorithm is applied to compute the policy gradient g for updating the sequence generator model. We keep pushing the best discovered structure into a buffer with size 1. This process is repeated until convergence. Finally, we finetune the layer thicknesses to obtain the design.

3.4 Experiments

We applied the proposed method to two optical design tasks that are relevant to energy applications, i.e., 1) designing ultra-wideband absorbers and 2) designing incandescent light bulb filters. The designed ultra-wideband absorbers can help solar thermal panels to absorb the sunlight more efficiently and the light bulb filter can enhance incandescent light bulb efficiency in emitting visible light while suppressing the radiation in the infrared range that represents energy loss. We also did an ablation study to understand the effect of non-repetitive gating and auto-regressive materials/thickness sampling.

Performance evaluation: In task 1 ultra-wideband absorber design, we measure the quality of the designed structure by *average absorption*. In task 2 incandescent light bulb filter, we calculate the visible light *enhancement factor* to measure the performance of designed structures.

3.4.1 Ultra-Wideband Absorber

Firstly, we apply our algorithm to the task of designing an ultra-wideband absorber for the wavelength range [400, 2000] nm. We choose the target spectrum as a constant 100% absorption under normal light incidence angle (i.e., the light is shining at the absorber at a right angle) to represent an ideal broadband absorber. This task has been previously studied by Yang et al. (*Yang et al.*, 2016) based on physical models, where the broadband absorption is achieved by overlapping multiple absorption resonances and with an overall graded-index structure to minimize reflection. The authors designed a 5-layer structure using MgF_2, TiO_2, Si, Ge, and Cr. The simulated average absorption of their structure over the wavelength range is 95.37% under normal incidence. If not specified otherwise, we assume normal incidence when reporting average absorption.

Table 3.1: Available materials for constructing the ultra-wideband absorber.

Ag	Al	Al_2O_3	Cr	Fe_2O_3	Ge	HfO_2	MgF_2
Ni	Si	SiO_2	Ti	TiO_2	ZnO	ZnS	ZnSe

We hypothesize that, when choosing from a larger set of materials than used in the previous work (*Yang et al.*, 2016), it is possible to design a structure with higher average absorption than the human-designed structure. Thus, we expanded the original material set (*Yang et al.*, 2016) to include 11 more materials (16 total). The set of materials is listed in Table 3.1. We set the available discrete thicknesses $\mathcal{D}$ to be $\{15, 20, 25, \ldots, 200\}$ nm with a total of 38 different values. When training the sequence generator, we set the learning rate to 5×10^{-5} and the maximum length to $L = 6$. The material embedding size d

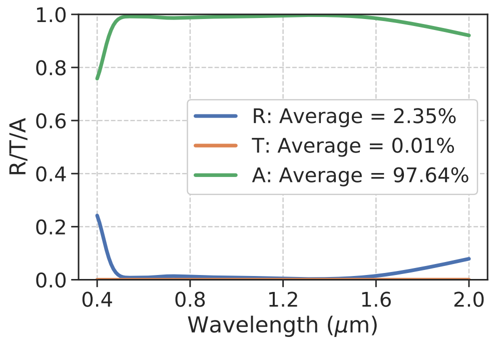
is set to 5, i.e., $emb_m \in \mathbb{R}^5$. The generator is trained for a total of $3,000$ epochs with the batch size set to be $1,000$ generation steps. We repeat the training for 10 runs with different random seeds. The best structure discovered in each run was recorded and finetuned using the quasi-Newton method.

Table 3.2: RL designed 14-layer structure with 99.24% average absorption.

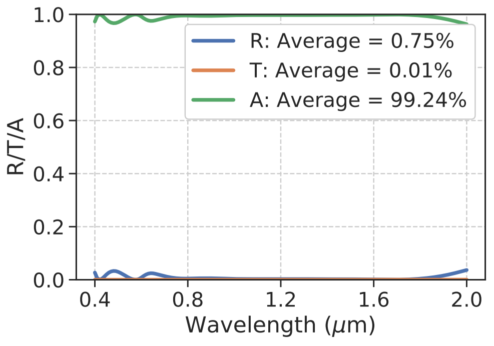
ID	**Material**	**Thickness**	**ID**	**Material**	**Thickness**
1	MgF_2	123 nm	8	Si	15 nm
2	TiO_2	32 nm	9	Cr	17 nm
3	MgF_2	21 nm	10	Ge	15 nm
4	Si	15 nm	11	TiO_2	33 nm
5	TiO_2	15 nm	12	Cr	29 nm
6	Si	15 nm	13	TiO_2	81 nm
7	Ge	15 nm	14	Cr	116 nm

It is worth noting that our algorithm can yield very similar structures as that reported in (*Yang et al.*, 2016), i.e., it can search for and find the structure designed based by human experts. One of such structures is {(MgF2, 112 nm), (TiO2, 55 nm), (Ti, 30 nm), (Ge, 30 nm), (Cr, 200 nm)} with an average absorption of 96.12%, which has exactly the same material composition as the one reported previously (*Yang et al.*, 2016). However, the best structure discovered by the algorithm, exhibiting a higher average absorption of 97.64%, is {(SiO2, 115 nm), (Fe2O3, 70 nm), (Ti, 15 nm), (MgF2, 124 nm), (Ti, 148 nm)}. The spectrum under normal incidence are plotted in Figure 3.5a.

We plot the best absorption values before and after finetuning of all ten runs in Figure. 3.6. After finetuning, the average absorptions for the discovered structures across all runs were improved. We found that the algorithm is robust to the randomness during training as 8 out of the 10 runs achieved an absorption that is higher than 95% after finetuning.



(a)



(b)

Figure 3.5: Normal incidence spectrum for the best discovered absorber structures with 5 and 14 layers. R: reflection, T: transmission, A: absorption. We design the multi-layer thin film to have high *absorption* over the entire wavelength range. (a) Normal incidence spectrum for the 5-layer structure. (b) Normal incidence spectrum for the 14-layer structure.

In an additional experiment, we explore whether the algorithm can design a structure with more layers to achieve even higher absorptions. We set the maximum length $L = 15$ and sample layer materials from MgF_2, TiO_2, Si, Ge, and Cr. The best discovered structure has 14 layers with an average absorption of 99.24%. The structure configuration is summarized in Table 3.2. We plot the normal incidence spectrum structure in Figure. 3.5b. The structure discovered by OML-PPO reaches close-to-perfect performance under normal incidence and has high absorption over a wide range of angles (Figure 3.7).

3.4.2 Incandescent Light Bulb Filter

To further test whether our method is scalable to more complicated tasks, we apply the proposed method for designing a filter that can enhance the luminous efficiency of incandescent light bulbs (*Zhou et al.*, 2016; *Ilic et al.*, 2016). The idea is to reflect the infrared light emitted by the light bulb filament so that

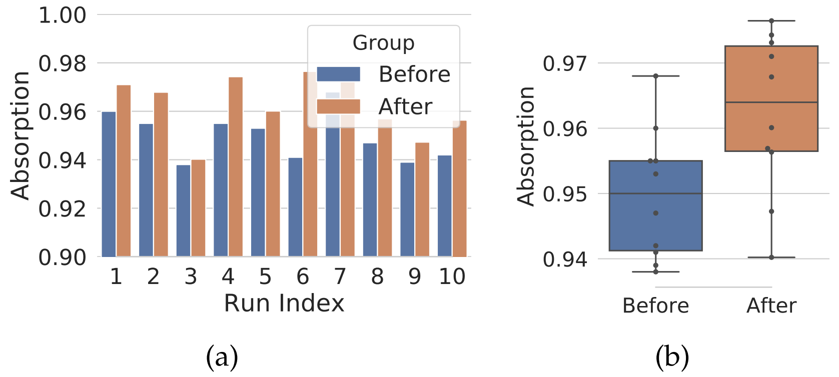


Figure 3.6: Absorption values before and after finetuning. finetuning improves the average absorption of every structure discovered in each run. (a) Average absorption values before and after finetuning for each individual run. (b) Box-plot for ten average absorptions values

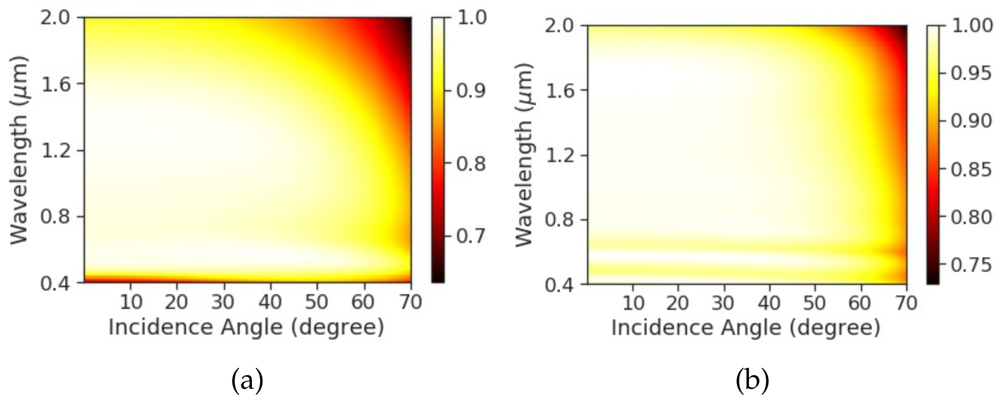


Figure 3.7: Angle-dependent absorption map for the best discovered absorber structures with 5 and 14 layers. Both achieves high absorption over a wide range of angles. (a) 5-layer structure. (b) 14-layer structure.

its energy can be recycled. To this end, we set the target reflectivity to be 0% in the range [480, 700] nm, and 100% outside this range (Figure. 3.8a). In this way, the infrared light, which cannot contribute to lighting, will be reflected back to heat up the emitter.

A similar design has been previously studied (*Ilic et al.*, 2016; *Shi et al.*, 2017). We choose the same seven dielectric materials as the available materials: Al_2O_3,

HfO_2, MgF_2, SiC, SiO_2, and TiO_2 (*Shi et al.*, 2017). Similar to our previous experiment, we train our policy for 10 runs with different random seeds. Here, we set the maximum allowed length $L = 45$ and the learning rate to be 5×10^{-5}. The number of epochs and batch size are 10,000 and 3,000, respectively. The best discovered structure is reported in Table 3.3.

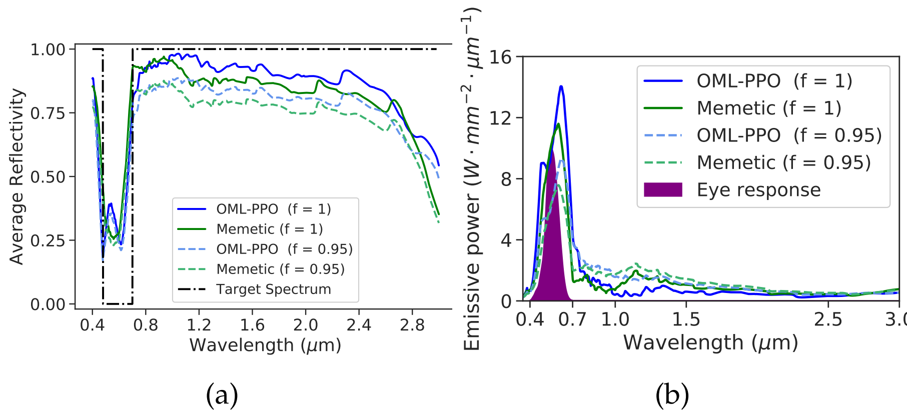


Figure 3.8: Results on the incandescent light bulb design. (a) Target spectrum and the average reflectivity of structures designed by OML-PPO and the memetic algorithm. (b) Emissive power spectrum. A good design will have high emissive power in the visible range [380, 780] nm. f is the view factor that equals the proportion of emitted light from the light bulb filament that can reach the light bulb filter. We report results under view factors 0.95 and 1.

For evaluating the performance of the design, we first calculated the angle averaged emissivity $\epsilon_{\text{avg}}(\lambda)$ over a hemisphere:

$$\epsilon_{\text{avg}}(\lambda) = \frac{2\pi \int_0^{\pi/2} \cos\delta \cdot \sin\delta \cdot \epsilon_{\text{eff}}(\lambda,\delta) d\delta}{2\pi \int_0^{\pi/2} \cos\delta \cdot \sin\delta d\delta} \tag{3.6}$$

$$= 2 \int_0^{\pi/2} \cos\delta \cdot \sin\delta \cdot \epsilon_{\text{eff}}(\lambda,\delta) d\delta, \tag{3.7}$$

where $\epsilon_{\text{eff}}(\lambda,\delta) = 1 - f^2 R(\lambda,\delta)$. $R(\lambda,\delta)$ is the reflection of the structure at wavelength λ under the incidence angle of δ. f is the view factor that equals to the proportion of the light from the emitter that can reach the filter. We compared two different view factors $f = 1$ and 0.95 in our calculation. In addition, we assume the light bulb operates at 100 W and the surface area of the emitter is equal to $Area = 20\ \mathrm{mm}^2$. Then, we can solve for the temperature t of the light emitter with the equation:

$$P_{\text{emitter}}\,(t) = Area \cdot \int I_{\text{emitter}}(\lambda, t)\epsilon_{\text{eff}}\,(\lambda) d\lambda, \tag{3.8}$$

where $I_{\text{emitter}}(\lambda, t) = \frac{2hc^2}{\lambda^5}\frac{1}{e^{hc/(\lambda k_B t)}-1}$ is the blackbody emission intensity spectrum. With view factor $f = 1$ (0.95), the OML-PPO designed filter leads to the emitter temperature of 3810 K (3553 K) while the structure designed by the memetic algorithm achieves a temperature of 3750 K (3498 K). The black body temperature under the same condition is calculated to be $t_0 = 2578$ K. We measure the enhancement factor by:

$$\chi = \frac{\int \epsilon_{\text{eff}}(\lambda) I_{\text{emitter}}(\lambda, t) V(\lambda) \mathrm{d}\lambda}{\int I_{\text{emitter}}(\lambda, t_0) V(\lambda) \mathrm{d}\lambda}, \tag{3.9}$$

where $V(\lambda)$ is the human eye's sensitity spectrum *Sharpe et al.* (2005). Our structure achieves an enhancement factor of 16.60 (10.67) while the memetic structure has an enhancement factor of 15.30 (9.72). The 42-layer structure designed by OML-PPO outperforms the previous 41-layer design by 8.5% (9.8%) in terms of the visible light enhancement.

In Figure 3.8, we compare the average reflectivity normalized over all incidence angles (0 - 90 degree) of the 42-layer structure designed with our algorithm and the 41-layer structure designed by a memetic algorithm (*Shi*

Table 3.3: RL designed incandescent light bulb filter with 42 layers. The total thickness is 8.54 µm.

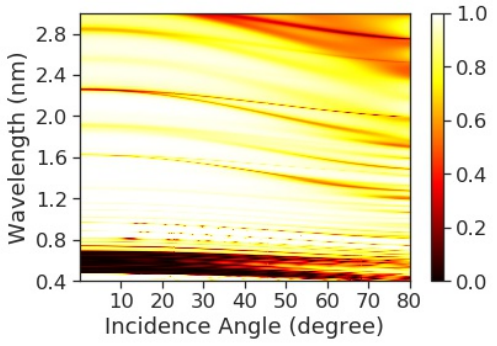
ID	Material	Thickness	ID	Material	Thickness	ID	Material	Thickness
1	SiO_2	289 nm	15	SiC	210 nm	29	SiC	117 nm
2	SiN	268 nm	16	SiN	168 nm	30	MgF_2	224 nm
3	MgF_2	185 nm	17	MgF_2	200 nm	31	SiC	122 nm
4	SiN	189 nm	18	SiC	227 nm	32	MgF_2	235 nm
5	SiC	214 nm	19	SiN	242 nm	33	SiC	127 nm
6	SiN	214 nm	20	MgF_2	222 nm	34	MgF_2	230 nm
7	MgF_2	210 nm	21	SiC	228 nm	35	SiC	234 nm
8	SiN	206 nm	22	MgF_2	216 nm	36	MgF_2	218 nm
9	SiC	205 nm	23	SiC	229 nm	37	SiC	235 nm
10	SiN	183 nm	24	MgF_2	203 nm	38	MgF_2	220 nm
11	MgF_2	184 nm	25	SiC	101 nm	39	SiC	231 nm
12	SiN	179 nm	26	MgF_2	209 nm	40	MgF_2	216 nm
13	SiC	203 nm	27	SiC	121 nm	41	SiC	233 nm
14	SiN	273 nm	28	MgF_2	225 nm	42	Al_2O_3	95 nm

et al., 2017). Our structure has a higher average reflectivity in the infrared range ($>$ 780 nm) than the 41-layer structure.

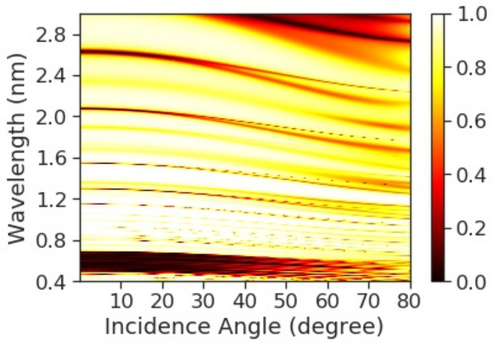
Table 3.4: Visible light enhancement. Our RL-designed structure achieved 8.5% higher visible light enhancement than the structure designed by a memetic algorithm.

Model	Enhancement factor
OML-PPO	**16.60**
Memetic (*Shi et al.*, 2017)	15.30

We quantitatively evaluated the performance of the designed filter by calculating the enhancement factor for visible light (400 - 780 nm) under a fixed operating power. The results are reported in Table 3.4. The performance improvement of the RL-designed filter can be attributed to the higher reflection than the structure designed by the memetic algorithm in the infrared wavelength range (Figure 3.9).



(a)



(b)

Figure 3.9: Angle-dependent reflection map for RL-designed incandescent light bulb filter structure and a structure designed by memetic algorithm. (a) structure designed by RL algorithm. (b) structure designed by memetic algorithm.

3.4.3 Chrome Coating Replacement Design

Lastly, we choose the highly task to demonstrate the effectiveness of OML-PPO, where we design an environmental-friendly five-layer optical thin film stack with an appearance similar to chrome, which could potentially replace the traditional highly toxic chrome plating process that poses great dangers to both workers and the environment. This is an important task because chrome coatings are widely used in the automotive industry for aesthetic purposes due to their pleasing appearance. Traditionally, the chrome layer is electroplated to the surface of the object by submerging it in a chemical electrolyte solution that contains hexavalent chrome (Cr (VI)). However, Cr (VI) is a strong human carcinogen and has been found to greatly increase the risk of lung cancer and nasal, and sinus cancer (*OSHA*, 2013). Cr (VI) can also cause severe nasal septum ulcerations and perforations, gastritis, and gastrointestinal ulcers (*Lieberman et al.*, 1941; *Lindberg and Hedenstierna*, 1983).

Workers can be exposed to Cr (VI) at the workplace from the mist generated during plating. The electroplating process can also lead to air pollution through the emissions of toxic air containing cadmium and cyanide, which could impact the nervous system, hearts, and lungs of millions of people (*EPA*, 2005). Thus, it is highly desirable to develop alternative solutions that can produce the chrome appearance but do not require the dangerous chrome plating process.

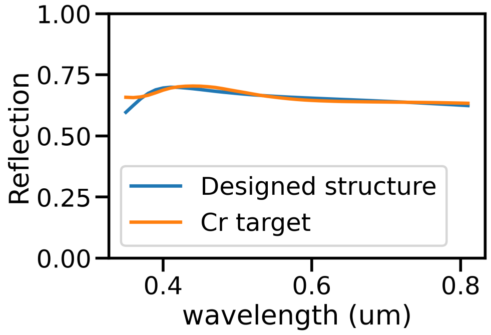
Table 3.5: The designed four-layer structure. The layers from I to IV are from the top surface where light is shone to the bottom layer.

Layer ID	**Material**	**Thickness**
1	Ge	17 nm
2	TiO_2	18 nm
3	SiO_2	82 nm
4	Ni	50 nm

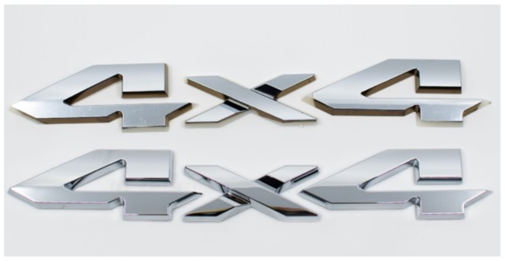
Here, we set the spectrum target to the reflection spectrum of a 100 nm thick Cr layer and optimize the designs with OML-PPO by constraining the maximum number of layers to be five. As shown in Figure 3.10, the designed structure (Table 3.5) has a reflection spectrum close to the Cr target reflection spectrum. The optical stack was fabricated through thermal evaporation (*George*, 1992) for experiment validation of the design. When compared to commercial Cr coatings (Figure 3.10b bottom row), the designed structure (Figure 3.10b top row) has an almost unnoticeable difference.

3.4.4 Ablation Study

On the ultra-wideband absorber design task, we conducted an ablation study to understand the effect of non-repetitive gating and auto-regressive generation of materials and thicknesses. We trained four different models:



(a)



(b)

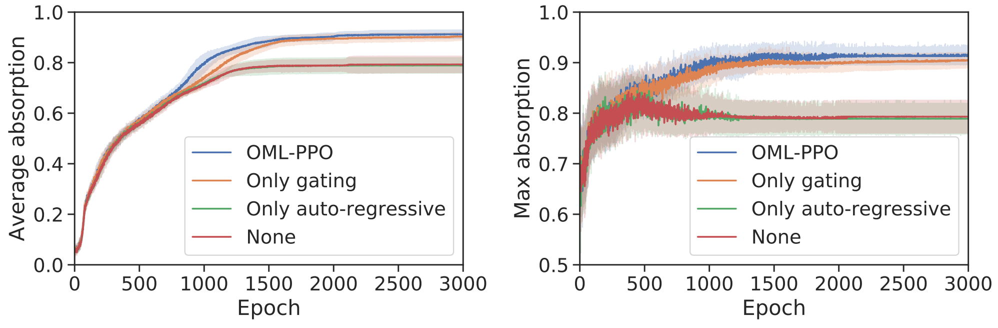
Figure 3.10: Chrome coating design results. (a) Reflection spectrum of the designed structure. (b) A photo for comparing the designed structure (top row) and the commercial Cr coating (bottom row).

1) OML-PPO with both non-repetitive gating and auto-regressive generation, 2) non-repetitive gating only, 3) auto-regressive generation only, 4) neither non-repetitive gating nor the auto-regressive generation. For each model, we repeated the training for ten times. The maximum absorption values discovered by each model before finetuning are reported in Table 3.6. Both non-repetitive gating and the auto-regressive material/thickness generation improve the performance of the baseline model.

