Reinforcement Learning Enabled Autonomous Manufacturing Using Transfer Learning and Probabilistic Reward Modeling

Md Ferdous Alam, Max Shtein, Kira Barton, *Senior Member, IEEE*, and David Hoelzle, *Member, IEEE*

***Abstract*—Here we propose a reinforcement learning enabled physical autonomous manufacturing system (AMS) that is capable of learning the manufacturing process parameters to autonomously fabricate a complex-geometry artifact with desired performance characteristics. The poor sample efficiency of traditional RL algorithms challenges real-world manufacturing decision making due to a high variable cost from raw material, machine utilization, and labor costs. To make decision making sample efficient, we propose to leverage a first-principles based source task for training, transfer effective representations from trained knowledge, and then use these representations to interact with the physical system to learn a probabilistic model of the target reward function. We deploy this idea to a novel dataset obtained from a custom physical AMS machine that can autonomously manufacture phononic crystals, a complex geometry artifact with spectral response as performance characteristic. We demonstrate that our method uses as low as 25 artifacts to model the interesting part of the target reward function and find an artifact with high reward. This task typically requires manual design of phononic crystals and extensive empirical iterations on the order of hundreds.**

***Index Terms*—Reinforcement learning, transfer learning, Gaussian process, autonomous manufacturing.**

I. Introduction

AN AUTONOMOUS manufacturing system (AMS) is capable of fabricating an artifact that matches a predefined desired artifact property without any human intervention [1], [2]. AMS achieves this capability by learning the process parameters through intelligent decision making [3]. The idea of AMS goes beyond typical inverse design problems [4] as it takes into consideration not only a desired performance characteristic from the artifact but the challenges of manufacturability of such artifact as well. We describe AMS as a 'high variable cost environment' (HVC-env) where data collection has high sampling cost because of machine utilization, materials and labor cost. Thus only a small number of interactions are allowed with the physical system to learn the process parameters. To this end, we aim to build manufacturing system that learns to fabricate an artifact with some desired performance characteristics by sequentially manufacturing artifacts on the order of 10^1 or 10^2, observing the corresponding output performance characteristics and taking intelligent decisions accordingly. Due to the sequential nature of the problem we consider reinforcement learning (RL) [5] to facilitate decision making in such HVC-envs. Instead of leveraging black-box optimization methods, we propose to use RL for such task for two reasons that will be explained throughout this letter: (1) RL provides more flexible framework for designing and building a physical AMS (2) RL can facilitate intelligent decision making in complex, hard-to-model manufacturing systems. RL has solved impressive tasks such as mastering complex games [6] and floorplanning for computer chip manufacturing [7].

In spite of these successes RL is still not considered suitable for many real-world applications due to poor sample efficiency [8], [9]. Hence implementing RL in a HVC-env, particularly for AMS, is even more challenging due to the small interaction budget. To this end, we consider transfer learning approaches to deploy RL in physical AMS [10], [11]. Transfer learning in the context of RL often helps to accelerate the learning procedure in a target task by transferring knowledge from a source task. In this letter, we focus on transfer learning between two tasks where the source and target task may have different reward distributions. We argue that this problem setting is crucial for many real-world applications, such as AMS, because it is often easier to build an inexpensive source task based on physics-based models and then transfer knowledge to the expensive target task. While leveraging data from these first-principles based models is a

Manuscript received 21 March 2022; revised 30 May 2022; accepted 15 June 2022. Date of publication 4 July 2022; date of current version 10 August 2022. This work was supported in part by NSF Award CMMI under Grant 1727894. Recommended by Senior Editor C. Seatzu. *(Corresponding author: Md Ferdous Alam.)*

Md Ferdous Alam and David Hoelzle are with the Department of Mechanical and Aerospace Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: alam.92@osu.edu; hoelzle.1@osu.edu).

Max Shtein is with the Department of Materials Science and Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: mshtein@umich.edu).

Kira Barton is with the Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109 USA (e-mail: bartonkl@umich.edu).

Digital Object Identifier 10.1109/LCSYS.2022.3188014

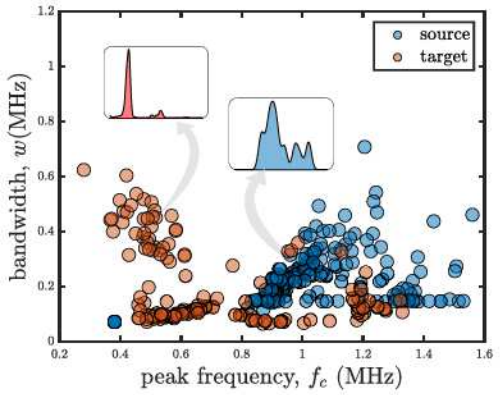


Fig. 1. Distribution of the output features: Each reward value of a manufactured phononic crystal artifact depends on the two output features f_c and w. The change in the distribution of f_c, w between the source and target task, in turn, changes the reward functions. Two example spectral responses for the same input feature values are also shown.

convenient idea, it comes with practical data-centric challenges that make transfer learning in the context of RL challenging for AMS. For example, in an AMS it is highly likely that black-box simulation engines and proprietary software are used in designing artifacts. These simulation engines often fail to accurately simulate complex manufacturing processes, utilize inaccurate or simplified models and ignore limitations of physical manufacturing systems. As a result, there exists discrepancies between source and target artifact properties which create discrepancies between the reward functions. This practical phenomena is visualized in Fig. 1. Here we show the distribution of output features obtained from Phononic crystals, a certain manufactured artifact that we consider in this letter as a proof-of-concept. It is easy to see that the distribution of these output features changes from the source to the target task which, in turn, changes the reward function. Traditionally engineers have to iterate on the design process numerous times to consider manufacturing issues which is inefficient.

In this letter we propose methods based on transfer learning in RL that can make this whole process autonomous and thus reduce a significant amount of manual process iteration. In the RL context, this problem setting is analogous to making data-efficient decisions in a target task by leveraging knowledge from a source task that has an inaccurate reward model. We draw motivation from previous studies [12] where reward function has been modeled as Gaussian process considering reward as a safety feature of the state. Our proposed method primarily consists of two steps. First, the RL agent is trained in a first-principles based source task. Second, a probabilistic model, specifically a Gaussian process model, of the target reward landscape in the physical manufacturing system is learned by utilizing the source knowledge. A key challenge in building a probabilistic target reward model is the sparsity of rewards in many tasks. Fortunately, we can exploit the fact that rewards are not sparse, but possibly nonlinear with respect to the features of states, for applications like AMS. Thus it is possible to build a probabilistic model of the reward function based on the observed data. Our contributions can be summarized as the following: 1) development of the conceptual framework for autonomous manufacturing in the context of sequential decision making, 2) development of a data efficient sequential decision making process for AMS using transfer RL, 3) dissemination of a novel experimental dataset for the proposed AMS case study and demonstration of the effectiveness of the proposed methods for this proof-of-concept physical AMS.

II. Background and Problem Formulation

A. Sequential Decision Making and Reinforcement Learning (RL)

The goal of sequential decision making is to maximize a performance objective by sequentially taking actions and observing corresponding feedback. We adopt the reinforcement learning (RL) framework with usual Markov decision process (MDP) [13] formalism to formulate sequential decision making in AMS. An MDP $\mathcal{M}$ is defined as a tuple $\mathcal{M} \doteq \langle \mathcal{X}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, where $\mathcal{X}$ is the state-space, $\mathcal{A}$ is the action-space, $\mathcal{P}$ is the transition probability that describes the dynamics of the system and $\mathcal{R} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ defines the reward function. At timestep t, the agent interacts with the environment by taking action $a_t \in \mathcal{A}$ at state $\mathbf{x}_t \in \mathcal{X}$ using a policy $\pi : \mathcal{X} \to \mathcal{A}$ and transitions to the next state $x_{t+1} \in \mathcal{X}$ according to the conditional probability $p(\mathbf{x}_{t+1}|\mathbf{x}_t, a_t)$ while receiving a scalar reward value $R_t = \mathcal{R}(\mathbf{x}_t, a_t)$ as the feedback. Ultimately, the goal is to find a policy π^* that maximizes the expected cumulative sum of discounted rewards, known as return G_t, from all states. One popular approach to address this problem is to use the concept of value function [5] where we can define action-value function of a policy π as $Q^\pi(\mathbf{x}, a) = \mathbb{E}^\pi[G_t|\mathbf{x}_t, a_t]$ to evaluate the policy. An improved policy which is at least as good as π, can be found by using dynamic programming (DP) approaches to choose the greedy policy with respect to $Q^\pi(\mathbf{x}_t, a_t)$. Under certain assumptions [14], this iterative policy evaluation and improvement leads to the optimal policy π^* that maximizes the expected return. Another popular approach to find π^* is to use policy gradient methods to directly optimize the parameterized policy π instead of relying on value-functions. In this letter we only consider value-function based methods although policy-based methods are feasible.

B. Transfer in Reinforcement Learning

Let us consider a source task described by the MDP $\mathcal{M}_\mathcal{S} = \langle \mathcal{X}, \mathcal{A}, \mathcal{R}_\mathcal{S}, \mathcal{P} \rangle$. We aim to utilize knowledge from $\mathcal{M}_\mathcal{S}$ to accelerate learning in a target task, described by the MDP $\mathcal{M}_\mathcal{T} = \langle \mathcal{X}, \mathcal{A}, \mathcal{R}_\mathcal{T}, \mathcal{P} \rangle$. Notice, only the reward function is different in these two MDPs. We also exploit the fact that rewards in AMS are only dependent on the states, not the state-action pairs, meaning $\mathcal{R}(\mathbf{x}, a) = \mathcal{R}(\mathbf{x})$, because each artifact in an AMS produces a reward that depends only on the features of the state. In this problem setting, there are at least two major challenges in RL transfer: identifying the types of knowledge to transfer from the source to the target task and evaluating the performance of the transfer [10]. Previous studies suggest various types of transfer including representation transfer, instance transfer and parameter transfer. For