Table 3.6: Highest absorption values discovered by each algorithm across 10 runs. The mean average absorption values and standard deviations of the 10 runs are reported.

Model	**Average Absorption**
OML-PPO	$\mathbf{94.98\% \pm 0.99\%}$
Only gating	$94.05\% \pm 1.39\%$
Only auto-regressive	$91.55\% \pm 1.14\%$
None (baseline)	$91.03\% \pm 0.87\%$

In Figure. 3.11, we plot the average absorption and maximum absorption of the structures generated in each epoch over the entire training trajectory.



(a) (b)

Figure 3.11: Training trajectory of OML-PPO and other baseline algorithms. (a) Average absorption trajectory. (b) Maximum absorption trajectory. The non-repetitive gating enables the model to converge to better solutions than models without the gating. The shaded area corresponds to one standard deviation.

The effect of non-repetitive gating is more significant than auto-regressive material/thickness generation as the OML-PPO and the only-gating variants both significantly outperform the other two variants. The non-repetitive gating significantly improves the model convergence during training. When non-repetitive gating and the auto-regressive sampling are combined together, the model achieves the best performance.

3.5 Conclusion

We introduced a novel sequence generation architecture and a deep reinforcement learning pipeline to automatically design optical multi-layer films. To the best of our knowledge, our work is the first to apply deep reinforcement learning to design multi-layer optical structures with the optimal number of layers not known beforehand. Using a sequence generation network, the proposed method can select material and thickness for each layer of a multi-layer structure sequentially. On the task of designing an ultra-wideband absorber, we demonstrate that our method can achieve high performance robustly. The algorithm automatically discovered a 5-layer structure with 97.64% average absorption over the [400, 2000] nm range, which is 2% higher than a structure previously designed by human experts. When applied to generate a structure with more layers, the algorithm discovered a 14-layer structure with 99.24% average absorption, approaching perfect performance. On the task of designing incandescent light bulb filters, our method achieves 8.5% higher visible light enhancement factor than a structure designed by a state-of-art memetic algorithm. Though the spectral requirements of our two examples are simpler than some other real-life applications(*Li et al.*, 2017), we expect no intrinsic difficulty when applying our algorithm to tasks that require more complicated spectra. Because the reward function used in our method can be easily calculated for any arbitrarily complicated spectrum, we believe that our algorithm can be directly applied to many other multi-layer thin film design tasks with more complex spectral requirements. Moreover, with the recent development of GPUs and TPUs, reinforcement learning algorithms could become more salable than evolutionary approaches for solving complicated design tasks.

Through an ablation study, we showed that customizing the sequence generation network based on optical design domain knowledge can greatly improve the optimization performance. Our results demonstrated the high performance of the proposed method on complicated optical design tasks. Because the proposed method does not rely on hand-crafted heuristics, we believe that it can be extended to many other multi-layer optical design tasks such as lens design and multi-layer metasurface design by modifying the action space of the sequence generation network. However, for complex designs that require micro-nano structures (*Li and Fan*, 2018), simulating the optical response can be computationally expensive. Since most deep reinforcement learning methods have a high sample complexity, it is important to develop sample-efficient reinforcement learning algorithms before such methods can be widely adopted for optical design tasks involving micro-nano structures.

CHAPTER 4

Neural Particle Swarm Optimization for Material-Aware Inverse Design of Structural Color

4.1 Introduction

Structural color refers to the color generated through the light interaction with patterned or layered optical structures. It is more stable than colors produced from chemical pigments, and serves as an environment-friendly alternative. However, designing the structures for producing desired colors is challenging due to the complex relationship between the optical structures and their spectral properties. Additionally, *color metamerism*, i.e., different spectra may correspond to the same color perceived by human eyes (*Foster et al.*, 2006; *Best*, 2017), makes the relationship between the structures and the perceived color more complex because multiple different structures could have the same color appearance. Human experts often design optical structures based on the understanding of the physical properties of structures, including multilayer thin films (*Wang et al.*, 2018b; *Yang et al.*, 2019) , metasurfaces (*Sun et al.*, 2017; *Yang et al.*, 2020), and self-assembled colloidal particles (*Kim et al.*, 2011; *Liu et al.*, 2021), to name a few. Due to the complex relationship between the structures and the generated color, the human-based design process is often slow and could lead to sub-optimal performance.

Recently, machine learning-based optical inverse design approaches have been developed to predict optical structures that can achieve user-specified properties (*Liu et al.*, 2018b; *Ma et al.*, 2019; *Unni et al.*, 2021; *Yeung et al.*, 2021b; *Yao et al.*, 2019; *Jiang et al.*, 2020; *Wiecha et al.*, 2021; *Ma et al.*, 2021). These inverse design methods often involve training a machine learning model such as deep neural networks (*Gao et al.*, 2019) or support vector machines (*Huang et al.*, 2019) on a curated dataset that contains a large number of datapoints mapping structural parameters to the corresponding color represented by coordinates in CIE xyY (*Gao et al.*, 2019) or LAB color space (*Sajedian et al.*, 2019a; *Dai et al.*, 2021). Though previous methods have been demonstrated to be efficient in designing a wide range of colors, they often require the materials constituting the optical structures to be fixed. Because the refractive index of materials affects their reflection and absorption properties, it could be challenging or even impossible to produce specific colors when the materials are not appropriately selected. Thus, the first step of screening appropriate materials for latter inverse design with machine learning models still requires intensive effort from human experts and could be time-consuming. For optical multilayer thin-film design, the recently reported reinforcement learning approach (*Wang et al.*, 2021) addresses the material selection challenge by searching the material and thickness design space simultaneously. However, this method can only design a single color at a time because the reward function for training the reinforcement learning algorithm has to be defined for a specific color. It could be impractical when many colors need to be designed, e.g., designing a large array of reflective color pixels to reconstruct a colored picture, which would take an extremely long time using the reinforcement learning method. To address the issues mentioned above, we propose an inverse structural color design method that can efficiently predict the materials and structural parameters.

The proposed method termed **Neu**ral Par**T**icle Swa**R**m **O**ptimizatio**N** (NEUTRON) is a hybrid approach that combines a **M**aterial-aware **M**ultitask **M**ixture **D**ensity **N**etwork (M^{3}DN) (*Caruana*, 1997; *Bishop*, 1994) and **P**article **S**warm **O**ptimization (PSO) (*Kennedy and Eberhart*, 1995). Instead of searching the material and thickness space simultaneously, NEUTRON first predicts the most suitable materials and provides a diverse set of initial guesses of the thicknesses in the form of probability distributions that could fulfill the target color, then applies particle swarm optimization to finetune the initial thickness designs. We demonstrate the effectiveness of the proposed approach on two optical multilayer thin film design tasks. The results show that our approach can lead to accurate color inverse designs efficiently. We believe that the proposed approach can be readily applied to many other optical design tasks where material selection and structural designs are important.

4.2 Related Work

The design task of optical structures for structural color generation is often high-dimensional. It may involve many degrees of freedom, including but not limited to the geometry of nanophotonic structures, number of layers, material selections, etc. In addition, different spectra could lead to the same perceived color by human eyes due to color metamerism (*Foster et al.*, 2006), which makes the high-dimensional color design task even more challenging due to the ambiguous many-to-one relationship between the design parameters and the generated color. Thus, manually designing optical structures for color generations could be time-consuming and may lead to sub-optimal designs. To speed up the optical structural color design process, intelligent inverse design algorithms that can automatically find the optical designs corresponding to specific color requirements have been previously applied for the design of multiple different types of optical structures for structural color generation (*Yang et al.*, 2013; *Wiecha et al.*, 2017; *Gao et al.*, 2019; *Baxter et al.*, 2019; *Dai et al.*, 2021; *Xu et al.*, 2021). These inverse design methods can be categorized into optimization-based approaches and machine learning-based approaches. In this part, we give detailed comparisons between these two different families of approaches for structural color inverse design based on case studies from previous works. In addition, we provide recommendations on how researchers should select inverse design methods to best suit their research needs. Such machine learning-based inverse methods can also be applied to many other photonic applications including photon detection (*Zhao et al.*, 2017), lasing (*Zhu et al.*, 2017), photonic integrated circuits (*Song et al.*, 2022), to name a few. We refer the readers to comprehensive reviews regarding machine learning methods for generic photonic design tasks (*Campbell et al.*, 2019; *Ma et al.*, 2021; *Wiecha et al.*, 2021; *Jiang et al.*, 2020).

All inverse design algorithms for metasurface-based structural color design take the color target potential design constraints as inputs and output the metasurface designs (Figure 4.1 (a)). However, optimization-based approaches

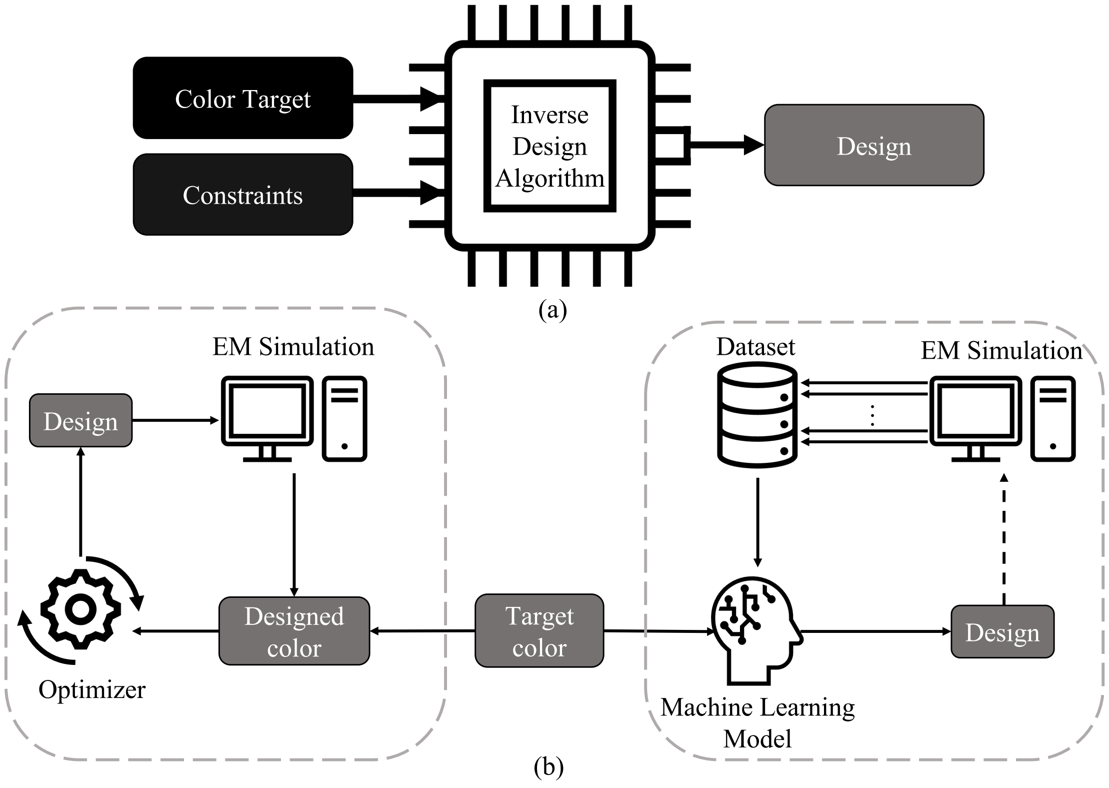


Figure 4.1: Sructural color inverse design paradigms. (a) An inverse design algorithm takes in the color design target and additional design constraints as the input to generate the design. (b) (left) Optimization-based inverse design approaches involve iterative evaluation of designs through EM simulations and update of the designs until an accuracy requirement is met. (right) Machine-learning based approaches require the users to first synthesize a dataset containing pairs of designs and color labels through EM simulations. A machine learning model is later trained on the synthesized dataset to predict the accurate design. When reinforcement learning is used instead of supervised learning, additional design data predicted by the model are added to the dataset to expand the explored design space for training a more accurate mode (indicated by the dotted arrow). After the model has been trained, machine learning-based inverse design process can directly predict designs corresponding to the desired color target without time-consuming iterative evaluations of the intermediate designs through EM simulations.

rely on iterative updates and evaluations of the designs through the feedback of EM simulations, and machine learning-based approaches train a model on a large training dataset to directly predict the design without iterative EM simulations during the design process.

Among all optimization algorithms, particle swarm optimization (PSO) and genetic algorithms (GAs) are most widely used for structural color designs. Both methods aim to maximize a merit function that measures the closeness between the properties of the designed structures and the target properties.

PSO is inspired by the group movement of bird flocks (*Kennedy and Eberhart*, 1995). To identify the global optimal that maximizes the merit function, PSO first creates a set of particles with randomly initialized positions, where each particle's position corresponds to a specific design. During the optimization process, each particle's position is updated with a velocity determined by the best position the particle itself has encountered and the best position found by all particles. The quality of each position is evaluated by EM simulations. After many rounds of iterations, all particles will converge to the same position that corresponding the final design found the PSO.

In (*Yang et al.*, 2013), the authors applied PSO for the design of color filters with high angular tolerance based on 2D gratings. To achieve this target, PSO is used to optimize the grating parameters including grating ridge, unit interval, and grating thickness. The merit function is a weighted sum of the difference between target reflection spectrum and reflection spectrum of the designed structure and the color difference between the color measured at normal incidence and 45°. The authors successfully obtained red, green, and blue filters with less than eight CIEDE2000 for incident angle up to 45°.

Similar to PSO, GAs also maintain a group of solutions (*Whitley*, 1994; *Deb et al.*, 2002) throughout the optimization process. However, GAs update the solutions not based on a velocity computed from previously found best solutions, but based on the survival of the fittest rule. Each iteration the GAs compute the fitness of each solution measured by the merit function value, and only select solutions with largest merit function values to proceed to the crossover step. During crossover, solutions are randomly paired to generate offspring solutions with a small probability of mutation, i.e., some design parameters are randomly changed by chance. The selection-crossover-mutation process of GAs stops when a convergence criterion is met.

In (*Liu et al.*, 2020), the authors applied genetic algorithms for meta-atom multiplications, i.e., including multiple meta-atoms within the same unit cell for metasurface designs. With the genetic algorithm aided meta-atom multiplication, the authors enlarged the achievable area in the CIE diagram by 106%. Genetic algorithms can also be applied for multiobjective optimization, where multiple potentially conflicting merit functions are being maximized at the same time. In (*Wiecha et al.*, 2017), the authors applied NSGA-II, a widely used multiobjective genetic algorithm for designing dielectric nanoantenna with double resonances at two different wavelengths for mutually crossed polarizations.

Machine learning based methods for photonic designs can be grouped into model-based optimization and reinforcement learning. Model-based optimization is based on supervised learning that trains a model on a fixed dataset without actively exploring the design space, while reinforcement learning does not require an initial dataset and can actively explore the design space through the feedback of a reward function that measures the quality of designs it has generated.

In model-based optimization, a surrogate forward model that can predict the color based on the design parameters is used to guide the design process. An inverse model that can directly predict the design parameters are also often trained to either provide the initial solutions for a search process or trained on a fixed forward model to directly predict the final solution.

In (*Huang et al.*, 2019), the authors trained four machine learning models including kernel ridge regression models to predict the CIE xy coordinates for dielectric ring arrays and dielectric pyramid arrays with different design parameters. The trained models are later used as a surrogate model for the computationally expensive EM simulation. With the surrogate model, the authors search the optimal design parameters that could lead to the target design with a simple greedy search process that iteratively updates each design parameter and evaluate the design's performance based on the surrogate model. Deep neural networks have also been trained as surrogate models to predict the structural color given fabrication parameters. In (*Baxter et al.*, 2019), authors trained neural networks to predict the RGB values based on laser machining parameters for producing plasmonic nanoparticles. A greedy search process is later applied to find the laser machining parameter that could produce the desired color.

Instead of relying on searching the input space of a trained forward model, *Gao et al.* (2019) applied a tandem network that combines the forward model and an inverse model to predict the metasurface designs for user-specified CIE xyY coordinates. Importantly, the forward model and the inverse model are trained in two separate steps. The forward model are used to guide the optimization of the inverse model while its own parameters being fixing during the training of the inverse model. The tandem network has recently been improved by using the Lab target to allow more uniform color design accuracy *Dai et al.* (2021), and introducing a mean squared error regularizers to avoid outputting infeasible designs (*Xu et al.*, 2021). In (*Roberts and Keshavarz Hedayati*, 2021),the authors augmented the labeled data with unlabeled synthetic data containing only the CIE xy coordinates to further improve the inverse design accuracy.

In addition to supervised learning, reinforcement learning (RL) has been applied to design silicon metasurfaces that can generate desired colors based on the CIE xyz coordintaes (*Sajedian et al.*, 2019a). RL is similar to conventional optimization-based approaches in the sense that both require iterative property evaluation. However, RL differs from conventional optimization-based approaches that a strategy is learned to map the color target to the optimal design parameters by intelligently explore the design space while conventional optimization-based approaches do not involve the learning process. As more data are encountered through the learning process, reinforcement learning can gradually learn a better strategy to efficiently come up with designs corresponding to a wide range of color targets. It has been shown that reinforcement learning based strategy can be applied to solve complicated optical design tasks (*Wang et al.*, 2021).

Both optimization and machine learning approaches can expedite the structural color design process. However, both have their pros and cons. When researchers choose the method that are most suitable to their design tasks, it is important to have good understanding of the differences between these two families of inverse design algorithms. The major difference between the optimization-based approach and machine-learning based approach is whether a model is trained on a dataset. Training a model allows one to efficiently predict for a new design target while optimization-based approaches always require a time-consuming iterative search process for every new design targets. However, collecting a large dataset for training accurate machine learning models is not trivial. If the researchers expect to only solve for a handful of designs, optimization-based inverse design approaches could be more practical. When many designs are needed (e.g., design structures for reconstructing all pixels in a high-resolution image), investing time in synthesizing datasets for training machine learning models would be worthwhile because the dataset generation process is a one-time cost. As more and more design targets are encountered, machine learning-based approaches tends to save more and more time than optimization-based approaches.

In addition to the consideration of efficiency, researchers should also consider the design accuracy of different inverse design methods. Because optimization-based approaches and reinforcement learning both involve iterative dataset collection that allows an active exploring the design space, they are often more accurate than the static model-based methods. Thus, if researchers value the efficiency more than accuracy, MBO approaches should be used for the design process while PSO, GAs, and RL are more suitable for obtaining highly accurate designs for a small set of color targets.

4.3 Methods

Unlike most previous machine learning-based inverse design work, where the datasets are synthesized with randomly sampled structural parameters of optical structures, we uniformly sample from the sRGB color target space to ensure good coverage of the entire color gamut. PSO is also used to obtain the datapoints to ensure high-quality designs to be included in the dataset. Additionally, we introduce a novel data augmentation approach that generates synthetic material refractive index data to broaden the input distribution that allows the trained neural network to general better. We also develop a novel multitask neural network that combines both a classification network and a mixture density network for material screening and layer thickness prediction, respectively. The chosen materials and predicted thickness distributions based on the multitask neural network are later used as the input for particle swarm optimization to further finetune the thickness for better approaching the color target. We present the dataset generation, neural network architecture and training, and the integration with PSO in this section. Later, we highlight the exceptional performance of the proposed approach with results on two applications, including 1) environmental-friendly chrome coating replacement design and 2) picture reconstructions.

4.3.1 Dataset Generation

Previous research shows that five-layer optical thin films with two absorbing layers sandwiched by two dielectric layers and a bottom metal reflecting layer (Figure 4.2 (a)) can achieve high color purity and brightness (*Yang et al.*, 2019),

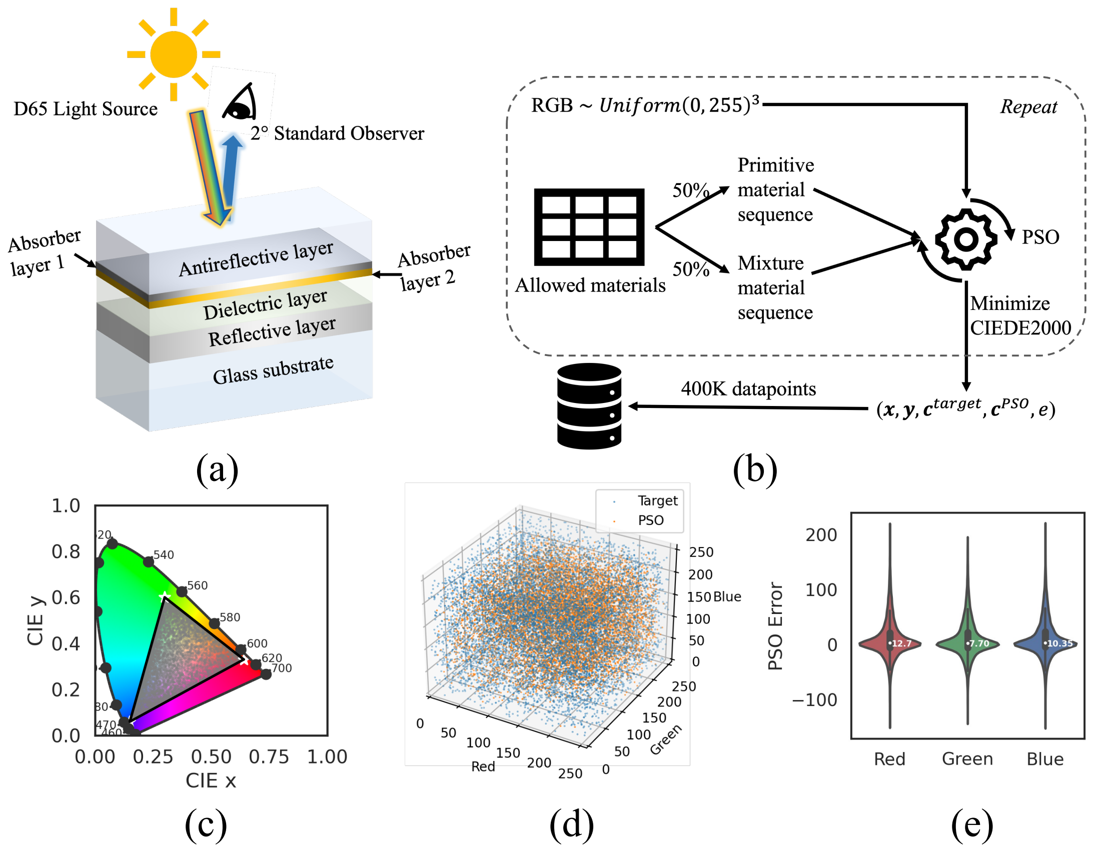


Figure 4.2: Five-layer optical structure and the data distribution. (a) A five-layer optical layer thin-film structure for generating a wide range of reflective colors covering the sRGB color gamut. Both the materials and their thicknesses are designed to obtain the target RGB color. (b) The dataset generation process. 400,000 datapoints are obtained through PSO for randomly sampled RGB color target and material combinations. (c) The color distribution of the validation set is visualized in the CIE 1931 xy space. The datapoints achieve good coverage of the entire sRGB color gamut spanned by the standard Red, Blue, and Green colors. (d) Randomly sampled RGB color target and the obtained RGB color through PSO. (e) RGB value design error in the validation set obtained with PSO through the data collection process.

where the layers can be easily deposited by physical vapor evaporation. Due to the high performance and feasibility for large-scale fabrications of such structures, we synthesize a dataset with diverse designs based on the same five-layer structural template by varying both the material and thickness of each layer. All designs are based on randomly sampled materials from ten candidate metal materials Au, Ag, Al, Cu, Cr, Ge, Ni, Ti, W, Zn and ten dielectric materials Al_2O_3, Fe_2O_3, HfO_2, MgF_2, SiO_2, Ta_2O_5, TiO_2, ZnO, ZnS, ZnSe. Both absorber layers and the bottom reflective layers are composed of metals while the other two layers are based on dielectric materials. Including a wide range of candidate materials with different refractive indices makes it possible to search for the most suitable materials combinations for specific color targets.