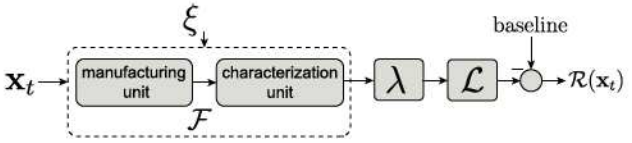


Fig. 2. Reward generation in an AMS.

data-efficient decision making in $\mathcal{M}_\mathcal{T}$, we choose a particular type of representation transfer known as temporal abstractions. In this type of transfer, temporally extended actions, also known as 'options', are identified in $\mathcal{M}_\mathcal{S}$ and then executed in $\mathcal{M}_\mathcal{T}$. As options are executed for multiple timesteps, unlike regular single timestep primitive actions, learning can be accelerated in $\mathcal{M}_\mathcal{T}$. It is important to note that options form a semi-Markov decisions process (SMDP) in the context of MDP [15]. Formally each option o in an SMDP is described by a three-element tuple $\mathcal{I} = \langle \mathcal{I}, \pi, \beta \rangle$ where o initiates from a state $\mathbf{x} \in \mathcal{I} \subseteq \mathcal{X}$, follows an intra-option policy π and then terminates at state $\mathbf{x}'$ with probability β.

C. Autonomous Manufacturing

To formalize autonomous manufacturing system in the context of sequential decision making and MDP, let's consider a generic manufacturing process $\mathcal{F}$,

$$\mathbf{g}_t = \mathcal{F}(\mathbf{x}_t) + \xi_t \text{ where } t \leq T_b \tag{1}$$

where process output $\mathbf{g}_t \in \mathbb{R}^{p\times 1}$ quantifies the characteristic properties of a manufactured artifact, process input $\mathbf{x}_t \in \mathbb{R}^{q\times 1}$ quantifies manufacturing process parameters and ξ_t is the unknown process noise, sample index t represents the time index as well. Additionally, the interaction budget T_b represents the total number of artifacts allowed for learning to manufacture the artifact with desired characteristics. This particular type of manufacturing system consists of both a manufacturing unit and an in-situ characterization unit for real-time characterization of the manufactured artifact. An artifact-property quantification $\mathbf{g}_t$ is obtained from each manufactured artifact as output from the characterization unit. Later, a post-processing function $\lambda(\cdot)$ takes $\mathbf{g}_t$ as input and produces a feature vector $\boldsymbol{\phi}_t$ as output, $\boldsymbol{\phi}_t = \lambda(\mathbf{g}_t)$. The goal is to learn the input $\mathbf{x}^*$ that produces desired output features of an artifact, $\boldsymbol{\phi}_d = \lambda(\mathbf{g}_d)$, while not manufacturing artifacts more than the interaction budget T_b. The learning process may start from a random initial position $\mathbf{x}_0 \sim \mathcal{U}(\mathbf{x}_{\min}, \mathbf{x}_{\max})$ or a specific initial position of interest. At each timestep, a loss is constructed based on the similarity between $\boldsymbol{\phi}_t$ and $\boldsymbol{\phi}_d$ meaning $\ell_t = \mathcal{L}(\boldsymbol{\phi}_t, \boldsymbol{\phi}_d)$, where $\mathcal{L}(\cdot)$ is 'part-property quantification similarity' (PQS), a similarity metric between the desired and current output features of the artifact. To convert the loss into reward values we use an arbitrary constant value as the baseline such that

$$R_t = \text{baseline} - \ell_t = \text{baseline} - \mathcal{L}(\boldsymbol{\phi}_t, \boldsymbol{\phi}_d). \tag{2}$$

Note that PQS can be a ℓ_2 norm or any distance metric or it can be a custom designed similarity metric. The full reward generation process is shown in Figure 2.

III. Sample Efficient Learning of a Probabilistic Model of the Target Reward Function

Our idea is to use transferred representations obtained from $\mathcal{M}_\mathcal{S}$ to learn a probabilistic model of the target reward function $\mathcal{R}_\mathcal{T}(\mathbf{x})$ in $\mathcal{M}_\mathcal{T}$. A pseudocode explaining these ideas can be found in the Algorithm 1. We use Gaussian process (GP) [16] for the probabilistic modeling of $\mathcal{R}_\mathcal{T}(\mathbf{x})$ meaning $\mathcal{R}_\mathcal{T}(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$. A GP describes a probability distribution over functions which can be fully specified by a mean function $\mu(\mathbf{x})$ and a kernel or covariance function $k(\mathbf{x}, \mathbf{x}')$. Initially, we train the agent in $\mathcal{M}_\mathcal{S}$ using any RL algorithm and then extract the optimal policy $\pi^*_\mathcal{S}$ from the trained action-values $Q^*_\mathcal{S}$ in case of value based RL methods or directly optimized $\pi^*_\mathcal{S}$ in case of policy gradient methods. Next we use $\pi^*_\mathcal{S}$ to guide the agent in $\mathcal{M}_\mathcal{T}$ to collect a dataset from the target task, and build a GP posterior conditioned on this dataset. Ideally we would like to interact with $\mathcal{M}_\mathcal{T}$ using all the actions suggested by $\pi^*_\mathcal{S}$. But as we want to keep the number of interactions small, we use the transferred temporal abstractions, otherwise known as options, from $\pi^*_\mathcal{S}$ to interact with $\mathcal{M}_\mathcal{T}$.

At each time step we roll-out a trajectory $\tau_H = \{\mathbf{x}_t, a_t, \mathbf{x}_{t+1}, \ldots, a_{t+H-1}, \mathbf{x}_{t+H}\}$ upto horizon H using $\pi^*_\mathcal{S}$. Next we choose a subgoal state, $\mathbf{x}_{\text{subgoal}}$ from this trajectory. One way to choose $\mathbf{x}_{\text{subgoal}}$ is to choose the most visited state within the trajectory τ_H. Alternatively $\mathbf{x}_{\text{subgoal}}$ can be a carefully chosen state from τ_H. For example, the subgoal state can be the state corresponding to the maximum change in rewards within τ_H. In the simplest setting we can execute $\pi^*_\mathcal{S}$ for a fixed horizon of H in an open loop fashion where $\mathbf{x}_{\text{subgoal}}$ would be the last state of τ_H. In this way an option o_s, is created that starts from $\mathbf{x}$, follows $\pi^*_\mathcal{S}$ and terminates at $\mathbf{x}' \equiv \mathbf{x}_{\text{subgoal}}$. Thus each option can be described as $o_s \equiv \langle \mathbf{x}, \pi^*_\mathcal{S}, \mathbf{x}' \rangle$. We only observe a target reward $\mathcal{R}_\mathcal{T}(\mathbf{x}')$ once the option terminates. The option we have just created is directly obtained from $\pi^*_\mathcal{S}$ and does not account for the exploration needed to overcome the bias induced by the source reward. Hence instead of creating a single option from $\pi^*_\mathcal{S}$ we create a set of options $\{o_i\}_1^m$ where each option is an ϵ-greedy variant of o_s. This idea is somewhat close to 'probabilistic policy reuse' [17] where guidance is provided from past learned similar policies. This means that each option o_i follows o_s with high probability most of the time and occasionally chooses random action to explore according to small probability ϵ, $\{o_i\} = \{o_i : o_i \leftarrow \epsilon\text{-greedy}(o_s)\}, i = 1, \ldots, m$. In this way if we choose n subgoals we will create $n \times m$ options which will give us $n \times m$ target reward values $R(\cdot) \sim \mathcal{R}_\mathcal{T}(\cdot)$. Now we can create a dataset $\mathcal{D} = \{\mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}}\}$ of observed target rewards values, where $\mathbf{X}_{\text{train}} = [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n\times m)}]^T$, $\mathbf{y}_{\text{train}} = [R^{(1)}, \ldots, R^{(n\times m)}]^T$. Using $\mathbf{y}_{\text{train}}$ it is possible to obtain $\hat{\mathcal{R}}_\mathcal{T}$, for the entire state-space $\mathbf{X}_{\text{test}} = \mathcal{X}$, which is the GP posterior of $\mathcal{R}_\mathcal{T}$ conditioned on the observed reward,

$$p(\hat{\mathcal{R}}_\mathcal{T}(\mathbf{x}) | \mathbf{X}_{\text{test}}, \mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}}) = \mathcal{N}(\hat{\mathcal{R}}_\mathcal{T}(\mathbf{x}) | \boldsymbol{\mu}_{\text{test}}, \Sigma_{\text{test}}) \tag{3}$$

$$\boldsymbol{\mu}_{\text{test}} = \mathbf{K}_*^T (\mathbf{K} + \sigma^2 \mathbf{I}) \mathbf{y}_{\text{train}} \tag{4}$$

$$\Sigma_{\text{test}} = \mathbf{K}_{**} - \mathbf{K}_*^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{K}_* \tag{5}$$