When sampling the materials for the adjacent absorbing layers, we introduce a constraint that the two adjacent absorber layers must be composed of different materials. Thus, the total number of unique material combinations of the five-layer stack is $10 \times 10 \times 9 \times 10 \times 10 = 9 \times 10^4$. Because sRGB is widely used in display industry design and production, we aim to provide the best coverage over the sRGB color gamut through our model. Most previous works randomly sample the layer thicknesses from a prefixed range to generate datapoints with different designs and color properties. However, this approach often leads to non-uniform coverage in the color space, i.e., the density of datapoints in a certain color regime is higher than other regimes. Since non-uniform data coverage could lead to an undesirable skewed inverse design performance where the inverse design model is more accurate for colors corresponding to the color regime with more datapoints, we propose to directly sample from the color target space uniformly to avoid this problem. To this end, instead of randomly sampling the thicknesses of the layers as in most previous works, we randomly sample the sRGB color target from the 3D sRGB space with a value range [0, 255] for each dimension. In Figure 4.2 (c) - (d), we show that this random color target sampling ensures uniform coverage of the color space.

Then, for each randomly sampled material combination and color target pair, we use PSO to optimize the thickness of the top four layers to minimize the color difference between the color of the designed structure and the color target while fixing the bottom reflective layer at 100 nm. PSO is a widely used global optimization algorithm for optical designs (*Shokooh-Saremi and Magnusson*, 2007; *Yang et al.*, 2013; *Rabady and Ababneh*, 2014). When solving a minimization problem, PSO maintains a group of particles (i.e., solutions) that individually explore the solution space and communicate with each other to share information about the explored solution space. All particles' positions are iteratively updated based on the best solution each particle has found and the best solution found by the entire group until a convergence criterion is met. In our five-layer optical thin film design task, each particle's "position" is a 4D vector corresponding to the top four layers' thickness. The final position all particles converge to is the final thickness design obtained through PSO. Throughout the optimization process, the reflection spectrum of multilayer designs is computed with transfer matrix method (*Byrnes*, 2016) for the wavelength range [400, 700] nm with a step size of 10 nm. Thus, each material's complex refractive index is described by a 62-dimensional vector for the entire 31 wavelength points. We use the Python package Colour (*Mansecal et al.*, 2021) for the conversion of reflection spectrum to sRGB and Lab color coordinates. Additionally, we use the industry-standard CIEDE2000 metric based on Lab coordinates to measure the color difference (*Sharma et al.*, 2005) between the target color and the color obtained through the designs. After particle swarm optimization converges, if CIEDE2000 for the given material combination and the color target is lower than or equal to two, we consider that the specific material combination allows accurate generation of the target color because CIEDE2000 lower than or equal to two is almost imperceivable by untrained people with normal visions (*ViewSonic*, 2021), and we assign a label $e = 1$; otherwise the label is $e = 0$. Overall, we observe that the average success ratio of obtaining the accurate color following the random data generation process is about 25%. Note that the Lab color coordinates and the sRGB coordinates for the same design have a one-to-one correspondence, and we only use Lab color coordinates for computing CIEDE2000. In contrast, sRGB coordinates are used explicitly as the color target. If users need to design specifically for a given Lab or CIE xyY target, sRGB coordinates can be uniquely obtained from the provided color coordinates defined on other color spaces.

In terms of the thickness of each layer, we set the thickness range for both dielectric layers from the range [5, 250] nm and the thickness of the absorbing layers to be in the range [5, 15] nm during the particle swarm optimization process. Again, the thickness of the bottom metal layer is fixed to be 100 nm because it is used as a reflector, whose thickness has a negligible effect on the reflective color as long as the layer is thick enough to reflect the light completely. Additionally, we constrain the thicknesses of all layers to be integers to allow fabrications. Thus, the size of the design space after considering variations in materials combinations and thickness designs is $9 \times 10^4 \times 11^2 \times 246^2 = 6.6 \times 10^{11}$.

Since the small number of primitive materials limits variations in the refractive index data, we augment the dataset by randomly mixing two dielectrics or two metals to form synthetic dielectric or metal composite materials to broaden the training dataset distribution through convex combination of their complex refractive indices. Inspired by our previous finding that the mixture of two thin material layers can be considered as a linear combination of the two materials and helpful to improve color purity (*Yang et al.*, 2019), we obtain the complex refractive index of synthetic mixture materials through the following formula:

$$\boldsymbol{n}_i = \beta_i \cdot \boldsymbol{n}_{\mathcal{M}_{1,i}} + (1-\beta_i) \cdot \boldsymbol{n}_{\mathcal{M}_{2,i}}, \tag{4.1}$$

where $\mathcal{M}_{1,i}$ and $\mathcal{M}_{2,i}$ are the two sampled materials for ith layer, $\beta_i \sim (0,1)$ is the mixing factor for the ith layer, $\boldsymbol{n}_{\mathcal{M}_{1,i}}, \boldsymbol{n}_{\mathcal{M}_{2,i}}$, and $\boldsymbol{n}_i$ are the complex refractive indices for randomly selected two primitive materials and the composite material synthesized from them for the ith layer. Note that, the composite material refractive index is synthetic and does not require to be experimentally achievable. It only serves as additional data to improve the data variation so a machine learning model can learn more efficiently from the data. Similar data augmentation strategies have been widely adopted in other fields (*Zhang et al.*, 2018). Following the described data generation procedure, we generated a total of 400,000 datapoints and about half of the samples are based on primitive (i.e. actual) materials while the others are based on synthetic materials via linear superposition. The entire dataset is randomly split into 380,000/10,000/10,000 datapoints for training/validation/testing. Importantly, both validation and test sets are based on primitive materials only because we are interested in the model's design accuracy for designs based on attainable actual materials, and mixtures materials are introduced only in the training set for the data augmentation purpose. Unless stated otherwise, all results reported in the paper are based on the test sets. Each datapoint $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c}^{target}, \boldsymbol{c}^{PSO}, e)$ is a tuple comprising of the refractive index data for all five layers concatenated as a single vector $\boldsymbol{x}$ with 310 dimensions (each material's complex refractive data is a 62-dimensional vector), the thickness of top four layers $\boldsymbol{y}$ obtained with PSO, the target color sRGB color $\boldsymbol{c}^{target}$, the obtained color through PSO $\boldsymbol{c}^{PSO}$, and the binary label $e \in [0, 1]$ indicating whether the CIEDE2000 between $\boldsymbol{c}^{target}$ and $\boldsymbol{c}^{PSO}$ is smaller than or equal to 2. We transform continuous variables $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c}^{target}, \boldsymbol{c}^{PSO})$ to the range [-1, 1]. A pictorial illustration of the full data generation pipeline is included in Figure 4.2 (b).

4.3.2 Material-Aware Multitask Mixture Density Networks

Mixture density networks (MDNs) have been previously applied for inverse problems extensively (*Bishop*, 1994; *Li and Lee*, 2019; *Unni et al.*, 2021). Instead of mapping the input to a single output, which is done by deterministic neural network models, an MDN maps the input to the probability density function of a multivariate Gaussian Mixture with m isotropic Gaussian mixture components. Because each Gaussian mixture component can learn a different mean and standard deviation value, MDNs are able to learn a one-to-many mapping, which is critical to solving the often ill-posed inverse design problems with non-unique solutions. The probability density function given by a Gaussian mixture for a pair of refractive data input, thickness design, and corresponding

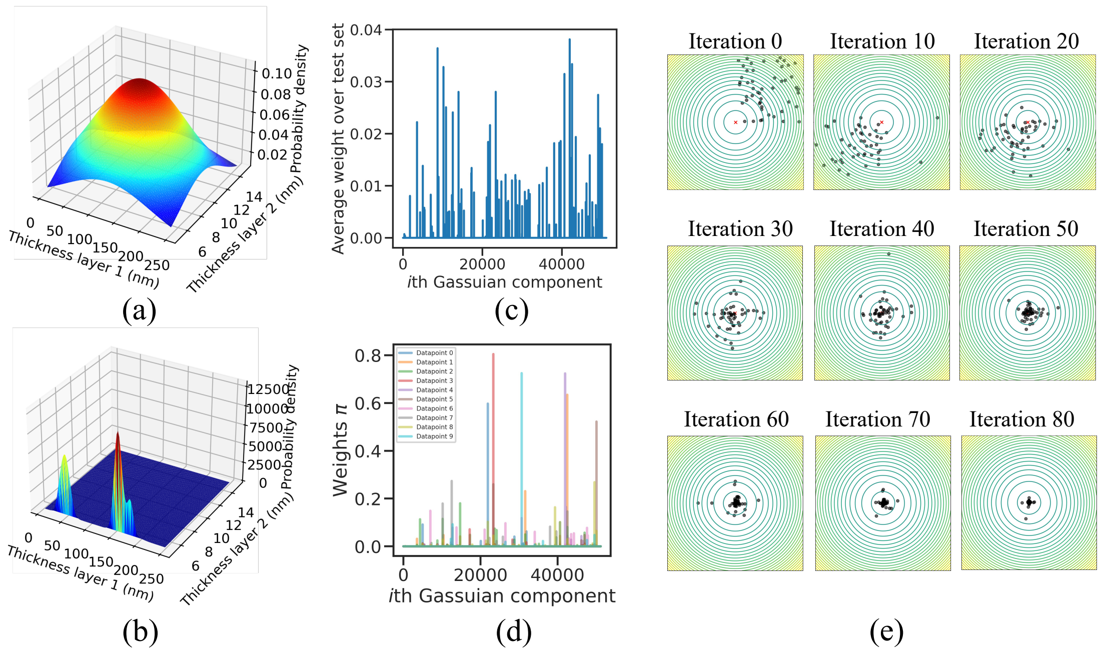


Figure 4.3: (a) Probability density function for the first two layers' thickness from a randomly initialized MDN. (b) The trained MDN predicts the probability density function for the top two layers' thickness. The probability density function converges to two modes after training, which indicates the training process has learned the corresponding thickness ranges are the most promising for producing the target color. (c) Average weights for all Gaussian mixture components. (d) Gaussian mixture weights for ten different inputs $(\boldsymbol{x}, \boldsymbol{c}^{target})$. The predicted weight vary based on the input data $(\boldsymbol{x}, \boldsymbol{c})$, which shows that different Gaussian mixture components are responsible for different regimes of the input space. (e) An illustrative example of the PSO process. The plots show the optimization trajectory of PSO for a simple 2D square function based on 50 randomly initialized particles. The global minimal of the function is located at the center of the 2D space. In the five-layer optical thin film design problem, each particle is initialized with the trained MDN in a 4D space (excluding the last layer with fixed 100 nm thickness) and optimized with PSO iteratively until convergence.

color $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{c})$ can be written as :

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{c}) &= \sum_{j=0}^{m-1} \boldsymbol{\pi}(\boldsymbol{x})_j p_j(\boldsymbol{x})(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{c}) \\ &= \sum_{j=0}^{m-1} \boldsymbol{\pi}_j \frac{1}{(2\pi)^{\frac{D}{2}} \prod_{d=0}^{D-1} \boldsymbol{\sigma}_{j,d}(\boldsymbol{x})} e^{-\left(\sum_{d=0}^{D-1} \frac{\left(y_d - \mu_{j,d}(\boldsymbol{x})\right)^2}{2\sigma_{j,d}^2(\boldsymbol{x})}\right)}, \end{aligned} \tag{4.2}$$

where $\boldsymbol{\pi}(\boldsymbol{x}), \boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{\sigma}(\boldsymbol{x})$ are the mixing weights, mean, and standard deviation outputted by the MDN with the input refractive data $\boldsymbol{x}$ and the color target $\boldsymbol{c}$. D is the dimension of the design parameter vector $\boldsymbol{y}$, which equals 4 in our design problem. Note that the mixing weights sum up to one (i.e., $\sum_{j=0}^{m-1} \boldsymbol{\pi}_j = 1$) so that the mixed function $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{c})$ is a proper probability density function.

In addition to predicting the thickness, we also predict whether a given refractive index data $\boldsymbol{x}$ can lead to accurate designs for a target color $\boldsymbol{c}$ with no greater than two CIEDE2000 (i.e., $\Delta E_{00} \leq 2$) after the thickness of each layer has been optimized. To this end, we include a sub-network to form a material-aware multitask mixture density network (M^{3}DN) that is composed of a mixture density network and a classification network (Figure 4.4 (a)).

To train the M^{3}DN, we minimize the multitask loss function:

$$\mathcal{L}_{M^3DN} = \sum_{i=0}^{N-1} - \log p(\boldsymbol{y}_i|\boldsymbol{x}_i, \boldsymbol{c}_i^{PSO}) + \alpha \cdot \mathrm{BCE}(e_i, p_{\Delta E_{00} \leq 2}(\boldsymbol{x}_i, \boldsymbol{c}_i^{target})) \qquad (4.3)$$

where the first term on the right hand side is the negative log likelihood for training the MDN, and the second term is the binary cross entropy loss for training the classifier, i.e.,

$$\begin{aligned}\mathrm{BCE}(e_i, p_{\Delta E_{00} \leq 2}(\boldsymbol{x}_i, \boldsymbol{c}_i^{target})) = & - e_i \cdot \log p_{\Delta E_{00} \leq 2}(\boldsymbol{x}_i, \boldsymbol{c}_i^{target}) \\ & - (1 - e_i) \cdot \log (1 - p_{\Delta E_{00} \leq 2}(\boldsymbol{x}_i, \boldsymbol{c}_i^{target})).\end{aligned}$$

The hyperparameter α in the M^{3}DN loss controls the knowledge sharing among the material screening task and the thickness prediction task. Note that, $\boldsymbol{c}_i^{target}$ is used as input for training the MDN while $\boldsymbol{c}_i^{PSO}$ is used as input for training the classifier because the thickness $\boldsymbol{y}_i$ corresponds to $\boldsymbol{c}_i^{PSO}$, which could differ slightly from the $\boldsymbol{c}_i^{target}$ in the generated dataset because PSO may only be able to find solutions with a color coordinate slightly different from the target color, i.e., $\boldsymbol{c}_i^{PSO} = \boldsymbol{c}_i^{target} + \delta_i$.

Through an extensive hyperparameter search based on the CIEDE2000 measured on the validation set, we found a seven-layer neural network with four separate output heads to give the best inverse design accuracy, which is shown the in Figure 4.4 (a). The number of of Gaussian mixture components is 51,200. We visualize the learned probability density function and the mixing weights in Figure 4.3 (a) - (d). Note that we use the *softplus* activation function for the $\boldsymbol{\sigma}$ output head to ensure the standard deviation prediction is always greater than 0, and *softmax* activation function is used for the mixing weight $\boldsymbol{\pi}$ output so that all mixing weights sum up to 1. We use *ELU* activations for all other layers except the last layer in the base network and the output layer for μ, where *tanh* is used to ensure their outputs are in the range [-1, 1].

The advantage of training a single multitask network for both classification and thickness prediction tasks is two-fold: 1) the classification module allows users to screen among possible material combinations to select those that could lead to an accurate generation of the target color; 2) improved sample efficiency through sharing knowledge among the related classification and thickness prediction tasks.

The test results for the classification AUC (area under the receiver operating curve, or AUROC) and the inverse design accuracy in terms of CIEDE2000 for both the target color and the PSO obtained color are shown in Figure 4.4. The material classification network achieves excellent performance with an AUC of 0.91. Both average test CIEDE2000 values are slightly above ten and can be further improved by finetuning with the PSO.

We train the M^{3}DN with PyTorch (*Paszke et al.*, 2017) on NVIDIA RTX 3090 GPUs with 24 GB of internal memory. Both our code and data are publicly available [1].

Based on the downstream task, the material screening module can be used in various ways with the predicted probabilities $p_{\Delta E_{00} \leq 2}$. On one hand, when the goal is to select the best material corresponding to a single color (e.g. in the first example below searching for chrome color replacement), the top K material combinations with largest $p_{\Delta E_{00} \leq 2}$ among all possible combinations can be chosen for further examinations of their performance. On the other hand, when researchers have a set of target colors to produce (e.g. in the second example of reproducing a color picture), the average $p_{\Delta E_{00} \leq 2}$ over the set of target colors can be computed for ranking and selecting the material combinations.

Given the selected materials, the thickness designs are predicted based on the color target and the material index data. However, since the output of the mixture density network is probabilistic, it is possible that the thickness output by the trained mixture density network does not correspond to the optimal value. To address this issue, we further finetune the designs with PSO. Combining the dataset generation process, M^{3}DN, and the PSO finetuning process completes the proposed NEUTRON method for structural color design.

In all of our experiments, for each material combination, we randomly sampled 32 thickness designs from the MDN to be used as the initial positions

[1] `https://github.com/hammer-wang/NEUTRON`

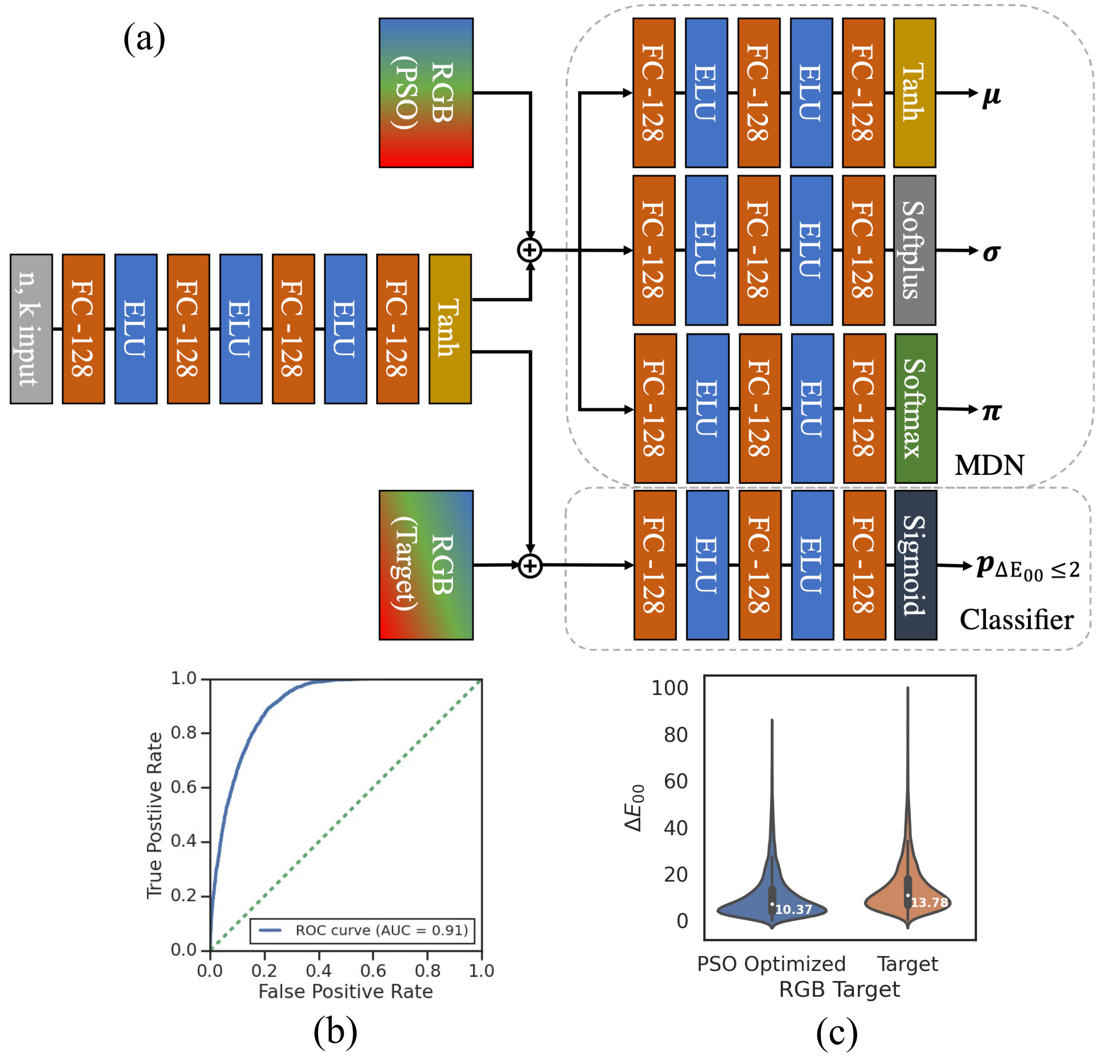


Figure 4.4: Material-aware multitask mixture density network (M^{3}DN). (a) M^{3}DN architecture with a shared base network for extracting features from the high dimensional refractive index data, and four subnetworks for outputting the $\mu, \sigma, \pi, p_{\Delta E_{00} \leq 2}$, separately. (b) Mixture density for a two-dimensional design space. (c) Receiver operating characteristic curve. (d) Design accuracy measured in CIEDE2000 with respect to both PSO optimized RGB colors and the corresponding original RGB color targets.

of the particle swarm optimization. Then, we optimize until convergence with a tolerance level of 1×10^{-5}. In each optimization iteration, the designs are evaluated through the transfer matrix method (*Byrnes*, 2016) and compared with the target spectrum. Compared to randomly initializing the positions of the particles, starting with designs sampled from the MDN allows PSO to find better solutions. More importantly, conventional PSO requires the materials to be provided by the user while NEUTRON predicts the best materials to be used with the PSO. Since the manual material screening process could be time-consuming, using NEUTRON for structural color design can significantly speed up the entire design process by directly predicting the best materials.