Algorithm 1 Learning GP Reward Model

input: $\mathcal{M}_S$, interaction budget T_b, data buffer $\mathcal{D} = \phi$
while T_b is available **do**
  Learn in $\mathcal{M}_S$ using any RL algorithm
  Extract source optimal policy π_S^*
  Create $\{o_i\}$ from π_S^* using method in Section III
  implement $\{o_i\}$, get $\{\mathbf{x}_i'\}$ and target rewards $\{R_i\}$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k$ where dataset $\mathcal{D}_k = \{(\mathbf{x}_i', R_i)\}$
  Calculate posterior $\hat{\mathcal{R}}_\mathcal{T}$ from $\mathcal{D}$ using Eq. 4 and 5
  Update source MDP, $\mathcal{M}_\mathcal{S} = \langle \mathcal{X}, \mathcal{A}, \hat{\mathcal{R}}_\mathcal{T}(\mathbf{x}), P \rangle$
end while
output: $\hat{\pi}_\mathcal{T}^* \leftarrow$ optimal policy from $\mathcal{M}_\mathcal{S}$

where $\mathbf{K}_{**} = k(\mathbf{X}_{\text{test}}, \mathbf{X}_{\text{test}})$, $\mathbf{K}_* = k(\mathbf{X}_{\text{test}}, \mathbf{X}_{\text{train}})$, $\mathbf{K} = k(\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{train}})$ and $k(\cdot, \cdot)$ is the kernel defined on the GP prior. Our intuition is that the positive definite kernel can capture correlation in the state-space with respect to the rewards. Throughout this letter we consider the squared exponential kernel, $k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp(-\frac{1}{2l^2}(\mathbf{x} - \mathbf{x}')^T(\mathbf{x} - \mathbf{x}'))$. Using the GP posterior as the estimated target reward function $\hat{\mathcal{R}}_\mathcal{T}(\mathbf{x})$ we can find the estimated optimal policy $\hat{\pi}_\mathcal{T}^*$ using any RL algorithm. We can repeat this whole process until we run out of T_b to improve the learned GP model $\hat{\mathcal{R}}_\mathcal{T}(\mathbf{x})$.

IV. Case Study

This letter provides a proof-of-concept autonomous manufacturing system where the task is to autonomously manufacture a type of material known as Phononic crystal (PnC). PnC, an artificially designed material with complex repetitive geometry, acts as an acoustic filter. Thus only certain sound frequencies will pass through the material while blocking the rest. Based on this property we aim to autonomously manufacture a PnC that can exhibit a passband with two output features: center frequency f_c and bandwidth w. The specific PnC we are considering in this letter has three dimensional repetitive unit cell as shown in Fig. 3. This 3D unit cell can be described by two input features: filament distance l_{xy} and filament diameter d, which dictates the passband characteristics of the entire PnC artifact.

A. Autonomous Manufacturing System for PnCs (AMSPnC)

We use a custom-built 3D printer as a proof-of-concept autonomous manufacturing system which we call AMSPnC. This modified 3D printer has a fused deposition based additive manufacturing unit for manufacturing PnC materials and an ultrasonic transducer based in-situ characterization unit to characterize the passband characteristics of PnCs in real-time. Here, each state can be represented by the input features l_{xy} and d. The software for AMSPnC is designed such that it receives these input feature values and autonomously perform the following sequence of operations; generate the Gcode, manufacture a PnC artifact, transport PnC artifact to the characterization location, characterize the artifact and post-process the passband characteristics to produce output features, $\boldsymbol{\phi}(\cdot)$.

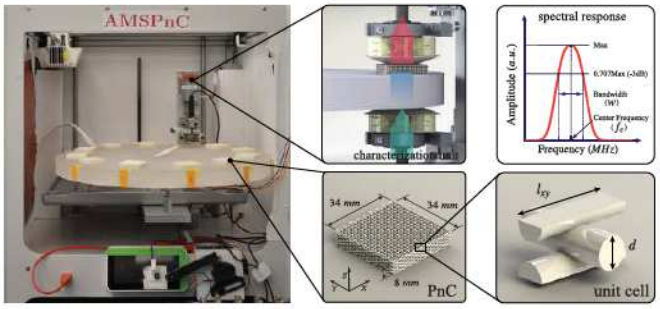


Fig. 3. AMSPnC: experimental testbed for autonomous manufacturing of PnC artifacts with 3D printing facility and ultrasonic transducer based characterization facility; an unit cell of a PnC described by two input features l_{xy}, d and corresponding spectral response are also shown.

We consider a finite range of values for each of these variables; $1050\mu m \geq l_{xy} \geq 700\mu m$ and $600\mu m \geq d \geq 350\mu m$. As we only allow a resolution of $50\mu m$ to the 3D printer, we end up with a finite state space size of $|\mathcal{X}| = 8 \times 6 = 48$. Due to numerous issues associated with additive manufacturing and heat transfer of a PnC artifact, it requires almost half an hour for a single artifact to be 3D printed and characterized. At each state the agent can take 9 possible actions where first eight actions are all the possible directions in the 2D state space and the last action is staying at the same state.