4.3.3 Model Implementation Details

For training the M^{3}DN, we random search hyperparameters from the range listed in Table. 4.1. The model with the best CIEDE2000 on the validation set has a learning rate of 0.000176, hidden dimension 128, the number of mixtures 51,200, weight decay of 0.0000117, and $\alpha = 5$. We train the model on the training set for 500 epochs and store the model with the model checkpoint with the best validation CIEDE2000. Our reported results in the main text are all based on the best model unless stated otherwise.

Table 4.1: Hyperparemeter search values for training the M^{3}DN model.

Hyperparameters	Values
Learning rate	loguniform$(10^{-5}, 10^{-3})$
Hidden dimension of FC layers	[128, 256, 512, 1024]
Number of mixtures	[200, 400, 800, 1600, 3200, 6400, 12800, 25600, 51200]
Weight decay factor	loguniform$(10^{-6}, 10^{-4})$
α	[1, 2, 5, 10]

In particle swarm optimization, each particle's position at time t is updated with a velocity that is determined by both the best solution found by the particle itself and the best solution found by the entire group. In the optical multilayer thin-film design task, the position vector x is the thickness designs for all layers. The velocity v is the update to each layer's thickness between two optimization steps. The velocity is calculated as:

$$\mathbf{v}_i^{t+1} = w \cdot \mathbf{v}_i^t + c_1 \cdot r_1 \cdot (\mathbf{x}_{i,best}^t - \mathbf{x}_i^t) + c_2 \cdot r_2 \cdot (\mathbf{x}_{best}^t - \mathbf{x}_i^t), \tag{4.4}$$

where $\mathbf{x}_{i,best}^t$ is the best solution found by the ith particle by time step t, while $\mathbf{x}_{best}^t$ is the best solution found by the entire group of particles by time step t. r_1, r_2 are two random numbers in the range [0, 1]. w is the inertia coefficient that controls how much the previous velocity is maintained, c_1 is the cognitive coefficient that controls how much the best solution found by the particle itself contributes to the velocity, and c_2 is the social coefficient that determines how the best solution found by the entire group alters the velocity. In our implementation, we set the $w = 0.9, c_1 = 0.5, c_2 = 0.3$ based on the results from a manual search. After the velocity is obtained at the $(t+1)$th step, we update the particle's position by:

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1}. \tag{4.5}$$

4.4 Experiments

4.4.1 Chrome Coating Replacement Design: A Revisit

Here, we apply the NEUTRON method to the chrome coating replacement design task again to explore whether different designs from those presented in

Chapter 3 can be discovered.

In the material screening phase, we exclude Cr from the available metals, and predicted the likelihood $p_{\Delta E_{00} \leq 2}(\boldsymbol{c}^{target}|\boldsymbol{x})$ for all $10 \times 9 \times 8 \times 10 \times 10 = 720,000$ possible material combinations. Then, we sample initial thickness designs from the MDN and finetune the designs with PSO for the top 100 combinations with the largest likelihoods. The emulated color of the five best designs are shown in Figure 4.4. We also provide the detailed designs and CIEDE2000 of all five designs in Table 4.2. PSO significantly improved the initial designs predicted by the MDN in terms of the CIEDE2000. All five final designs obtained CIEDE2000 lower than two and are highly promising for replacing the traditional Cr plating process through the environmental-friendly thermal evaporation process. Moreover, the discovered designs all have dielectric materials as the top materials, which is different from the design in Chapter 3. This top dielectric layer could provide a protection to the bottom metal layers from environmental degradation, thus making the designs presented here more practical than designs in Chapter 3 (Table 3.5).

Table 4.2: Chrome color designs based on M^{3}DN only and the finetuned design by PSO (**bold**).

Design	Antireflective Layer (nm)	Absorber Layer I (nm)	Absorber Layer II (nm)	Dielectric Layer (nm)	Reflective Layer (nm)	ΔE_{00}
I	HfO_2 120 / **105**	W 13 / **15**	Ge 15 / **10**	SiO_2 122 / **83**	Zn 100 /**100**	7.6 / **0.8**
II	HfO_2 12 / **7**	Zn 14 / **14**	Ge 8 / **13**	Al_2O_3 160 / **98**	Ni 100 / **100**	15.8 / **1.2**
III	SiO_2 273 / **250**	Ag 14 / **14**	Ge 15 / **15**	SiO_2 147 / **130**	Au 100 / **100**	9.8 / **1.4**
IV	HfO_2 61 / **5**	Zn 14 / **14**	Ge 15 / **15**	Al_2O_3 202 / **207**	Au 100 / **100**	25.0 / **1.6**
V	HfO_2 97 / **93**	Zn 15 / **15**	Ge 12 / **11**	MgF_2 249 / **218**	Au 100 / **100**	5.4 / **1.6**

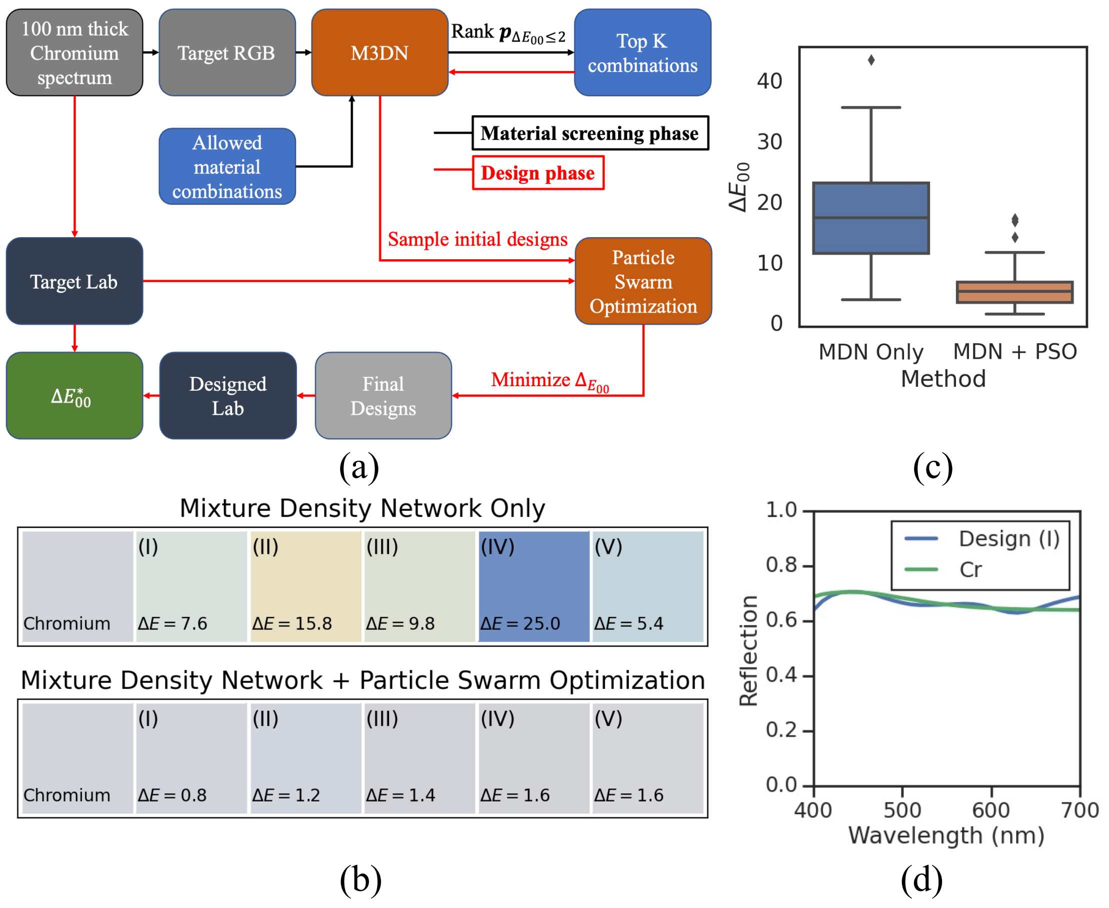


Figure 4.5: Single color design pipeline and chrome color design results. (a) The top K material combinations obtaining the target color is obtained through ranking the predicted probability vector $p_{\Delta E_{00}}$ for all allowed material combinations. Then a set of initial designs are sampled from the M^{3}DN as the initial solutions for particle swarm optimization to finetune the thickness of each layer. (b) Top five designs based on the final CIEDE2000 values. The particle swarm optimization step leads to significant performance improvement compared to the designs sampled directly from the mixture density network. (c) CIEDE2000 before and after the particle swarm optimization finetuning for the top 100 material combinations. (d) The reflection spectrum of the best design and a 100 nm thick Cr film.

4.4.2 Picture Reconstruction

Next, we apply NEUTRON to design optical thin film structures for a large set of color targets. We choose the picture reconstruction task (*Gao et al.*, 2019; *Dai et al.*, 2021) because it has been previously studied in machine learning-based structural color inverse design works. High-resolution pictures generally have more than $\sim 10^5$ unique pixel RGB values, which makes it impossible to directly reconstruct when iterative optimizations are is involved. In our work, the particle swarm optimization process for finetuning a single pixel requires ~ 10 seconds, and reconstructing the entire picture takes $\sim 10^6$ seconds. To speed up the reconstruction, we apply a quantization approach to group similar pixels values to the nearest integer values that can be divided by a quantization step size s. We explored quantized step size $s \in [5, 10, 20, 30]$ and found the step size $s = 10$ led to the best tradeoff between the compression ratio of unique pixels values and the quality of the quantized picture. In 4.6, we compare the pictures before and after quantization with a step size 10. With a negligble difference in picture appearance, we achieved more than **40X** compression ratio of unique pixel values for all pictures.

The detailed design pipeline is provided in Figure 4.5 (a), where the top material combination is selected based on the average $p_{\Delta E_{00} \leq 2}$ computed over the entire picture to be reconstructed. Note that we select the same material combination of all pixels across the entire picture to make it possible for fabrication through grayscale lithography (*Wang et al.*, 2018b). We plot average $p_{\Delta E_{00} \leq 2}$ distributions for all allowed material combinations in Figure 4.5 (b), which show that only a few material combinations can lead to high average

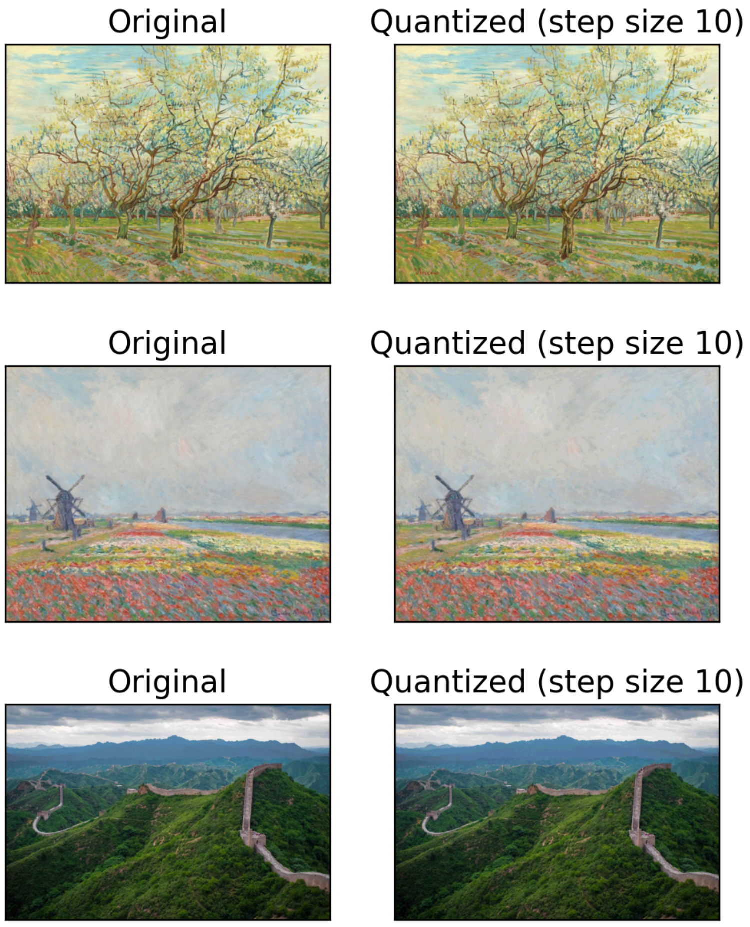


Figure 4.6: Picture quantization results. All pictures are 512-pixel wide. The quantization process does not lead to noticeable difference but significantly reduces unique pixel values. The White Orchard picture's unique pixel values were reduced to 2,428 from 156,778. The Tulip Field picture's unique pixel values dropped to 2,279 from 96,906. The Great Wall picture's unique pixel values decreased to 1,395 from 67,007.

success design rates and also demonstrate the importance of the material classification network. Again, in Figure 4.5 (c), we show that PSO significantly improved the average design accuracy upon the initial designs based on the trained MDN.

As another demonstration, we applied our method to pictures obtained

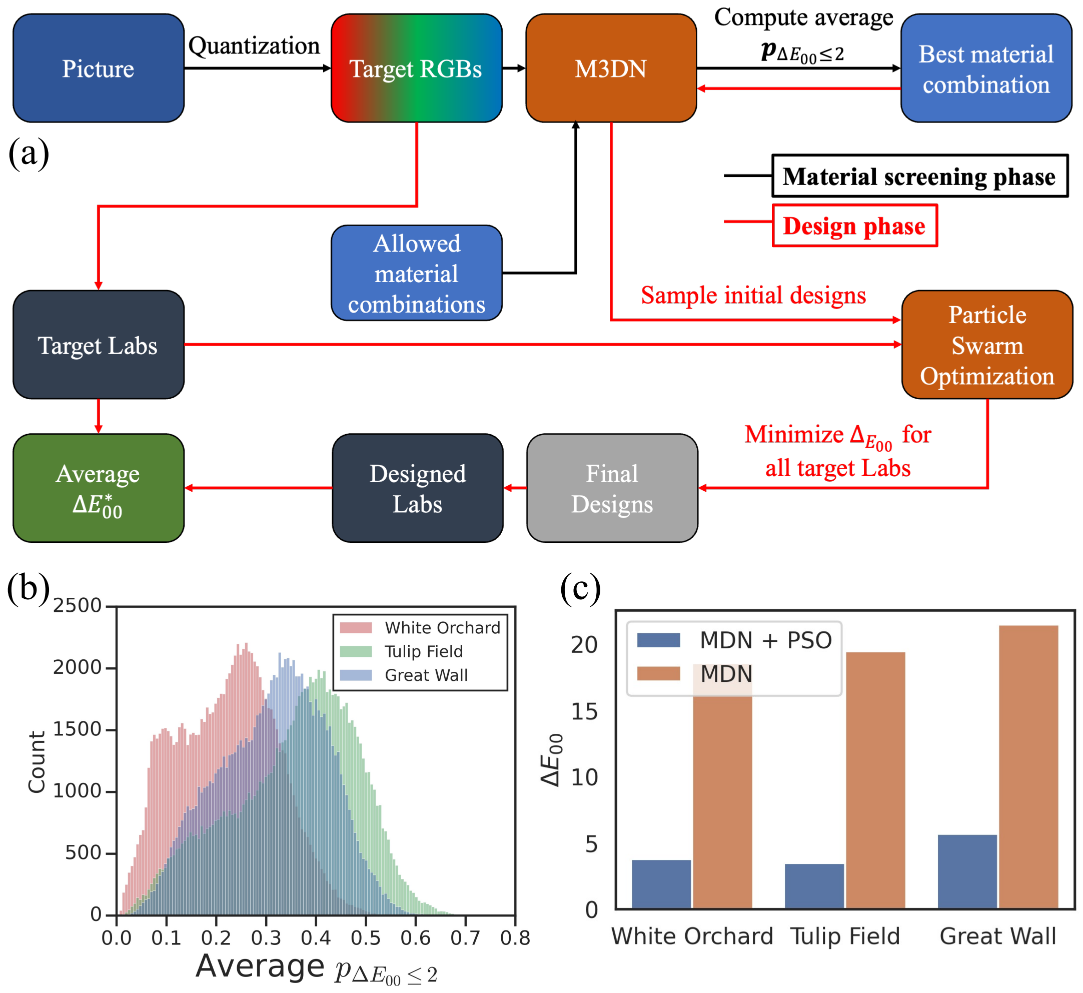


Figure 4.7: Picture reconstruction design pipeline. (a) Similar to the design pipeline of a single color, the material screening is the first phase of the design process for identifying the best material for each layer to reconstruct the entire picture. Unlike the single-color case, the average probability is computed for all target colors, through which the material combination that gives the largest probability is selected. (b) Average $p_{\Delta E_{00} \le 2}$ distribution for all 100,000 allowed material combinations. (c) Average CIEDE2000 computed over entire images before and after particle swarm optimization finetuning.

through neural style transfer (*Gatys et al.*, 2016), which is a computer vision method that can transform images to possess an appearance similar to a style source image. As shown in Figure 4.8, we first apply the neural style transfer

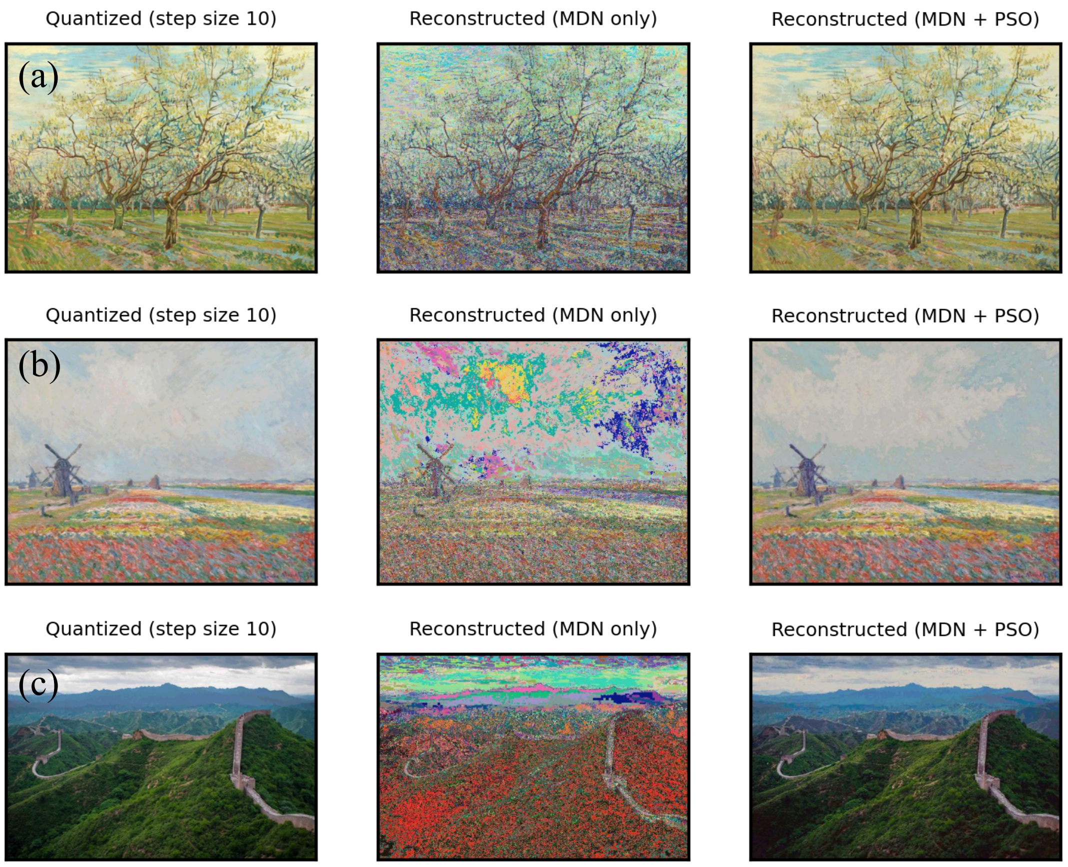


Figure 4.8: Picture reconstruction results. The first column is the target picture. The second column is the reconstruction based on NEUTRON. The last column is without the particle swarm finetuning. (a) *The White Orchard* by Vincent van Gogh. (b) *The Tulip Field* by Vincent van Gogh. (c) A photo of The Great Wall of China taken by photographer Severin Stalder. (a - b) are reproduced with the permission from the Van Gogh Museum, Amsterdam (Vincent van Gogh Fundation). (c) is reproduced with the permission from Wikimedia Commons.

method to transfer real photos into stylized pictures, then apply NEUTRON to obtain designs for reconstructing the transferred pictures.

Results in Figure 4.7 and Figure 4.8 demonstrates that NEUTRON allows accurate picture reconstructions with almost unnoticeable errors. Additionally, the entire design process can be accomplished within a few hours for high-

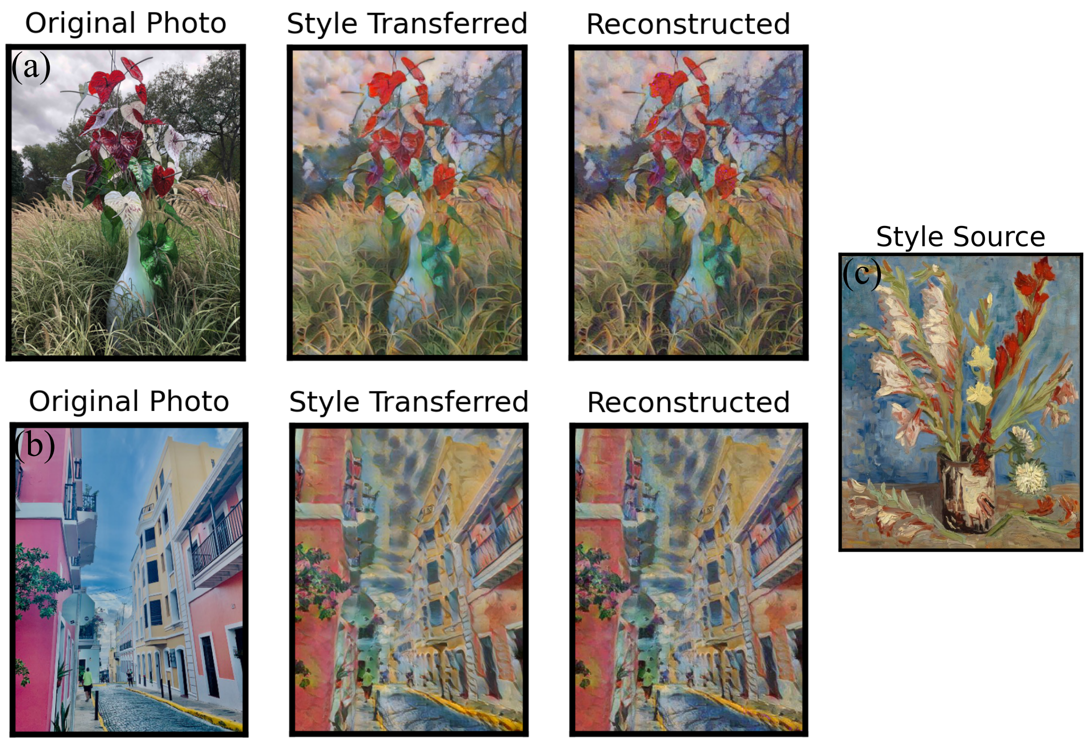


Figure 4.9: Picture reconstruction results based on neural style transferred photos. (a) Flower. (b) Street. Both original photos in (a - b) were taken by the authors. (c) The source style painting *Vase with Gladioli and Chinese Asters* for generating the style transferred photos is reproduced with the permission from the Van Gogh Museum, Amsterdam (Vincent van Gogh Foundation).

resolution pictures (Table 4.3), which makes NEUTRON a highly practical algorithm for real applications. Note that, although PSO also achieves good performance in terms of ΔE_{00}, the PSO results reported in Table 4.3 is based on the material combination predicted by the classification model of the trained M^{3}DN and does not take the amount of time required for screening the materials when such a model is not available. Additionally, when initializing the PSO with the initial designs predicted by the M^{3}DN, the design accuracy ΔE_{00} can be further improved upon using randomly-initialized PSO.