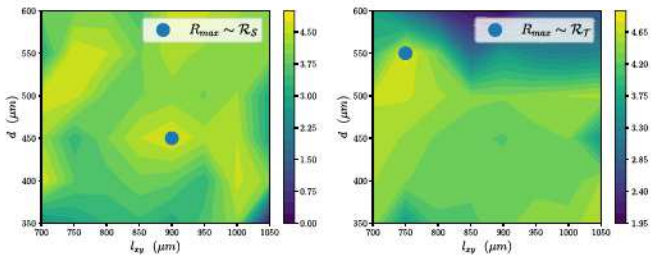
B. Dataset

We develop two novel datasets to validate the proposed methods. The source task is a collection of 48 Finite element method (FEM) simulations of PnCs with various input feature values. Each simulation takes approximately 20 minutes to perform. The target task uses an experimental dataset that is created by autonomously manufacturing PnCs using the AMSPnC machine. A full factorial Design-of-Experiment (DOE) is performed to build a dataset of 48 artifacts, where each sample is printed three times to account for the noise in the experimental system, resulting in a total of 144 physically manufactured PnC artifacts. It took almost two weeks to 3D print all the samples for this experimental dataset. We provide a feature map in Fig. 1 for both datasets that shows the distribution of the output features. Numerous studies on PnCs support the fact that numerical simulations can not predict experimental features accurately for many reasons including microscale material irregularities [18].

C. Reward Generation

The passband spectrum obtained from PnC characterization at timestep t acts as the artifact-property quantification $\mathbf{g}_t$. Now we can follow the method described in Section II-C to generate reward values. Initially, we use a two-term Gaussian approximation as $\lambda(\cdot)$ to extract features $f_{c,t}, w_t$ from $\mathbf{g}_t$ and generate corresponding feature vector $\boldsymbol{\phi}_t$. Next we use predefined fixed values, $f_{c,d}$ and w_d, as the desired output features and generate corresponding feature vector $\boldsymbol{\phi}_d$. Then we use a custom PQS metric as follows,

$$\mathcal{L}(\boldsymbol{\phi}_t, \boldsymbol{\phi}_d) = \frac{|f_{c,d} - f_{c,t}|}{f_{c,d}} + \frac{|w_d - w_t|}{w_d}. \quad (6)$$



(a) $\mathcal{R}_S$ (b) $\mathcal{R}_T$

Fig. 4. Reward distributions in (a) source and (b) target where R_{max} represents the maximum reward value.

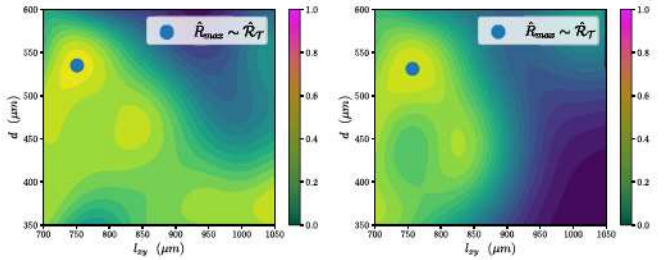


Fig. 5. Two examples of learned GP reward model using Algorithm 1. Notice that only a certain portion of the state space is explored. The reward values are normalized to have a maximum reward of 1.0.

We intentionally omitted the detail mechanism for obtaining a reward value experimentally from a 3D printed PnC artifact as it is fairly complex and less pertinent to the study of the proposed algorithm. For this letter we used $f_{c,d} = 0.85\ MHz$ and $w_d = 0.15\ MHz$ as the desired output features and baseline value of 10.0 to calculate the rewards. Finally each reward is generated using Eqn. 2. The reward distribution in the source and target task for this specification can be seen in Figs. 4(a) and 4(b).

D. Implementation Details

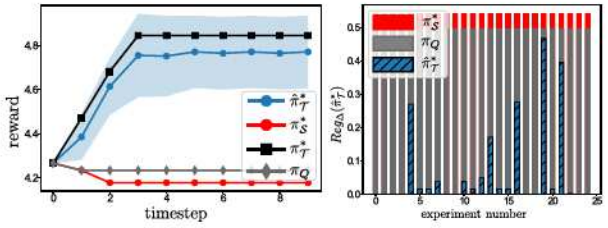
To train the agent in $\mathcal{M}_S$ we use off-policy random sample Q-learning, a simple variant of Q-learning algorithm that collects samples using a random policy and then updates the Q-values. The intention behind using random policy for data collection is to break the sequential correlation between the samples. Once trained for sufficiently long time, 10^5 timesteps in this case, a greedy policy is extracted from the Q-values to obtain π^*_S. We perform two types of experiments to demonstrate the effectiveness of our method. First, we use fixed initial input feature values, $l_{xy} = 800\mu m, d = 400\mu m$, for all experiments and later use random values within the range. During each epoch π^*_S is followed for a fixed horizon length of $H = 3$ and the final state of this trajectory acts as the subgoal state. Using the method described earlier we create 5 options from this trajectory using high exploration value, $\epsilon = 0.75$. Implementing these options provide us with a dataset $\mathcal{D}$ that we use to build a GP posterior of $\mathcal{R}_T$. We use zero mean GP as the prior without loss of generality [16]. To optimize the hyperparameters associated with the GP equations, we restart the optimization multiple times. This GP posterior, $\hat{\mathcal{R}}_T$, is used as the source reward to calculate $\hat{\pi}^*_T$ using random-sample Q-learning algorithm. Instead of using all the subgoals to create the dataset we create a set of options for one subgoal during each epoch. Later we execute the optimal option extracted from $\hat{\pi}^*_T$ from the current initial state to move to the next state that acts as the initial state for the next iteration. We repeat this process for 25 epochs. Finally the estimated target optimal policy $\hat{\pi}^*_T$ can be obtained by updating the dataset and thus updating the GP posterior $\hat{\mathcal{R}}_T$. The performance of each optimal policy is evaluated by calculating the total return for a fixed number of timesteps obtained using the benchmark target optimal policy π^*_T and then subtracting the return obtained from $\hat{\pi}^*_T$. Due to the non-episodic nature of this problem we call this performance evaluation metric Δ-*regret* where each policy is executed for Δ timesteps.