Though the designs across an entire picture are based on the same material combination, varying the top four layers' thickness is still challenging for fabrication. When the user can accept a higher CIEDE2000, it is possible to reconstruct the picture by varying only a single layer. In Figure. 4.10, we show results when fixing the top antireflective to be 150 nm, both absorber layers to be 9 nm, and only optimizing the bottom dielectric layer. These thickness values are the majority predicted values by the mixture density network for the entire picture. Compared to varying all four layers thickness, the reconstruction quality of varying a single layer is significantly worse. However, if fabrication constraints are the top priority for the reconstruction task, such an approach of varying a single layer could be adopted to allow an easy fabrication process.

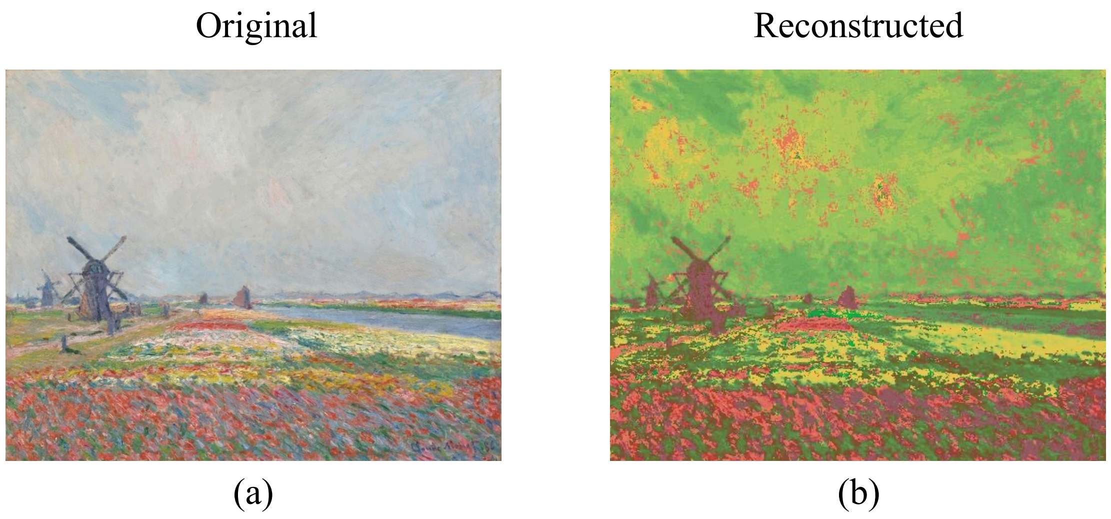


Figure 4.10: Picture reconstruction results when tuning only the dielectric layer, i.e., the fourth layer counting from the top. The average CIEDE2000 is 13.36.

4.4.3 Understanding Algorithmic Design Choices

We conducted a series of experiments to understand the effects of three important algorithmic design choices for the proposed algorithm NEUTRON,

Table 4.3: The best material combinations and CIEDE2000 with MDN only, PSO only, and NEUTRON for reconstructing pictures in Figure 4.7 and Figure 4.8. The results reported for the last two pictures in italic font are for the style transferred photos.

Picture	Material Combination					MDN	PSO	NEUTRON	Time (s)
White Orchard	Al_2O_3	Ge	Al	Al_2O_3	Zn	18.55	5.16	**3.76**	9,917
Tulip Field	Ta_2O_5	Au	Ti	Al_2O_3	Au	19.45	4.34	**3.48**	9,246
Great Wall	HfO_2	Au	Ti	SiO_2	Au	21.46	6.45	**5.69**	6,835
Flower	Ta_2O_5	Au	Ti	Al_2O_3	Au	19.37	5.00	**3.93**	11,249
Street	ZnO	Ti	Au	MgF_2	Au	21.60	4.45	**3.28**	7,349

including the number of mixture components, multitask learning, and dataset augmentation with mixture materials.

4.4.3.1 Effect of Number of Mixture Components

The optimal number of mixture components m is dependent on the one-to-many mapping between the input $(\boldsymbol{x}, \boldsymbol{c})$ and the potential designs $\{\boldsymbol{y}\}$. A large m is required when many designs could lead to the similar color target. Here, we vary the number of mixtures from 200 to 51,200. As shown in Figure 4.11, $m = 51,200$ leads to the best performance while reducing m cause the validation CIEDE2000 to increase significantly.

4.4.3.2 Effect of Multitask Learning

We vary the weighting parameter α to investigate if positive transfer exists between the material classification task and the design prediction task. Results in Figure 4.11 (c) and (d) show that $\alpha = 5$ leads to the best CIEDE2000 measured on the validation set while the classification AUC is insensitive to nonzero α values. This indicates that positive knowledge transfer exists

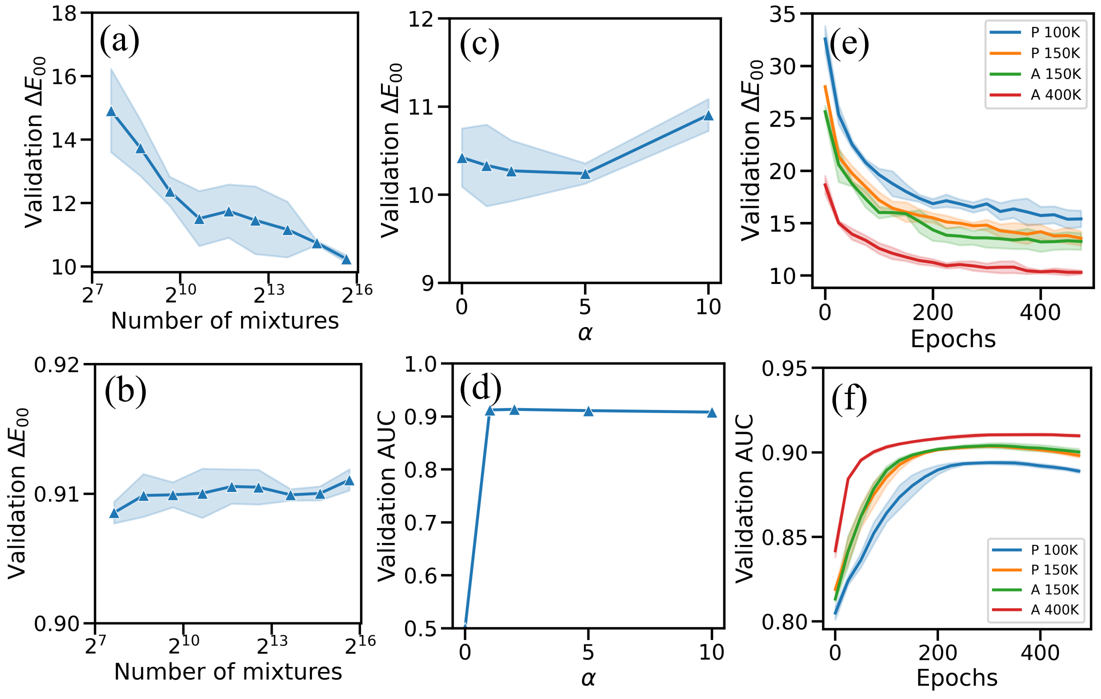


Figure 4.11: Effects of important algorithmic design choices. All results are based on three runs initialized with different random seeds. The shaded area corresponds to one standard deviation. (a) Varying the number of mixtures. The network with 51,200 mixture components achieves the best validation performance among the nine experiments. (b) Effect of weighting parameter for the classification loss. (c) Effect of dataset augmentation with mixed materials. By adding 50,000 datapoints from mixed materials (A 150K), the trained network achieves a more significant improvement of CIEDE2000 over the dataset with 100,000 datapoints simulated from primitive materials (P 100K), compared to adding 50,000 additional datapoints simulated from only primitive materials (P 150K). Further increasing the dataset size to 400,000 datapoints with 50% of the data simulated from the mixed materials leads to further performance improvement of both validation CIEDE2000 and the validation AUC.

between the material classification task and the thickness prediction task. Thus, incorporating the material classification task does not only allows users to select the best material combinations for downstream tasks but also improves the thickness inverse design accuracy due to the benefit of multitask learning.

4.4.3.3 Effect of Dataset Augmentation

Introducing datapoints simulated from a mixture of materials increases the variation of the input refractive data $\boldsymbol{x}$, which could allow the trained M^{3}DN to generalize better. To test this hypothesis, we first train M^{3}DN on a dataset with 100,000 datapoints simulated entirely based on primitive materials. Then, we add additional 50,000 datapoints either simulated from primitive materials or mixture materials and train two networks separately. By comparing the validation ΔE_{00}, we observe that adding 50,000 additional datapoints from mixture materials leads to greater performance improvement than adding 50,000 datapoints simulated from primitive materials.

4.5 Discussion

In both the chrome color design and the picture reconstruction tasks, NEUTRON achieves exceptional design accuracy efficiently through combining a material classification network, a mixture density network for thickness predictions, and the final PSO finetuning. Unlike previous methods that either assume fixed materials (*Gao et al.*, 2019; *Dai et al.*, 2021) or search the materials and thickness designs simultaneously (*Wang et al.*, 2021), our method split the materials and thickness design into two stages to enable an approach that can search for the best materials and design the thickness accordingly in an efficient manner. The reason why splitting the material and thickness design is helpful can be understood by the fact that the entire design space is much larger than the material design space or the thickness design space alone. In our five-layer optical thin film design task, the material design space size is 9×10^4, and the thickness design space size is 7.3×10^6, which lead to a huge full design space with a size of $(9 \times 7.3) \times 10^{10} = 6.3 \times 10^{11}$ when considering the material design space and the thickness design space simultaneously. With the two-step process, we first narrow down the promising material combinations with the material classification model and only optimize the thickness designs for the selected small set of materials. Thus, we only need to consider a design space that is one or multiple times of the thickness design space, which is a $\sim 10,000$X complexity reduction compared to searching both materials and thickness in one step.

We combine M^{3}DN and PSO by initializing the particle positions with designs sampled from the MDN. This process is easy to implement and highly effective in obtaining optimal designs due to the probabilistic nature of the mixture density network. On both design tasks, we show that PSO significantly improved the initial solutions by the M^{3}DN. This result is not surprising because PSO involves iterative updates of the design parameters based on the feedback from optical simulations. Additionally, the exceptional design accuracy after finetuning the initial M^{3}DN designs with PSO indicates that probabilistic machine learning models that can output diverse predictions are highly compatible with PSO, which requires an initial population of designs to begin with.

Our experiment on the number of mixtures suggests that a strong degree of one-to-many mapping exists (Figure 4.9 (a), (b)) for the problem of inverse designing the structural color with a five-layer optical stack, thus a large number of mixture components of 51,200 is required to obtain an accurate inverse prediction of the designs. Due to the limit of GPU memory, we cannot further increase the number of mixture components, but we expect the design accuracy to improve as the number is increased further.

Another interesting finding on multitask learning suggests that future optical inverse design models should leverage the task relationships by learning the main and auxiliary tasks simultaneously. As shown in the experiment for varying the classification loss weight α, we observe optimal inverse design performance at $\alpha = 5$. In optical design problems, many tasks are related, such as the electric field distribution prediction, far-field intensity prediction, spectrum prediction, and color prediction, etc. Learning multiple tasks simultaneously could allow researchers to obtain high inverse design accuracy with fewer datapoints.

Moreover, our results show that data augmentation that increases the refractive index data variation by mixing the primitive materials is an efficient strategy to improve the inverse design accuracy. We believe such data augmentation should be incorporated for training material-aware inverse design models when possible.

NEUTRON combines a classification model, a probabilistic regression model, and particle swarm optimization for finetuning the designs predicted by the M^{3}DN. Compared to using ML or the optimization approach alone, NEUTRON brings the best of both worlds to allow efficient and accurate inverse designs for large-scale design tasks.

4.6 Conclusion

We develop a material-aware inverse design algorithm that combines a multitask mixture density network and particle swarm optimization to select the optimal material combination and optimize structural parameters. On two practical tasks for producing reflective structural color with a five-layer optical stack, NEUTRON demonstrates the proposed algorithm's exceptional design accuracy and efficiency. Though we develop NEUTRON for structural color design based on multilayer optical thin films, it can be adopted for wide range of other optical design tasks that require optimizing the material selections and structural parameters.

CHAPTER 5

Model-based Offline Meta-Reinforcement Learning via Planning

5.1 Introduction

Most manufacturing problems can be treated as a sequential decision making problem. Modern industrial manufacturing systems are often powered by classical control algorithms, such as PID controller or Linear-Quadratic Regulator (LQR), etc. However, classical control algorithms often do not involve learning, i.e., their performance does not improve as more experience is gained. Also, the shared dynamics among similar manufacturing systems could not be shared to boost the control performance. On the other hand, reinforcement learning holds a great premise in learning from diverse experiences from similar systems to achieve a better control performance than classical control algorithms.

However, most existing reinforcement learning algorithms require learning from online feedback from the environment, which is infeasible for many high-stake manufacturing applications. In addition, these algorithms often

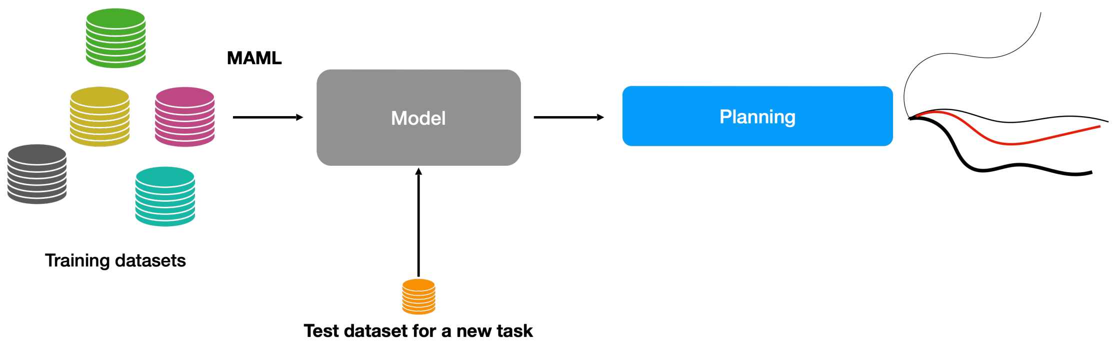


Figure 5.1: Meta model-based offline planning. The proposed method combines meta-learning for environment models and planning based on the environment models. Planning trajectories are averaged based on the expected returns. And the first action in the average trajectory is executed in the real environment.

do not leverage the fact that many control problems share similar underlying dynamics, allowing the learning to be more sample efficient. Thus, it is important to enable reinforcement learning to learn from offline datasets collected from related tasks. This way, we can enable a data-driven decision-making engine similar with a much broader impact than what the current reinforcement learning algorithm can achieve. To this end, we propose a model-based offline reinforcement learning algorithm that can effectively combine gradient-based meta-learning and offline planning. Our contributions are:

1. we proposed the first model-based offline meta-RL algorithm by combining gradient-based meta-learning and offline planning.

2. our method achieves superior performance than baseline approaches on standard MuJoCo environments.

3. we provided detailed empirical analysis of the proposed algorithm and discussed the benefits of model-based approaches for offline meta-RL.

5.2 Background

Our work concerns the *offline meta-reinforcement learning* problem. By combining *gradient-based meta-learning* and *model-based offline planning*, we propose a novel offline reinforcement learning approach that can efficiently adapt to new environments. We review the related work in this section.

5.2.1 Gradient-based Meta-Learning

Gradient-based meta-learning is based on the idea of learning neural network initialization parameters that are amenable to fast adaptation (*Finn et al.*, 2017; *Nichol and Schulman*, 2018). It has been widely used for both supervised learning and reinforcement learning (*Vanschoren*, 2018). Here, the neural network initialization parameters are treated as the meta-parameters and learned through meta-gradient, which is computed based on two disjoint sets of data used for simulating the few-shot learning setting during training. One of the most widely gradient-based meta-learning methods is model-agnostic meta-learning (MAML), which is also used in our work. Specifically, the meta-gradient of MAML is computed through the following second-order derivative:

$$\theta' \leftarrow \theta - \beta \overbrace{\nabla_\theta \sum_{\tau_i \sim p(\tau)} \mathcal{L}^{qry}_{\tau_i}\left(f_{\underbrace{\theta - \alpha \nabla_\theta \mathcal{L}^{spt}_{\tau_i}(f_\theta)}_{\text{adaptation}}} \right)}^{\text{meta-gradient}},$$

where $\mathcal{L}^{spt}_{\tau_i}, \mathcal{L}^{qry}_{\tau_i}$ are loss function values evaluated on two disjoint sets of data called the support set and the query set. Note that support set is used for the inner loop update called adaptation and the query set is used for the outer loop meta-gradient evaluation.

5.2.2 Offline Meta-Reinforcement Learning

Meta-reinforcement learning studies the application of meta-learning in reinforcement learning for improved sample efficiency (*Rakelly et al.*, 2019). Most previous meta-reinforcement learning methods are based on the online setting, where the agent can interact with the environment during training. However, training an RL agent in this online manner is prohibited in many real-world applications, such as healthcare and autonomous driving, due to the expensive or even dangerous online data collection. Thus, offline meta-Reinforcement learning methods (*Mitchell et al.*, 2021; *Li et al.*, 2021; *Pong et al.*, 2021; *Dorfman and Tamar*, 2020) have been proposed recently to tackle this problem by enabling the meta-RL agents to learn from static offline datasets containing useful demonstrations. Such demonstrations are often logged observational data generated by human experts and exist for many important applications.

In (*Dorfman and Tamar*, 2020), the authors proposed the BoREL, which is based on the recent Bayesian RL method VariBAD (*Zintgraf et al.*, 2020). BoREL can efficiently explore a new environment after being meta-trained on offline meta-datasets. In (*Li et al.*, 2021), a contextual meta-RL approach called FOCAL is applied to the offline setting and demonstrates good performance. MACAW (*Mitchell et al.*, 2021) is a gradient-based offline meta-RL method based on MAML. However, instead of learning a model, MACAW directly learns a policy through advantage-weighted regression (*Peng et al.*, 2019). BoREL, FOCAL, and MACAW all require reward information when adapting to a new task, which may not always be available. In (*Pong et al.*, 2021), the authors proposed SMAC that can adapt itself through state observations only via a self-supervised approach.

Table 5.1: Comparison between offline meta-RL methods.

Method	**Online Reward Labels**	**Model-based**	**Gradient-based Meta-Learning**
BoREL (*Dorfman and Tamar*, 2020)	✓	✗	✗
FOCAL (*Li et al.*, 2021)	✓	✗	✗
MACAW (*Mitchell et al.*, 2021)	✓	✗	✓
SMAC (*Pong et al.*, 2021)	✗	✗	✗
M^{2}BOP (ours)	✓	✓	✓

5.2.3 Model-based Offline Planning

Model-based planning is a popular approach for solving reinforcement learning problems without explicitly learning a policy (*Chua et al.*, 2018). Recently, *Argenson and Dulac-Arnold* (2021) extended the idea to the offline setting by training a deep ensemble as the environment model. Specifically, ensembles for the behavior policy, dynamics, and reward models are trained to evaluate rollouts without interacting with the real environment. Then model-predictive path-integral (*Chua et al.*, 2018) is applied to optimize the action trajectory based on the model only. The action noise parameter σ when rolling out the behavior policy and the path-integral temperature κ are the two most important hyperparameters that require careful tuning.

5.3 Methods

Offline model-based meta-reinforcement learning can be implemented in two ways. In the first approach, meta-model and meta-policy can be trained simultaneously by meta-train a Dyna-style approach that uses the model to generate rollouts for training a policy. However, the meta-training process could be challenging due to coupling the meta-model and the meta-policy. Additionally, computing meta-gradient that flows back to the meta-model through the meta-policy becomes computationally intractable when long rollouts are used for training. Thus, we take the second approach by combining a meta-trained model with a planning procedure, which avoids the complexity of training a meta-policy with the meta-model. The same idea was previously studied in online meta-RL (*Clavera et al.*, 2019). Here, we extend the idea to the offline setting with a carefully designed offline-planning method (*Argenson and Dulac-Arnold*, 2021). Next, we provide the details on the neural network architecture of the environment model, the meta-training, and the meta-testing process that performs planning with the meta-trained environment model.

5.3.1 Environment Models

Similar to MBOP (*Argenson and Dulac-Arnold*, 2021), we train models to predict the transition state, immediate reward, behavior cloned action and a truncated value function for a fixed horizon. However, instead of training these models separately, we train a multitask network to boost the sample efficiency (*Caruana*, 1997). The activation from a shared backbone network is concatenated with either the previous action a_{t-1} or the current action a_t as the

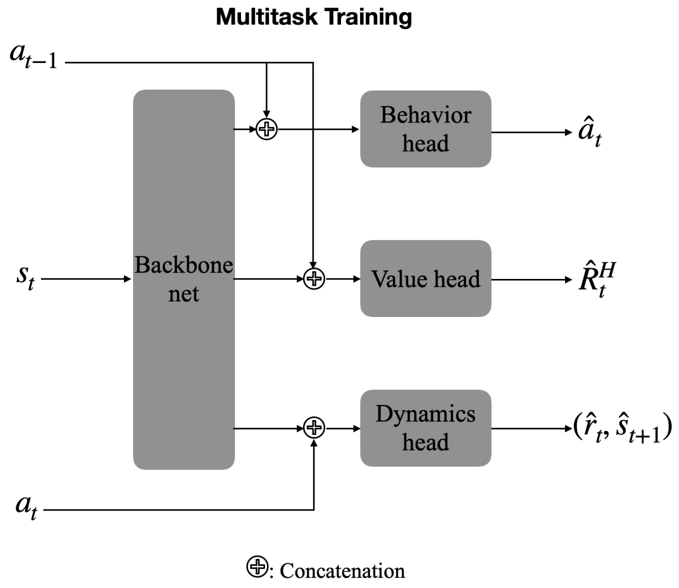


Figure 5.2: Mutlitask Neural Network for predicting environment dynamics and rewards.

input to three separate output heads for predicting the targets (Figure 5.2).