$$Reg_\Delta(\hat{\pi}^*_T) = \frac{1}{\Delta}\Big(G^{\pi^*_T}_\Delta(\mathbf{x}_0) - G^{\hat{\pi}^*_T}_\Delta(\mathbf{x}_0)\Big) \tag{7}$$

where $G^\pi_\Delta(\mathbf{x}_0) = \sum_{k=1}^{\Delta} R_k$ is the undiscounted return while policy π is executed for Δ timesteps starting from state $\mathbf{x}_0$. We perform 25 similar experiments to obtain the statistics of the optimal policy performance. To show the effectiveness of the learned policy $\hat{\pi}^*_T$ we compare it against two other traditional transfer methods, offline policy transfer and Q-function transfer. Specifically, we compare $\hat{\pi}^*_T$ against offline implementation of π^*_S in the target task and π_Q which is obtained by transferring source Q-function to the target task.

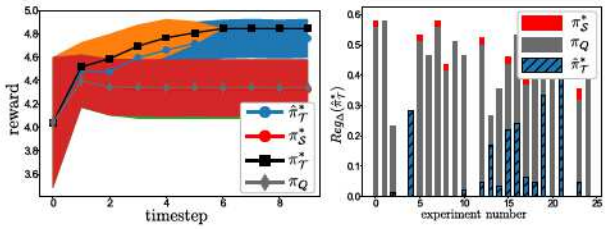
E. Results and Discussion

Two learned reward models, $\hat{\mathcal{R}}_T$, using temporal abstractions and corresponding maximum reward values are shown in Fig. 5. Here the agent interacted with the target environment 25 times but the number of visited unique states are only 22 and 15 respectively. For comparison, traditional full-factorial design process requires 144 PnC artifacts to find the high reward state which is described in Section IV-B. Notice that only a certain portion of the reward values have been learned while rest of the reward values are zero due to the GP prior with zero mean. In fact these zero rewards are not important in learning the optimal policy in the target task from this particular initial state. Hence, the agent was able to explore only a 'useful' portion of the state-space to learn the GP reward model. A comparison of the rewards is obtained by executing each policy for 10 timesteps in the target task. Here, rewards obtained from the learned optimal policy $\hat{\pi}^*_T$ is compared against rewards obtained from π^*_T, π^*_S and π_Q from Q-function transfer as shown in Figure 6(a). The optimal policy $\hat{\pi}^*_T$ obtained using the reward model performs better than the offline policy π^*_S directly implemented on the target task and performs very close to the target optimal policy π^*_T benchmark. Q-function transfer improves the policy slightly but is insignificant for this small number of timesteps. We also show the Δ-regrets obtained from all of the 25 experiments to demonstrate the effectiveness of $\hat{\pi}^*_T$. Fig. 7 shows similar results for randomly chosen initial states where, $\hat{\pi}^*_T$ outperforms π^*_S, π_Q and performs very close to π^*_T. Finally we show the 3D printed PnC artifacts suggested by an example $\hat{\pi}^*_T$ in Fig. 8. It also visualizes the spectral response corresponding



(a) obtained rewards (b) Δ-*regret*

Fig. 6. Experiments using same initial state: (a) rewards obtained using policy $\hat{\pi}^*_{\mathcal{T}}$, $\pi^*_{\mathcal{T}}$, $\pi^*_{\mathcal{S}}$ and π_Q, rewards obtained from $\hat{\pi}^*_{\mathcal{T}}$ are shown within one standard deviation range, (b) Δ-*regret* from 25 experiments calculated using Eq. (7) for the same initial state.



(a) obtained rewards (b) Δ-*regret*

Fig. 7. Experiments using random initial states: (a) rewards obtained using policy $\hat{\pi}^*_{\mathcal{T}}$, $\pi^*_{\mathcal{T}}$, $\pi^*_{\mathcal{S}}$ and π_Q, all rewards are shown within one standard deviation range, (b) Δ-*regret* from 25 experiments calculated using Eq. (7) for these random initial states.

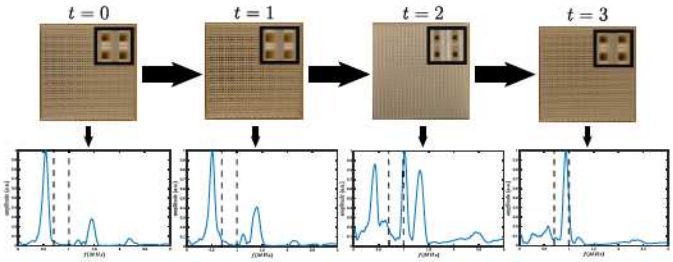


Fig. 8. An example $\hat{\pi}^*_{\mathcal{T}}$ showing evolution of manufactured PnC artifacts; top: 3D printed PnC artifacts with corresponding unit cell, bottom: spectral response corresponding to each artifact, dashed lines represent desired passband characteristics.

to each 3D printed PnC artifact. It can be clearly seen that a good correspondence between the desired and actual output features of the spectral response is obtained using $\hat{\pi}^*_{\mathcal{T}}$.

We anticipate two potential limitations of the proposed method; the initial state may affect the performance of the algorithm and the source optimal policy may not provide any useful exploration in the target task. If the source policy fails to provide useful representations to guide the agent in the target task, then this approach might not be very effective. This makes sense because the purpose of using a source task is to provide somewhat useful guidance in the target task.

V. Conclusion

For real-world implementation of RL algorithms in high variable cost environments such as autonomous manufacturing systems, transfer learning holds great potential. But this approach becomes challenging when the source reward model is inaccurate. Our proposed method of learning a probabilistic model of the target reward function using effective representations from the source task, is a sample efficient approach for building next generation of manufacturing systems. Using a real-world case study of AMS, we empirically show that our method outperforms traditional transfer learning approaches in RL. It has the ability to generate high reward, thus creating high quality manufactured artifacts, despite source reward inaccuracies and a high variable cost of sampling the target task.

Acknowledgment

The authors would like to acknowledge Zhi Zhang for helping with the experimental data collection.

References

[1] M. F. Alam, M. Shtein, K. Barton, and D. J. Hoelzle, "A physics guided reinforcement learning framework for an autonomous manufacturing system with expensive data," in *Proc. Amer. Control Conf. (ACC)*, 2021, pp. 484–490.

[2] J. R. Deneault *et al.*, "Toward autonomous additive manufacturing: Bayesian optimization on a 3D printer," *MRS Bull.*, vol. 46, no. 3, pp. 1–10, 2021.

[3] M. F. Alam, M. Shtein, K. Barton, and D. J. Hoelzle, "Autonomous manufacturing using machine learning: A computational case study with a limited manufacturing budget," in *Proc. ASME 15th Int. Manuf. Sci. Eng. Conf.*, 2020, pp. 1–12.

[4] Y. Augenstein and C. Rockstuhl, "Inverse design of nanophotonic devices with structural integrity," *ACS Photon.*, vol. 7, no. 8, pp. 2190–2196, 2020.

[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[6] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[7] A. Mirhoseini, A. H. Goldie, M. Yazgan, and J. W. Jiang, "A graph placement methodology for fast chip design," *Nature*, vol. 594, no. 7862, pp. 207–212, 2021.

[8] M. Cutler and J. P. How, "Efficient reinforcement learning for robots using informative simulated priors," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2015, pp. 2605–2612.

[9] A. Marco *et al.*, "Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017, pp. 1557–1563.

[10] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, no. 7, pp. 1633–1685, 2009.

[11] A. Lazaric, "Transfer in reinforcement learning: A framework and a survey," in *Reinforcement Learning*. Heidelberg, Germany: Springer, 2012, pp. 143–173.

[12] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration in finite Markov decision processes with Gaussian processes," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 4312–4320.

[13] M. L. Puterman, "Markov decision processes," *Handbooks Operations Research Management Science*, vol. 2. Basel, Switzerland: Baltzer, 1990, pp. 331–434.

[14] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA, USA: Athena Sci., 1996.

[15] D. Precup, *Temporal Abstraction in Reinforcement Learning*, Univ. Massachusetts Amherst, Amherst, MA, USA, 2000.

[16] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School Machice Learning*. Cham, Switzerland: Springer, 2003, pp. 63–71.

[17] F. Fernández and M. Veloso, "Probabilistic policy reuse in a reinforcement learning agent," in *Proc. 5th Int. Joint Conf. Auton. Agents Multiagent Syst.*, 2006, pp. 720–727.

[18] A. Kruisová *et al.*, "Ultrasonic bandgaps in 3D-printed periodic ceramic microlattices," *Ultrasonics*, vol. 82, pp. 91–100, Jan. 2018.