Instead of outputting a point estimate of each target, we predict the mean and standard deviation of an isotropic Gaussian distribution with each. Instead of training a single multitask network, we train a bootstrap ensemble with M randomly initialized base learners (*Lakshminarayanan et al.*, 2017) to alleviate the model-bias issue due to the distributional shift when the model is later deployed in a testing environment. The outputs of all base learners form a mixture of Gaussian distribution (Figure 5.3), whose mean and standard deviation is plugged in the negative log-likelihood loss for training the deep ensemble model. Specifically, the loss function is the total negative log-likelihood that counts for all four prediction targets i.e.,

$$\mathcal{L} = \mathbb{E}_{(a^-,s,a,r,s',R^H)\sim\mathcal{D}}\left[NLL_{s'} + NLL_r + NLL_a + NLL_{R^H}\right] \tag{5.1}$$

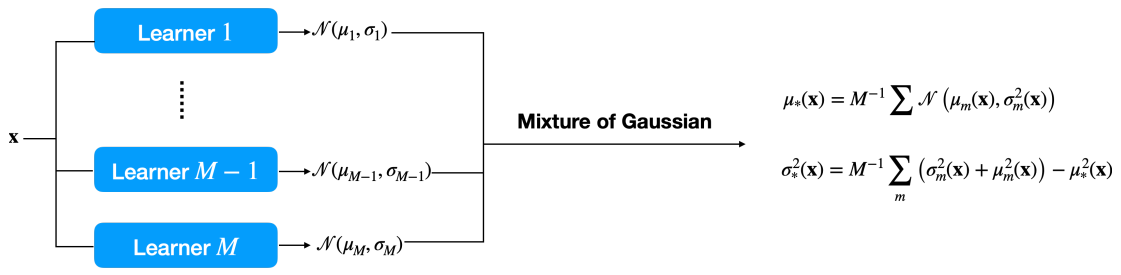

Learner 1
Learner $M-1$
Learner M
$\mathbf{x}$
$\mathcal{N}(\mu_1, \sigma_1)$
$\mathcal{N}(\mu_{M-1}, \sigma_{M-1})$
$\mathcal{N}(\mu_M, \sigma_M)$
Mixture of Gaussian
$\mu_*(\mathbf{x}) = M^{-1} \sum \mathcal{N}\left(\mu_m(\mathbf{x}), \sigma_m^2(\mathbf{x})\right)$
$\sigma_*^2(\mathbf{x}) = M^{-1} \sum_m \left(\sigma_m^2(\mathbf{x}) + \mu_m^2(\mathbf{x})\right) - \mu_*^2(\mathbf{x})$

Figure 5.3: Bootstrap ensemble for combining base learners.

where a^- is the previous action, s is the current state, a is the current action, r is the immediate reward after taking action a in state s, s' is the next state, and R^H is the truncated value function that equals to the sum of rewards for the next H steps. And

$$
\begin{aligned}
NLL_{s'} &= \frac{\log \sigma_{s'}^2(s,a)}{2} + \frac{(s' - \mu_{s'}(s,a))^2}{2\sigma_{s'}^2(s,a)} \\
NLL_r &= \frac{\log \sigma_r^2(s,a)}{2} + \frac{(r - \mu_r(s,a))^2}{2\sigma_r^2(s,a)} \\
NLL_a &= \frac{\log \sigma_a^2(s,a^-)}{2} + \frac{(a - \mu_a(s,a^-))^2}{2\sigma_a^2(s,a^-)} \\
NLL_{R^H} &= \frac{\log \sigma_{R^H}^2(s,a^-)}{2} + \frac{(R^H - \mu_{R^H}(s,a^-))^2}{2\sigma_{R^H}^2(s,a^-)}.
\end{aligned}
$$

5.3.2 Meta-Learning with MAML

Gradient-based meta-learning is applied to the environment model learning to allow fast adaptation to new tasks when only a limited amount of offline data is available. Specifically, we resort to MAML-style training (*Finn et al.*, 2017) to learn a set of neural network initialization parameters that can be updated with a few gradient steps to a local optimum using data collected from a new task. The pseudocode for meta-learning is described in Algorithm.

1. Note that, instead of randomly sampling from the offline buffer on the transition tuple level, we always sample on the trajectory level to mimic how the model will be deployed in practice. Unlike online gradient-based meta-RL (*Rakelly et al.*, 2019), the environmental model is trained entirely with offline data without any online data collection.

Algorithm 1 M^{2}BOP (train time)

```
1: Require: Offline buffers $\{\mathcal{D}_i\}_{i=1}^N$
2: Require: inner loop learning rates $\alpha$, outer loop learning rates, $\beta$, training iterations $n$
3: Randomly initialize meta-parameters $\theta$
4: for $n$ steps do
5:     for $i = 1, \ldots K$ do
6:         Sample offline buffer $\mathcal{D} \sim \{\mathcal{D}_i\}_{i=1}^N$
7:         Sample disjoint episodes $\mathcal{D}^{tr} = \{\tau_1^{tr}, \ldots, \tau_K^{tr}\}$, $\mathcal{D}^{te} = \{\tau_1^{te}, \ldots, \tau_K^{te}\} \sim \mathcal{D}$
8:         $\theta_i' \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}\left(\theta, \mathcal{D}^{tr}\right)$
9:     end for
10:    $\theta \leftarrow \theta - \beta \nabla_\theta \frac{1}{K} \sum_{i=1}^K \mathcal{L}(\theta_i', D_i^{te})$
11: end for
12: Return $\theta$ as $\theta^*$
```

5.3.3 Meta Model-Based Offline Planning

After the environment model has been properly trained with MAML, it can achieve accurate dynamics and reward function with only a few trajectories collected on a new environment. After adaptation steps on the new environment, the environment model can be plugged into a planning controller for deployment (Algorithm. 2). The controller used for offline planning can be variants of cross-entropy method (*Rubinstein*, 1999) or model predictive path integral (*Williams et al.*, 2016). Specifically, we apply the planning procedure developed in MBOP (*Argenson and Dulac-Arnold*, 2021) during the testing phase.

Algorithm 2 M^{2}BOP (test time)

```
1: Require: $\theta^*, \mathcal{D}$
2: Require: controller
3: Initialize $a_0 \leftarrow \mathbf{0}$, $R \leftarrow 0, \theta_0 \leftarrow \theta^*$
4: for $k = 0 \dots K-1$ do
5:     $\theta_{k+1} \leftarrow \theta_k - \alpha \nabla_\theta \mathcal{L}(\theta_k, \mathcal{D})$
6: end for
7: for $t = 1 \dots H$ do
8:     Observe $s_t, r_t$ by executing $a_{t-1}$ in the real environment
9:     $T^t$ = controller($s_t, \theta'$)
10:    $a_t = T_0^t$
11:    $R \leftarrow R + r_t$
12: end for
13: Return $R$ as the episodic return
```

5.4 Experiments

We tested the proposed method on the standard meta-RL benchmarks based on the robot locomotion environments in the MuJoCo simulator. First we collected the offline data by training the state-of-the-art online RL algorithm soft actor-critic (SAC, (*Haarnoja et al.*, 2018a)) on MuJoCo tasks with randomly initialized different reward functions (cheetah-dir, cheetah-vel, ant-dir), or different dynamics (walker-rand-params).

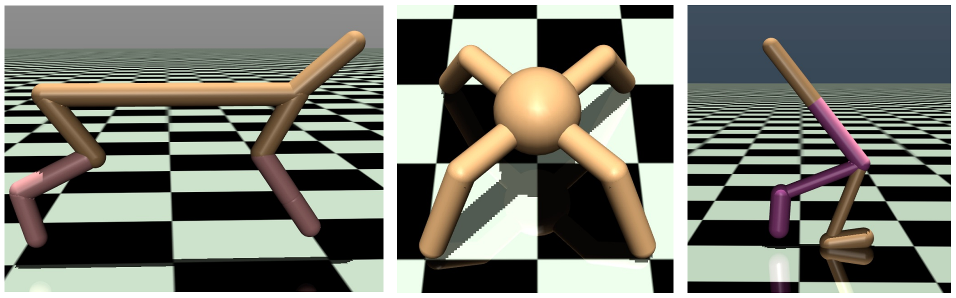

(a) (b) (c)

Figure 5.4: MuJoCo environments for testing the proposed offline meta-RL algorithm. (a) Half-cheetah. (b) Ant. (c) Walker.

- Cheetah-dir: This meta-testing benchmark is based on the half-cheetah robot (Figure. 5.4a). Two different reward functions that assign large rewards when the robot moves towards the left or the right direction create two tasks. Both tasks are used for both meta-training and meta-testing. This experiment does not test the meta-generalization performance but instead serves as a simple sanity check.

- Cheetah-vel: Same to cheetah-dir, cheetah-vel is also based on the half-cheetah robot. However, we randomly generated 40 reward functions that assign the highest reward when the robot moves at a specific velocity toward the right direction, each corresponding to a unique task. Thirty-five tasks are used for meta-training, and five tasks are held-out sets for meta-testing.

- Ant-dir: Ant-dir is another benchmark where we vary the reward function among tasks. It is based on the ant robot (Figure 5.4b) that has more complicated dynamics than the half-cheetah due to a larger number of joints. We generated fifty random tasks with the different target moving directions in the entire 360-degree horizontal plane, among which five are held out for meta-testing.

- Walker-rand-params: Unlike the above three benchmarks, walker-rand-params are based on randomly initialized joint parameters of the walker robot (Figure. 5.4c) that determines the environment dynamics. Same as ant-dir, we create forty-five tasks for meta-training and five tasks for meta-testing.

For each unique task in all four tasks, we train a SAC policy for one million steps. The policy checkpointed at 0th, 40,000th, and 960,000th steps are used for performing rollouts in the corresponding task to collect offline buffers with random, medium, expert quality data, respectively. All offline buffers contain one-million steps of transition tuples. Additionally, the entire transition data encountered by the SAC policy during training is used as another type of dataset that contains a mixture of random, medium, and expert-level data. The SAC policy training results for all four environments are shown in (Figure 5.5).

Next, we evaluate the performance of M^{2}BOP on all 16 offline datasets (four environments, each with four different quality levels). Through the experiment, we aim to answer three questions: 1) can M^{2}BOP outperform previous offline meta-RL methods, especially when low-quality demonstrations exist in the offline buffer? 2) how do different quality levels affect the performance of M^{2}BOP? 3) how sensitive is M^{2}BOP to hyperparameters, including the total number of adaptation steps K, and the planning parameters including action noise σ and temperature κ?

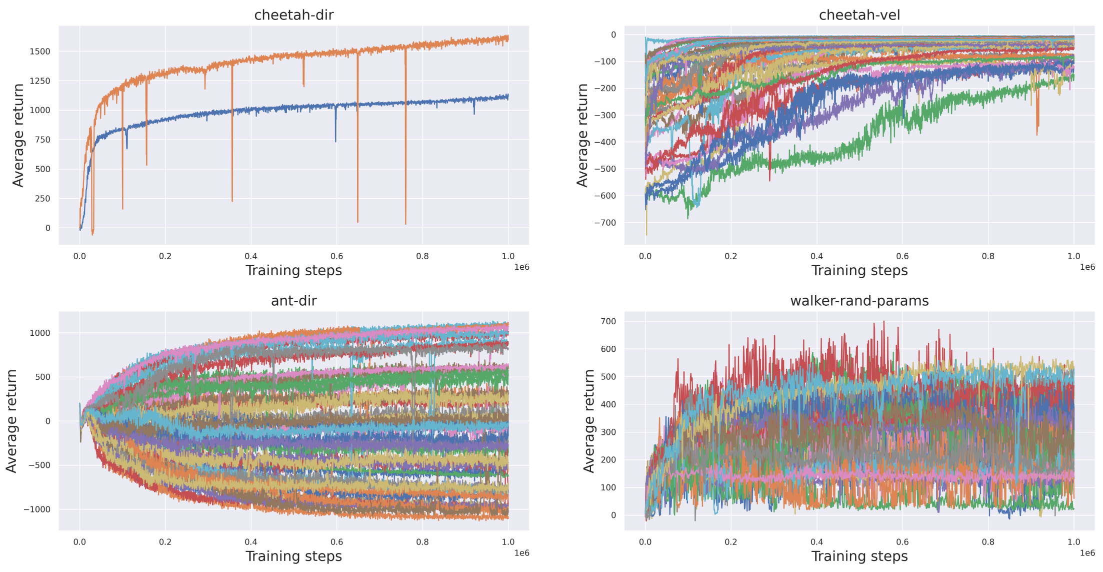


Figure 5.5: Training progress of SAC on all four environments. We rollout the saved policy at step 0, 40,000, and 960,000 collecting random, medium, and mixed datasets, respectively. For mixed datset, we use the entire experience replay buffer accumulated throughout the entire training process.

Table 5.2: Hyperparameters for grid search of M^{2}BOP.

Hyperparameter	Values
Adaptation steps K	[1, 3, 5, 10, 20, 40, 100]
Action noise σ	[0.1, 0.2, 0.3, 0.4]
Temperature κ	[1, 2, 3, 4]

5.4.1 Impact of Dataset Quality

As the first experiment, we evaluate M^{2}BOP on all four MuJoCo environments, each with four different dataset quality levels. For each testing task, we did a grid search of three critical hyperparameters (Table. 5.2) of M^{2}BOP and report the best performance. We randomly sampled five trajectories from the offline dataset for each task to adapt the environment model. We used an inner learning rate of 0.01 and an outer learning rate of 0.00005 for all our experiments during the meta-training and meta-testing process.

As a comparison, we also include the performance of meta-behavior cloning (Meta-BC), which is the ablated version of M^{2}BOP without the planning module, and two state-of-the-art offline meta-RL algorithms including FOCAL (*Li et al.*, 2021) and MACAW (*Mitchell et al.*, 2021) in Table 5.3.

Table 5.3: Comparison between our proposed algorithm and baseline algorithms on all four environments, each with four dataset types. Return values of MACAW and FOCAL are the test performance obtained through processing the original papers' learning curve figures. Some cells are filled with ***** because the results were not reported.

Dataset Type	**Environment**	**Meta-BC**	**M^{2}BOP**	**MACAW** (*Mitchell et al.*, 2021)	**FOCAL** (*Li et al.*, 2021)
Random	cheetah-dir	-9.62	**−3.35**	*****	*****
	cheetah-vel	-278.24	**−273.60**	*****	*****
	ant-dir	185.26	**194.06**	*****	*****
	walker-params	75.60	**122.64**	*****	*****
Medium	cheetah-dir	826.44	**881.75**	*****	*****
	cheetah-vel	-241.70	**-230.71**	*****	*****
	ant-dir	439.40	**572.65**	*****	*****
	walker-params	377.18	**380.99**	*****	*****
Expert	cheetah-dir	1212.36	**1276.06**	*****	*****
	cheetah-vel	-65.31	**-59.19**	*****	-122.57
	ant-dir	**646.92**	626.52	*****	*****
	walker-params	395.50	**424.49**	*****	*****
Mixed	cheetah-dir	1155.72	**1275.60**	919.22	1139.71
	cheetah-vel	-85.81	**-62.46**	-114.18	*****
	ant-dir	525.47	**620.64**	354.93	537.41
	walker-params	339.02	358.96	**381.80**	379.01

Notably, M^{2}BOP outperforms meta-BC on 15/16 tasks, which proves the effectiveness of the offline planning module. Additionally, M^{2}BOP outperforms MACAW and FOCAL on 3/4 mixed type datasets. This result confirms our hypothesis that M^{2}BOP is especially useful when low-quality demonstrations exist in the offline dataset.

5.4.2 Effect of Adaptation Steps

Next, we performed a fine-grained analysis of the meta-testing performance by varying the number of adaptation steps K (Algorithm 2). We note that a small number of adaption steps (≤ 3) suffice for the half-cheetah environment with the simplest dynamics, while it's beneficial to have more adaptation steps in environments with more complicated dynamics (Figure 5.6). The same trend holds for both meta-BC and M^{2}BOP.

5.4.3 Sensitivity to Planning Hyperparameters

Another important factor that affects the performance of M^{2}BOP is the hyperparameters used in the planning phase. Specifically, we investigate the sensitivity of M^{2}BOP to action noise σ and temperature κ (Figure 5.7). Notably, for datasets that allow accurate environment model training and behavior policy (i.e., expert, mixed), the highest return was obtained with a low action noise and a relatively large temperature value. This is reasonable because the learned behavior policies provide high-quality action samples, making a high-level of exploration unnecessary. Additionally, since the trained models accurately estimate the returns, a large temperature value can be selected to focus on action trajectories that are predicted to give high returns.

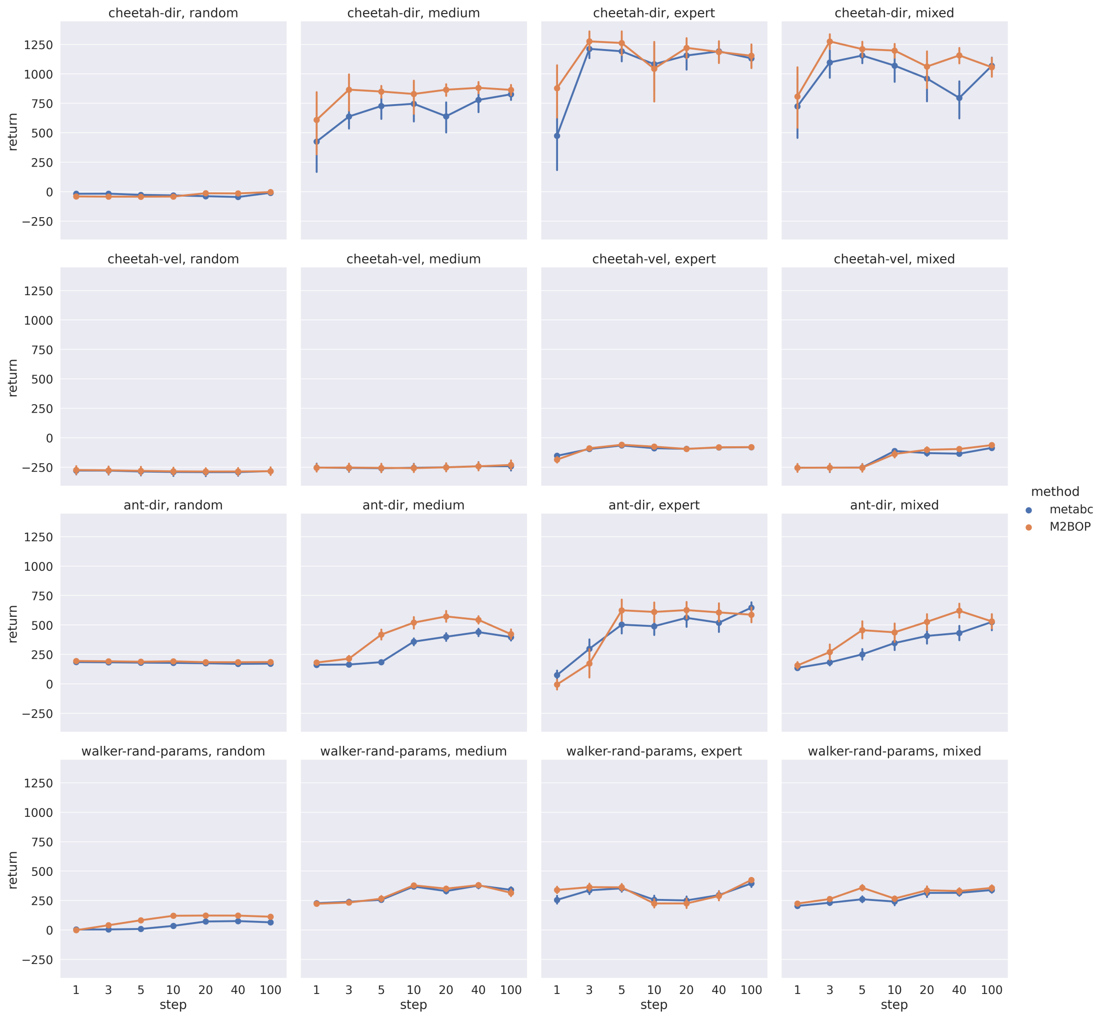


Figure 5.6: Effects of the number of adaptation steps on the testing returns.

5.5 Conclusion

We proposed the first model-based offline meta-RL algorithm M^{2}BOP. On four different MuJoCo environments across random, medium, expert, and mixed datatype, M^{2}BOP demonstrates superior performance compared to baseline methods. The proposed method is especially effective in learning performant policies when the dataset contains both low-quality and high-quality demonstrations. We believe the superior performance of M^{2}BOP is due to the

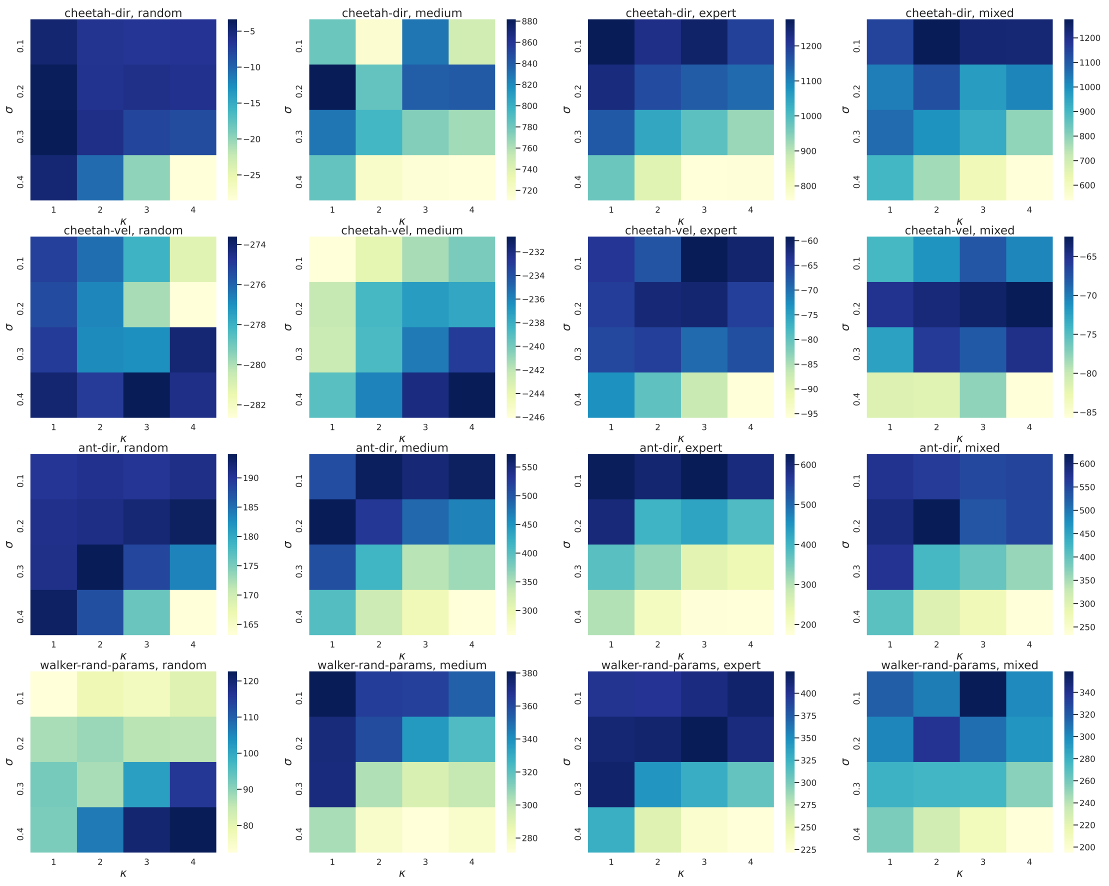


Figure 5.7: Sensitivity of M^{2}BOP to the planning parameters σ and κ. The highest returns on random datasets are obtained with large σ values. As the data contain more optimal demonstrations, a smaller action perturbation of 0.1 is required for achieving the optimal performance.

benefit of learning an explicit model, which could provide better extrapolation that counter-strikes the distributional shift issue in offline reinforcement learning. In the future, improved variants of model-based offline planning can be incorporated to further improve the offline learning performance of M^{2}BOP.

CHAPTER 6

Conclusions and Future Work

Learning to optimize is an efficient approach to solving complicated optimization problems in physical science and engineering, including inverse design and manufacturing process parameter optimization. We identify 1) large design space, 2) non-uniqueness of global optima and 3) costly data collection as the most common challenges when developing learning-to-optimize methods. To tackle these challenges and make the learning-to-optimize paradigm widely adopted in physical science and engineering fields, we developed reinforcement learning, hybrid machine learning, and optimization approaches to efficiently navigate the promising design space. Specifically, we applied the developed methods for multiple important applications, including perfect absorber, incandescent light bulb filter, and chrome coating replacement designs. The discovered optimal designs can outperform previous designs obtained by experts. Additionally, we proposed a generic offline meta-reinforcement learning algorithm that can be used for manufacturing optimization applications when observational datasets on related manufacturing tasks are available. We demonstrated that the proposed methods could perform better than several baseline approaches on standard robotic locomotion simulators. However, challenges remain when applying learning-to-optimize to solve real-world problems. We list the important future research directions as follows.

6.1 Offline Model-Based Optimization

Similar to the offline meta-reinforcement learning problem we investigated in Chapter 5, offline model-based optimization aims to learn to optimize a target property with completely offline data (*Trabucco et al.*, 2021a,b; *Fu and Levine*, 2021; *Yu et al.*, 2021a; *Kumar and Levine*, 2020). Like offline reinforcement learning, offline data for model-based optimization problems allows one to train a machine learning model with an adequate understanding of the underlying process. Due to the inability to collect additional data, model-based optimization shares the distributional shift challenge with offline reinforcement learning. The major difference between offline model-based optimization and offline reinforcement learning is the procedure used to obtain the best solution from the offline-trained model. In offline reinforcement learning, approximate dynamic programming or planning is used for obtaining the best action sequence. The offline model-based optimization process is arguably much easier, which can be a simple gradient-based procedure. As long as the distributional shift issue is under control, the optimized solution can often outperform the best solution contained in the dataset (*Trabucco et al.*, 2021a). Offline model-based optimization is especially suitable for inverse design and manufacturing process parameter optimization problems involving expensive or dangerous merit function evaluation processes.

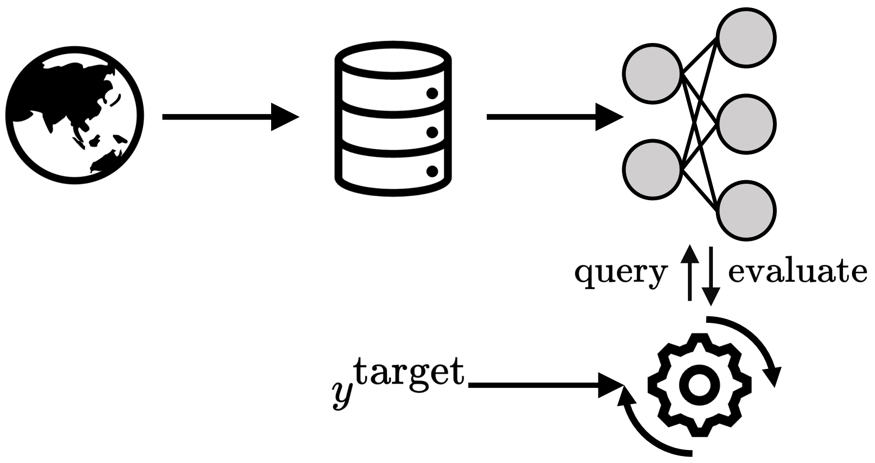


Figure 6.1: Offline model-based optimization. An observational dataset is collected from the real environment and later used for training a surrogate model for the true underlying process. A query process, often a gradient-based approach, is applied to query and find the best input to the trained surrogate model. The optimized input to the model is returned as the optimal solution.

In Chapter 5, we proposed an offline meta-reinforcement learning algorithm that can be used for closed-loop control in the manufacturing process. Another family of manufacturing process optimization problems does not require adaptive control. Instead, the optimal process parameters are chosen at the beginning of the manufacturing process and fixed throughout the entire process. Additive manufacturing methods for printed electronics including inkjet printing (*Minnette and Sebastian*, 2021) and aerosol printing (*Zhang et al.*, 2019a) are notable examples. Offline model-based optimization has not been applied to the additive manufacturing process optimization problem yet, but we believe that adopting the offline model-based optimization paradigm can greatly speed up the additive manufacturing process optimization process.

As a concrete application example, we are working with our collaborators from Boise State University on both inkjet printing and aerosol jet printing to optimize both manufacturing processes using offline data. Two performance metrics for the printing process are *overspray* and *line smoothness*. As the first step, we used design of experiment (DOE) to sample process parameters that uniformly cover the entire design space and collected real printing results (Figure 6.2). The next step is to apply offline model-based optimization to optimize the printing process parameters and validate the optimization parameters via experiments.

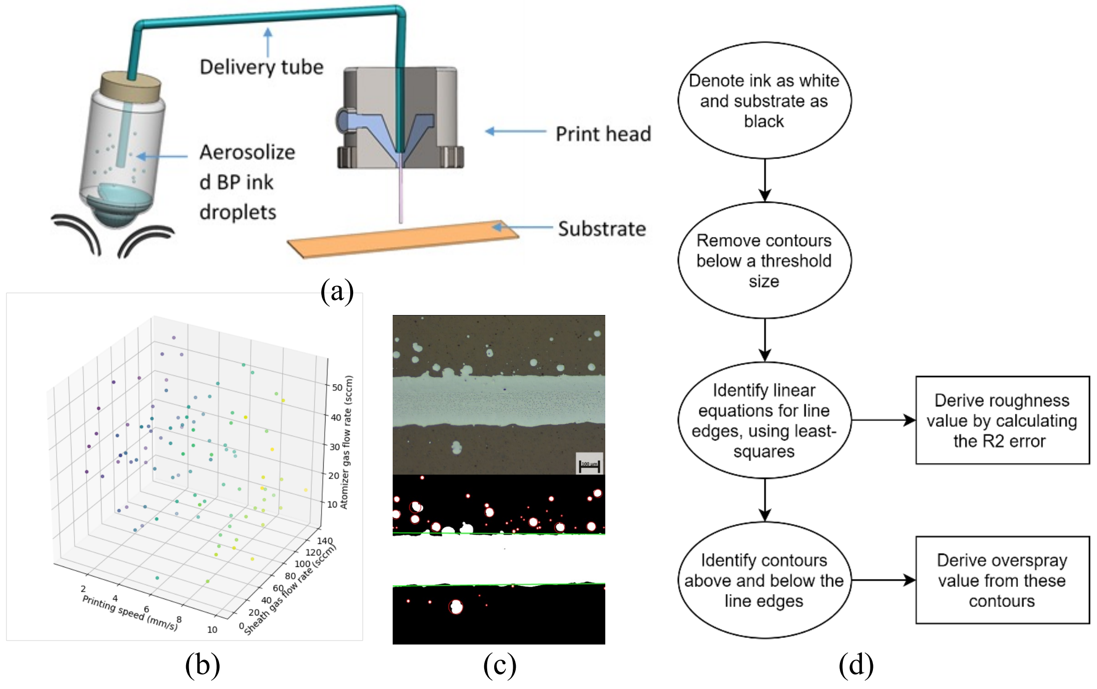

Delivery tube
Aerosolized BP ink droplets
Print head
Substrate
(a)
Printing speed (mm/s)
Sheath gas flow rate (sccm)
Atomizer gas flow rate (sccm)
(b)
(c)
Denote ink as white and substrate as black
Remove contours below a threshold size
Identify linear equations for line edges, using least-squares
Derive roughness value by calculating the R2 error
Identify contours above and below the line edges
Derive overspray value from these contours
(d)

Figure 6.2: Aerosol jet printing data collection process. (a) The apparatus for aerosol jet printing. (b) Process parameters sampled with Latin Hypercube Sampling. (c) A printed conducting line structure. Original microscope image (top) and processed image for measuring printing quality (bottom). (d) Image processing process for obtaining the overspray and line smoothness labels from the collected images.

6.2 Integrating Physics with Learning-to-Optimize

Meta-learning provides an efficient approach to reduce the sample complexity of learning-to-optimize methods. However, it still has a strong limitation that data from related tasks must be provided in order to meta learn, which may not always be feasible. Other methods that could reduce the sample complexity must be considered in this case. Fortunately, most real-world engineering problems are grounded on well-studied physics models in the form of partial differential equations (PDEs), which can provide the machine learning model with valuable information and speed up the learning-to-optimize process.

Recently, physics-informed neural networks have been proposed to infuse knowledge of PDEs into neural network training (*Raissi et al.*, 2019) and solve inverse design problems (*Chen et al.*, 2020; *Lu et al.*, 2021). However, the inverse design problems solvable with existing physics-informed neural networks are mostly simple toy examples and can be accurately described by a set of PDEs. It remains an open research question on how physics-informed neural networks could be applied to problems where PDEs can only describe partial behavior or provide an approximation to the underlying physical process.

6.3 Extracting Knowledge from Learned Optimizers

Finally, obtaining new knowledge from solving a physical science or engineering problem is as, if not more, important than solving the problem itself. Fortunately, various methods have been developed to interpret trained neural networks. Such methods include 1) knowledge extraction approaches such as symbolic regression (*Udrescu and Tegmark*, 2020) and 2) indirect model explanation approaches such as counterfactual explanation (*Karimi et al.*, 2020), LIME (*Ribeiro et al.*, 2016), and SHAP (*Lundberg and Lee*, 2017; *Yeung et al.*, 2021a), to name an important few. To allow the learning-to-optimize paradigm to truly revolutionize the scientific discovery and engineering innovation process, methods that allow knowledge extraction should be incorporated in learning-to-optimize methods. Additionally, researchers should carefully study the interaction between users and such interpretation methods to allow easy adoption by domain experts (*Wang et al.*, 2018a; *Rudin*, 2019).

Concluding Remarks We believe that integrating physics with sample-efficient and explainable machine learning approaches for learning-to-optimize can lead to a *data-driven scientific discovery engine*. The research is still in its infancy. Such an AI system could revolutionize how science and engineering are done and lead to new innovations at a pace we could never imagine before. However, we believe humans will always be at the center of innovation. To maximize the utility of scientific AI systems, researchers should build to *assist* rather than to *replace* domain experts for a synergistic combination of human intelligence and machine intelligence that can advance the frontiers of science and technology.

BIBLIOGRAPHY

Achiam, J. (2018), *Spinning Up in Deep Reinforcement Learning*.

Agarwal, R., D. Schuurmans, and M. Norouzi (2020), An optimistic perspective on offline reinforcement learning, in *Proceedings of the 37th International Conference on Machine Learning*, *Proceedings of Machine Learning Research*, vol. 119, edited by H. D. III and A. Singh, pp. 104–114, PMLR.

Agrawal, M., and P. Peumans (2008), Broadband optical absorption enhancement through coherent light trapping in thin-film photovoltaic cells, *Optics express*, *16*(8), 5385–5396.

Angermueller, C., D. Belanger, A. Gane, Z. Mariet, D. Dohan, K. Murphy, L. Colwell, and D. Sculley (2020a), Population-based black-box optimization for biological sequence design, in *Proceedings of the 37th International Conference on Machine Learning*, *Proceedings of Machine Learning Research*, vol. 119, edited by H. D. III and A. Singh, pp. 324–334, PMLR.

Angermueller, C., D. Dohan, D. Belanger, R. Deshpande, K. Murphy, and L. Colwell (2020b), Model-based reinforcement learning for biological sequence design, in *International Conference on Learning Representations*.

Ardizzone, L., J. Kruse, C. Rother, and U. Köthe (2019), Analyzing inverse problems with invertible neural networks, in *International Conference on Learning Representations*.

Argenson, A., and G. Dulac-Arnold (2021), Model-based offline planning, in *International Conference on Learning Representations*.

Audet, C., and W. Hare (2017), Derivative-free and blackbox optimization.

Bagloee, S. A., M. Asadi, M. Sarvi, and M. Patriksson (2018), A hybrid machine-learning and optimization method to solve bi-level problems, *Expert Systems with Applications*, *95*, 142–152.

Baxter, J., A. C. Lesina, J.-M. Guay, A. Weck, P. Berini, and L. Ramunno (2019), Plasmonic colours predicted by deep learning, *Scientific reports*, *9*(1), 1–9.

Bello, I., H. Pham, Q. V. Le, M. Norouzi, and S. Bengio (2016), Neural combinatorial optimization with reinforcement learning, *arXiv preprint arXiv:1611.09940*.

Best, J. (2017), *Colour design: theories and applications*, Woodhead Publishing.

Bishop, C. M. (1994), Mixture density networks, in *Neural Computing Research Group Report*, vol. 4, pp. 1–25, Aston University.

Boyd, S., S. P. Boyd, and L. Vandenberghe (2004), *Convex optimization*, Cambridge university press.

Byrnes, S. J. (2016), Multilayer optical calculations, *arXiv preprint arXiv:1603.02720*.

Campbell, S. D., D. Sell, R. P. Jenkins, E. B. Whiting, J. A. Fan, and D. H. Werner (2019), Review of numerical optimization techniques for meta-device design, *Optical Materials Express*, *9*(4), 1842–1863.

Caruana, R. (1997), Multitask learning, *Machine learning*, *28*(1), 41–75.

Chen, X., and Y. Tian (2019), Learning to perform local rewriting for combinatorial optimization, in *Advances in Neural Information Processing Systems*, pp. 6278–6289.

Chen, Y., L. Lu, G. E. Karniadakis, and L. D. Negro (2020), Physics-informed neural networks for inverse problems in nano-optics and metamaterials, *Opt. Express*, *28*(8), 11,618–11,633, doi:10.1364/OE.384875.

Chua, K., R. Calandra, R. McAllister, and S. Levine (2018), Deep reinforcement learning in a handful of trials using probabilistic dynamics models, in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, p. 4759–4770, Curran Associates Inc., Red Hook, NY, USA.

Chung, J., C. Gulcehre, K. Cho, and Y. Bengio (2014), Empirical evaluation of gated recurrent neural networks on sequence modeling, *arXiv preprint arXiv:1412.3555*.

Clavera, I., A. Nagabandi, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn (2019), Learning to adapt in dynamic, real-world environments through meta-reinforcement learning, in *International Conference on Learning Representations*.

Cook, D. F., C. T. Ragsdale, and R. Major (2000), Combining a neural network with a genetic algorithm for process parameter optimization, *Engineering applications of artificial intelligence*, *13*(4), 391–396.

Dai, H., E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song (2017), Learning combinatorial optimization algorithms over graphs, in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, p. 6351–6361, Curran Associates Inc., Red Hook, NY, USA.

Dai, P., Y. Wang, Y. Hu, C. de Groot, O. Muskens, H. Duan, and R. Huang (2021), Accurate inverse design of fabry–perot-cavity-based color filters far beyond srgb via a bidirectional artificial neural network, *Photonics Research*, *9*(5), B236–B246.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan (2002), A fast and elitist multiobjective genetic algorithm: Nsga-ii, *IEEE transactions on evolutionary computation*, *6*(2), 182–197.

Deisenroth, M., and C. E. Rasmussen (2011), Pilco: A model-based and data-efficient approach to policy search, in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, Citeseer.

Dimopoulos, C., and A. M. Zalzala (2000), Recent developments in evolutionary computation for manufacturing optimization: problems, solutions, and comparisons, *IEEE transactions on evolutionary computation*, *4*(2), 93–113.

Dorfman, R., and A. Tamar (2020), Offline meta learning of exploration, *arXiv: Learning*.

Duan, Y., J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel (2016), Rl\$^{2}\$: Fast reinforcement learning via slow reinforcement learning, *ArXiv*, *abs/1611.02779*.

EPA (2005), Reducing air pollution from: Electroplating operations, https://www.epa.gov/sites/default/files/2017-06/documents/electroplating_comm_info.pdf, accessed: 2021-10-10.

Finn, C., P. Abbeel, and S. Levine (2017), Model-agnostic meta-learning for fast adaptation of deep networks, *arXiv preprint arXiv:1703.03400*.

Foster, D. H., K. Amano, S. M. Nascimento, and M. J. Foster (2006), Frequency of metamerism in natural scenes, *Josa a*, *23*(10), 2359–2372.

Fu, J., and S. Levine (2021), Offline model-based optimization via normalized maximum likelihood estimation, *ArXiv*, *abs/2102.07970*.

Fujimoto, S., D. Meger, and D. Precup (2019), Off-policy deep reinforcement learning without exploration, in *International Conference on Machine Learning*, pp. 2052–2062, PMLR.

Gao, L., X. Li, D. Liu, L. Wang, and Z. Yu (2019), A bidirectional deep neural network for accurate silicon color design, *Advanced Materials*, *31*(51), 1905,467.

Gatys, L. A., A. S. Ecker, and M. Bethge (2016), Image style transfer using convolutional neural networks, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2414–2423.

George, J. (1992), Preparation of thin films.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014), Generative adversarial nets, in *Advances in neural information processing systems*, pp. 2672–2680.

Goodfellow, I., Y. Bengio, and A. Courville (2016), *Deep learning*, MIT press.

Gottesman, O., F. Johansson, M. Komorowski, A. Faisal, D. Sontag, F. Doshi-Velez, and L. A. Celi (2019), Guidelines for reinforcement learning in healthcare, *Nature medicine*, *25*(1), 16–18.

Graves, A. (2013), Generating sequences with recurrent neural networks, *arXiv preprint arXiv:1308.0850*.

Guo, K., Z. Yang, C.-H. Yu, and M. J. Buehler (2021), Artificial intelligence and machine learning in design of mechanical materials, *Materials Horizons*, *8*(4), 1153–1172.

Haarnoja, T., A. Zhou, P. Abbeel, and S. Levine (2018a), Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in *International conference on machine learning*, pp. 1861–1870, PMLR.

Haarnoja, T., et al. (2018b), Soft actor-critic algorithms and applications, *arXiv preprint arXiv:1812.05905*.

Ho, J., and S. Ermon (2016), Generative adversarial imitation learning, *Advances in neural information processing systems*, *29*, 4565–4573.

Hochreiter, S., and J. Schmidhuber (1997), Long short-term memory, *Neural computation*, *9*(8), 1735–1780.

Huang, Z., X. Liu, and J. Zang (2019), The inverse design of structural color using machine learning, *Nanoscale*, *11*(45), 21,748–21,758.

Hutter, F., H. H. Hoos, and K. Leyton-Brown (2011), Sequential model-based optimization for general algorithm configuration, in *International conference on learning and intelligent optimization*, pp. 507–523, Springer.

Hutter, F., H. Hoos, and K. Leyton-Brown (2013), An evaluation of sequential model-based optimization for expensive blackbox functions, in *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, pp. 1209–1216.

Ilic, O., P. Bermel, G. Chen, J. D. Joannopoulos, I. Celanovic, and M. Soljačić (2016), Tailoring high-temperature radiation and the resurrection of the incandescent source, *Nature nanotechnology*, *11*(4), 320.

Janner, M., J. Fu, M. Zhang, and S. Levine (2019), *When to Trust Your Model: Model-Based Policy Optimization*, Curran Associates Inc., Red Hook, NY, USA.

Jiang, J., D. Sell, S. Hoyer, J. Hickey, J. Yang, and J. A. Fan (2019), Free-form diffractive metagrating design based on generative adversarial networks, *ACS nano*, *13*(8), 8872–8878.

Jiang, J., M. Chen, and J. A. Fan (2020), Deep neural networks for the evaluation and design of photonic devices, *Nature Reviews Materials*, pp. 1–22.

Karimi, A.-H., G. Barthe, B. Balle, and I. Valera (2020), Model-agnostic counterfactual explanations for consequential decisions, *ArXiv*, *abs/1905.11190*.

Kennedy, J., and R. Eberhart (1995), Particle swarm optimization, in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, IEEE.

Khalil, E., H. Dai, Y. Zhang, B. Dilkina, and L. Song (2017), Learning combinatorial optimization algorithms over graphs, in *Advances in Neural Information Processing Systems*, pp. 6348–6358.

Kidambi, R., A. Rajeswaran, P. Netrapalli, and T. Joachims (2020), Morel: Model-based offline reinforcement learning, in *Advances in Neural Information Processing Systems*, vol. 33, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, pp. 21,810–21,823, Curran Associates, Inc.

Kim, S.-H., S. Y. Lee, S.-M. Yang, and G.-R. Yi (2011), Self-assembled colloidal structures for photonics, *NPG Asia Materials*, *3*(1), 25–33.

Kingma, D. P., and J. Ba (2014), Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., and M. Welling (2013), Auto-encoding variational bayes, *arXiv preprint arXiv:1312.6114*.

Köksal, G., I. Batmaz, and M. C. Testik (2011), A review of data mining applications for quality improvement in manufacturing industry, *Expert systems with Applications*, *38*(10), 13,448–13,467.

Kostrikov, I., J. Tompson, R. Fergus, and O. Nachum (2021), Offline reinforcement learning with fisher divergence critic regularization, in *ICML*.

Kumar, A., and S. Levine (2020), Model inversion networks for model-based optimization, *ArXiv*, *abs/1912.13464*.

Kumar, A., J. Fu, G. Tucker, and S. Levine (2019), Stabilizing off-policy q-learning via bootstrapping error reduction, in *NeurIPS*.

Kumar, A., A. Zhou, G. Tucker, and S. Levine (2020), Conservative q-learning for offline reinforcement learning, in *Advances in Neural Information Processing Systems*, vol. 33, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, pp. 1179–1191, Curran Associates, Inc.

Kuo, J.-T., Y.-Y. Wang, and W.-S. Lung (2006), A hybrid neural–genetic algorithm for reservoir water quality management, *Water research*, *40*(7), 1367–1376.

Lakshminarayanan, B., A. Pritzel, and C. Blundell (2017), Simple and scalable predictive uncertainty estimation using deep ensembles, in *NIPS*.

Lalau-Keraly, C. M., S. Bhargava, O. D. Miller, and E. Yablonovitch (2013), Adjoint shape optimization applied to electromagnetic design, *Optics express*, *21*(18), 21,693–21,701.

Levine, S., A. Kumar, G. Tucker, and J. Fu (2020), Offline reinforcement learning: Tutorial, review, and perspectives on open problems, *arXiv preprint arXiv:2005.01643*.

Li, C., and G. H. Lee (2019), Generating multiple hypotheses for 3d human pose estimation with mixture density network, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9887–9895.

Li, D., Y. Yang, Y.-Z. Song, and T. M. Hospedales (2018a), Learning to generalize: Meta-learning for domain generalization, *ArXiv*, *abs/1710.03463*.

Li, J., W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao (2016), Deep reinforcement learning for dialogue generation, in *Proceedings of EMNLP 2016*, pp. 1192–1202.

Li, K., and J. Malik (2017), Learning to optimize, in *International Conference on Learning Representations*.

Li, L., R. Yang, and D. Luo (2021), FOCAL: Efficient fully-offline meta-reinforcement learning via distance metric learning and behavior regularization, in *International Conference on Learning Representations*.

Li, W., and S. Fan (2018), Nanophotonic control of thermal radiation for energy applications, *Optics express*, *26*(12), 15,995–16,021.

Li, W., Y. Shi, K. Chen, L. Zhu, and S. Fan (2017), A comprehensive photonic approach for solar cell cooling, *ACS Photonics*, *4*(4), 774–782.

Li, W., Y. Shi, Z. Chen, and S. Fan (2018b), Photonic thermal management of coloured objects, *Nature communications*, *9*(1), 1–8.

Lieberman, H., et al. (1941), Chrome ulcerations of the nose and throat., *New England Journal of Medicine*, *225*(4), 132–3.

Lillicrap, T. P., J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra (2015), Continuous control with deep reinforcement learning, *arXiv preprint arXiv:1509.02971*.

Lindberg, E., and G. Hedenstierna (1983), Chrome plating: symptoms, findings in the upper airways, and effects on lung function, *Archives of Environmental Health: An International Journal*, *38*(6), 367–374.

Liu, C., S. A. Maier, and G. Li (2020), Genetic-algorithm-aided meta-atom multiplication for improved absorption and coloration in nanophotonics, *ACS Photonics*, *7*(7), 1716–1722.

Liu, D., Y. Tan, E. Khoram, and Z. Yu (2018a), Training deep neural networks for the inverse design of nanophotonic structures, *ACS Photonics*, *5*(4), 1365–1369.

Liu, T., B. VanSaders, J. T. Keating, S. C. Glotzer, and M. J. Solomon (2021), Effect of particles of irregular size on the microstructure and structural color of self-assembled colloidal crystals, *Langmuir*.

Liu, Z., D. Zhu, S. P. Rodrigues, K.-T. Lee, and W. Cai (2018b), Generative model for the inverse design of metasurfaces, *Nano letters*, *18*(10), 6570–6576.

Lu, H., X. Zhang, and S. Yang (2020), A learning-based iterative method for solving vehicle routing problems, in *International Conference on Learning Representations*.

Lu, L., R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson (2021), Physics-informed neural networks with hard constraints for inverse design, *SIAM Journal on Scientific Computing*, *43*(6), B1105–B1132, doi: 10.1137/21m1397908.

Lundberg, S. M., and S.-I. Lee (2017), A unified approach to interpreting model predictions, in *Proceedings of the 31st international conference on neural information processing systems*, pp. 4768–4777.

Ma, T., M. Tobah, H. Wang, and L. J. Guo (2022), Benchmarking deep learning-based models on nanophotonic inverse design problems, *Opto-Electronic Science*, *1*(1), 210,012–1.

Ma, W., F. Cheng, and Y. Liu (2018), Deep-learning-enabled on-demand design of chiral metamaterials, *ACS nano*, *12*(6), 6326–6334.

Ma, W., F. Cheng, Y. Xu, Q. Wen, and Y. Liu (2019), Probabilistic representation and inverse design of metamaterials based on a deep generative model with semi-supervised learning strategy, *Advanced Materials*, *31*(35), 1901,111.

Ma, W., Z. Liu, Z. A. Kudyshev, A. Boltasseva, W. Cai, and Y. Liu (2021), Deep learning for the design of photonic structures, *Nature Photonics*, *15*(2), 77–90.

Mahadevan, S., and G. Theocharous (1998), Optimizing production manufacturing using reinforcement learning., in *FLAIRS conference*, vol. 372, p. 377.

Mansecal, T., M. Mauderer, and M. Parsons (2021), Colour, https://github.com/colour-science/colour.

Minnette, F., and Z. Sebastian (2021), Deep learning for zero-defect inkjet-printing of electronics, in *2021 IEEE International Workshop on Metrology for Industry 4.0 IoT (MetroInd4.0 IoT)*, pp. 458–463, doi:10.1109/MetroInd4.0IoT51437.2021.9488493.

Mirhoseini, A., et al. (2017), Device placement optimization with reinforcement learning, in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2430–2439, JMLR. org.

Mirhoseini, A., et al. (2020), Chip placement with deep reinforcement learning, *arXiv preprint arXiv:2004.10746*.

Mitchell, E., R. Rafailov, X. B. Peng, S. Levine, and C. Finn (2021), Offline meta-reinforcement learning with advantage weighting, in *International Conference on Machine Learning*, pp. 7780–7791, PMLR.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller (2013), Playing atari with deep reinforcement learning, *arXiv preprint arXiv:1312.5602*.

Mnih, V., A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu (2016), Asynchronous methods for deep reinforcement learning, in *International conference on machine learning*, pp. 1928–1937, PMLR.

Molesky, S., Z. Lin, A. Y. Piggott, W. Jin, J. Vucković, and A. W. Rodriguez (2018), Inverse design in nanophotonics, *Nature Photonics*, *12*(11), 659–670.

Nian, R., J. Liu, and B. Huang (2020), A review on reinforcement learning: Introduction and applications in industrial process control, *Computers & Chemical Engineering*, *139*, 106,886.

Nichol, A., and J. Schulman (2018), Reptile: a scalable metalearning algorithm, *arXiv: Learning*.

O'Donoghue, B., I. Osband, R. Munos, and V. Mnih (2018), The uncertainty bellman equation and exploration, *ArXiv*, *abs/1709.05380*.

Osband, I., C. Blundell, A. Pritzel, and B. V. Roy (2016), Deep exploration via bootstrapped dqn, in *NIPS*.

OSHA (2013), Controlling hexavalent chromium exposures during electroplating, https://www.osha.gov/sites/default/files/publications/OSHA_FS-3648_Electroplating.pdf, accessed: 2021-10-10.

Paszke, A., et al. (2017), Automatic differentiation in pytorch.

Paszke, A., et al. (2019), Pytorch: An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems*, pp. 8024–8035.

Pearl, J. (2009), *Causality*, 2 ed., Cambridge University Press, Cambridge, UK, doi:10.1017/CBO9780511803161.

Pelikan, M., D. E. Goldberg, E. Cantú-Paz, et al. (1999), Boa: The bayesian optimization algorithm, in *Proceedings of the genetic and evolutionary computation conference GECCO-99*, vol. 1, pp. 525–532, Citeseer.

Peng, X. B., A. Kumar, G. Zhang, and S. Levine (2019), Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, *ArXiv*, *abs/1910.00177*.

Perez, E., F. Strub, H. de Vries, V. Dumoulin, and A. C. Courville (2018), Film: Visual reasoning with a general conditioning layer, in *AAAI*.

Pfrommer, J., C. Zimmerling, J. Liu, L. Kärger, F. Henning, and J. Beyerer (2018), Optimisation of manufacturing process parameters using deep neural networks as surrogate models, *Procedia CiRP*, *72*, 426–431.

Pong, V. H., A. Nair, L. Smith, C. Huang, and S. Levine (2021), Offline meta-reinforcement learning with online self-supervision, *arXiv preprint arXiv:2107.03974*.

Popova, M., O. Isayev, and A. Tropsha (2018), Deep reinforcement learning for de novo drug design, *Science advances*, *4*(7), eaap7885.

Pukelsheim, F. (2006), *Optimal design of experiments*, SIAM.

Rabady, R. I., and A. Ababneh (2014), Global optimal design of optical multilayer thin-film filters using particle swarm optimization, *Optik*, *125*(1), 548–553.

Raissi, M., P. Perdikaris, and G. Karniadakis (2019), Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *Journal of Computational Physics*, *378*, 686–707, doi:https://doi.org/10.1016/j.jcp.2018.10.045.

Rakelly, K., A. Zhou, D. Quillen, C. Finn, and S. Levine (2019), Efficient off-policy meta-reinforcement learning via probabilistic context variables, *ArXiv*, *abs/1903.08254*.

Raman, A. P., M. A. Anoma, L. Zhu, E. Rephaeli, and S. Fan (2014), Passive radiative cooling below ambient air temperature under direct sunlight, *Nature*, *515*(7528), 540–544.

Ren, M., E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel (2018), Meta-learning for semi-supervised few-shot classification, *ArXiv*, *abs/1803.00676*.

Ribeiro, M. T., S. Singh, and C. Guestrin (2016), ”why should i trust you?”: Explaining the predictions of any classifier, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Roberts, N. B., and M. Keshavarz Hedayati (2021), A deep learning approach to the forward prediction and inverse design of plasmonic metasurface structural color, *Applied Physics Letters*, *119*(6), 061,101.

Rubinstein, R. Y. (1999), The cross-entropy method for combinatorial and continuous optimization, *Methodology And Computing In Applied Probability*, *1*, 127–190.

Rudin, C. (2019), Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, *Nature Machine Intelligence*, *1*(5), 206–215.

Rusu, A. A., D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell (2018), Meta-learning with latent embedding optimization, *arXiv preprint arXiv:1807.05960*.

Sajedian, I., T. Badloe, and J. Rho (2019a), Optimisation of colour generation from dielectric nanostructures using reinforcement learning, *Optics express*, *27*(4), 5874–5883.

Sajedian, I., H. Lee, and J. Rho (2019b), Double-deep q-learning to increase the efficiency of metasurface holograms, *Scientific reports*, *9*(1), 1–8.

Salcedo-Sanz, S., C. Bousoño-Calzón, and A. R. Figueiras-Vidal (2003), A mixed neural-genetic algorithm for the broadcast scheduling problem, *IEEE Transactions on Wireless Communications*, *2*(2), 277–283.

Sallab, A. E., M. Abdou, E. Perot, and S. Yogamani (2017), Deep reinforcement learning framework for autonomous driving, *Electronic Imaging*, *2017*(19), 70–76.

Schmidhuber, J. (1994), On learning how to learn learning strategies.

Schubert, M. F., F. W. Mont, S. Chhajed, D. J. Poxson, J. K. Kim, and E. F. Schubert (2008), Design of multilayer antireflection coatings made from co-sputtered and low-refractive-index materials by genetic algorithm, *Optics express*, *16*(8), 5290–5298.

Schulman, J., P. Moritz, S. Levine, M. Jordan, and P. Abbeel (2015), High-dimensional continuous control using generalized advantage estimation, in *International Conference on Learning Representations*.

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

Sharma, G., W. Wu, and E. N. Dalal (2005), The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations, *Color Research & Application: Endorsed by Inter-Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color*

Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur, *30*(1), 21–30.

Sharpe, L. T., A. Stockman, W. Jagla, and H. Jägle (2005), A luminous efficiency function, v*(λ), for daylight adaptation, *Journal of Vision*, *5*(11), 3–3.

Shi, Y., W. Li, A. Raman, and S. Fan (2017), Optimization of multilayer optical films with a memetic algorithm and mixed integer programming, *ACS Photonics*, *5*(3), 684–691.

Shokooh-Saremi, M., and R. Magnusson (2007), Particle swarm optimization and its application to the design of diffraction grating filters, *Optics letters*, *32*(8), 894–896.

Silver, D., et al. (2017), Mastering the game of go without human knowledge, *Nature*, *550*(7676), 354–359.

Snoek, J., H. Larochelle, and R. P. Adams (2012), Practical bayesian optimization of machine learning algorithms, *Advances in neural information processing systems*, *25*.

Song, H., H. Wang, and V. Van (2022), An analytical method for evaluating the robustness of photonic integrated circuits, *Journal of Lightwave Technology*.

Spielberg, S., R. Gopaluni, and P. Loewen (2017), Deep reinforcement learning approaches for process control, in *2017 6th International Symposium on Advanced Control of Industrial Processes (AdCONIP)*, pp. 201–206, doi: 10.1109/ADCONIP.2017.7983780.

Sun, S., Z. Zhou, C. Zhang, Y. Gao, Z. Duan, S. Xiao, and Q. Song (2017), All-dielectric full-color printing with TiO_2 metasurfaces, *ACS nano*, *11*(5), 4445–4452.

Sutton, R. S., and A. G. Barto (2018), *Reinforcement learning: An introduction*, MIT press.

Tikhonravov, A. V., M. K. Trubetskov, and G. W. DeBell (1996), Application of the needle optimization technique to the design of optical coatings, *Applied optics*, *35*(28), 5493–5508.

Trabucco, B., X. Geng, A. Kumar, and S. Levine (2021a), Design-bench: Benchmarks for data-driven offline model-based optimization.

Trabucco, B., A. Kumar, X. Geng, and S. Levine (2021b), Conservative objective models for effective offline model-based optimization, in *ICML*.

Udrescu, S.-M., and M. Tegmark (2020), Ai feynman: A physics-inspired method for symbolic regression, *Science Advances*, *6*.

Unni, R., K. Yao, X. Han, M. Zhou, and Y. Zheng (2021), A mixture-density-based tandem optimization network for on-demand inverse design of thin-film high reflectors, *Nanophotonics*.

Van Hasselt, H., A. Guez, and D. Silver (2016), Deep reinforcement learning with double q-learning, in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30.

Vanschoren, J. (2018), Meta-learning: A survey, *ArXiv*, *abs/1810.03548*.

ViewSonic (2021), DeltaE <= 2 color accuracy, https://color.viewsonic.com/explore/content/DeltaE<=2Color-Accuracy_2.html, accessed: 2021-12-09.

Vinyals, O., M. Fortunato, and N. Jaitly (2015), Pointer networks, in *Advances in neural information processing systems*, pp. 2692–2700.

Vinyals, O., et al. (2019), Grandmaster level in starcraft ii using multi-agent reinforcement learning, *Nature*, *575*(7782), 350–354.

Vuorio, R., S.-H. Sun, H. Hu, and J. J. Lim (2019), Multimodal model-agnostic meta-learning via task-aware modulation, in *NeurIPS*.

Wang, H., and L. J. Guo (2022), Neutron: Neural particle swarm optimization for material-aware inverse design of structural color, *Available at SSRN 3992098*.

Wang, H., Z. Zheng, C. Ji, and L. J. Guo (2021), Automated multi-layer optical design via deep reinforcement learning, *Machine Learning: Science and Technology*, *2*(2), 025,013.

Wang, J., J. Oh, H. Wang, and J. Wiens (2018a), Learning credible models, in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2417–2426.

Wang, X., J. Bi, S. Yu, and J. Sun (2014), On multiplicative multitask feature learning., in *NIPS*, edited by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, pp. 2411–2419.

Wang, Y., et al. (2018b), Stepwise-nanocavity-assisted transmissive color filter array microprints, *Research*, *2018*.

Whitley, D. (1994), A genetic algorithm tutorial, *Statistics and computing*, *4*(2), 65–85.

Wiecha, P. R., A. Arbouet, C. Girard, A. Lecestre, G. Larrieu, and V. Paillard (2017), Evolutionary multi-objective optimization of colour pixels based on dielectric nanoantennas, *Nature nanotechnology*, *12*(2), 163–169.

Wiecha, P. R., A. Arbouet, C. Girard, and O. L. Muskens (2021), Deep learning in nano-photonics: inverse design and beyond, *Photonics Research*, *9*(5), B182–B200.

Williams, G., P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou (2016), Aggressive driving with model predictive path integral control, *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1433–1440.

Xu, X., C. Sun, Y. Li, J. Zhao, J. Han, and W. Huang (2021), An improved tandem neural network for the inverse design of nanophotonics devices, *Optics Communications*, *481*, 126,513.

Yang, C., L. Hong, W. Shen, Y. Zhang, X. Liu, and H. Zhen (2013), Design of reflective color filters with high angular tolerance by particle swarm optimization method, *Optics express*, *21*(8), 9315–9323.

Yang, C., C. Ji, W. Shen, K.-T. Lee, Y. Zhang, X. Liu, and L. J. Guo (2016), Compact multilayer film structures for ultrabroadband, omnidirectional, and efficient absorption, *Acs Photonics*, *3*(4), 590–596.

Yang, W., S. Xiao, Q. Song, Y. Liu, Y. Wu, S. Wang, J. Yu, J. Han, and D.-P. Tsai (2020), All-dielectric metasurface for high-performance structural color, *Nature communications*, *11*(1), 1–8.

Yang, Z., C. Ji, D. Liu, and L. J. Guo (2019), Enhancing the purity of reflective structural colors with ultrathin bilayer media as effective ideal absorbers, *Advanced Optical Materials*, *7*(21), 1900,739.

Yao, K., R. Unni, and Y. Zheng (2019), Intelligent nanophotonics: merging photonics and artificial intelligence at the nanoscale, *Nanophotonics*, *8*(3), 339–366.

Yeung, C., D. Ho, B. Pham, K. T. Fountaine, and A. P. Raman (2021a), Enhancing adjoint optimization-based photonics inverse design with explainable machine learning, *arXiv preprint arXiv:2109.14886*.

Yeung, C., R. Tsai, B. Pham, B. King, Y. Kawagoe, D. Ho, J. Liang, M. W. Knight, and A. P. Raman (2021b), Global inverse design across multiple photonic structure classes using generative deep learning, *Advanced Optical Materials*, *9*(20), 2100,548.

You, C., C. T. Matyas, Y. Huang, J. Dowling, and G. Veronis (2020), Optimized multilayer structures with ultrabroadband near-perfect absorption, *IEEE Photonics Journal*.

Yu, S., S. Ahn, L. Song, and J. Shin (2021a), Roma: Robust model adaptation for offline model-based optimization, *ArXiv*, *abs/2110.14188*.

Yu, T., D. Quillen, Z. He, R. C. Julian, K. Hausman, C. Finn, and S. Levine (2019), Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, *ArXiv*, *abs/1910.10897*.

Yu, T., G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma (2020), Mopo: Model-based offline policy optimization, in *Advances in Neural Information Processing Systems*, vol. 33, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, pp. 14,129–14,142, Curran Associates, Inc.

Yu, T., A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn (2021b), Combo: Conservative offline model-based policy optimization, *arXiv preprint arXiv:2102.08363*.

Zhan, X., X. Zhu, and H. Xu (2021), Model-based offline planning with trajectory pruning, *ArXiv*, *abs/2105.07351*.

Zhang, H., M. Cisse, Y. N. Dauphin, and D. Lopez-Paz (2018), mixup: Beyond empirical risk minimization, in *International Conference on Learning Representations*.

Zhang, H., S. K. Moon, and T. H. Ngo (2019a), Hybrid machine learning method to determine the optimal operating process window in aerosol jet 3d printing., *ACS applied materials & interfaces*, *11 19*, 17,994–18,003.

Zhang, X. S., F. Tang, H. H. Dodge, J. Zhou, and F. Wang (2019b), Metapred: Meta-learning for clinical risk prediction with limited patient electronic health records, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Zhao, Q.-Y., D. Zhu, N. Calandri, A. E. Dane, A. N. McCaughan, F. Bellei, H.-Z. Wang, D. F. Santavicca, and K. K. Berggren (2017), Single-photon imager based on a superconducting nanowire delay line, *Nature Photonics*, *11*(4), 247–251.

Zheng, H., and A. Louri (2019), An energy-efficient network-on-chip design using reinforcement learning, in *Proceedings of the 56th Annual Design Automation Conference 2019*, pp. 1–6.

Zhou, J., X. Chen, and L. J. Guo (2016), Efficient thermal–light interconversions based on optical topological transition in the metal-dielectric multilayered metamaterials, *Advanced Materials*, *28*(15), 3017–3023.

Zhou, Z., S. Kearnes, L. Li, R. N. Zare, and P. Riley (2019), Optimization of molecules via deep reinforcement learning, *Scientific reports*, *9*(1), 1–10.

Zhu, C., R. H. Byrd, P. Lu, and J. Nocedal (1997), Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization, *ACM Transactions on Mathematical Software (TOMS)*, *23*(4), 550–560.

Zhu, W., T. Xu, H. Wang, C. Zhang, P. B. Deotare, A. Agrawal, and H. J. Lezec (2017), Surface plasmon polariton laser based on a metallic trench fabry-perot resonator, *Science advances*, *3*(10), e1700,909.

Zintgraf, L., K. Shiarli, V. Kurin, K. Hofmann, and S. Whiteson (2019), Fast context adaptation via meta-learning, in *International Conference on Machine Learning*, pp. 7693–7702, PMLR.

Zintgraf, L. M., K. Shiarlis, M. Igl, S. Schulze, Y. Gal, K. Hofmann, and S. Whiteson (2020), Varibad: A very good method for bayes-adaptive deep rl via meta-learning, *ArXiv*, *abs/1910.08348*.