Proceedings of the ASME 2020 15th International & Manufacturing Science and Engineering Conference MSEC2020
June 22-26, 2020, Cincinnati, OH, USA

MSEC2020-8472

AUTONOMOUS MANUFACTURING USING MACHINE LEARNING: A COMPUTATIONAL CASE STUDY WITH A LIMITED MANUFACTURING BUDGET

Md Ferdous Alam
Department of Mechanical and Aerospace Engineering
The Ohio State University
Columbus, OH 43210
Email: alam.92@osu.edu

Max Shtein
Department of Materials Science and Engineering
University of Michigan
Ann Arbor, Michigan 48109
Email: mshtein@umich.edu

Kira Barton
Department of Mechanical Engineering
University of Michigan
Ann Arbor, Michigan 48109
Email: bartonkl@umich.edu

David J. Hoelzle
Department of Mechanical and Aerospace Engineering
The Ohio State University
Columbus, OH 43210
Email: hoelzle.1@osu.edu

ABSTRACT

This paper studies the concept of manufacturing systems that autonomously learn how to build parts to a user-specified performance. To perform such a function, these manufacturing systems need to be adaptable to continually change their process or design parameters based on new data, have inline performance sensing to generate data, and have a cognition element to learn the correct process or design parameters to achieve the specified performance. Here, we study the cognition element, investigating a panel of supervised and reinforcement learning machine learning algorithms on a computational emulation of a manufacturing process, focusing on machine learning algorithms that perform well under a limited manufacturing, thus data generation, budget. The case manufacturing study is for the manufacture of an acoustic metamaterial and performance is defined by a metric of conformity with a desired acoustic transmission spectra. We find that offline supervised learning algorithms, which dominate the machine learning community, require an infeasible number of manufacturing observations to suitably optimize the manufacturing process. Online algorithms, which continually modify the parameter search space to focus in on favorable parameter sets, show the potential to optimize a manufacturing process under a considerably smaller manufacturing budget.

NOMENCLATURE

T	Total budget/observation number
	Loss
	Design space or state-space
	Dataset
$\mathbf{x}$	Optimum value of inputs
$\mathbf{x}$	Vector of input variables
$\mathbf{y}$	Output
$\mathbf{y}_d$	Desired output
	policy of reinforcement learning agent
	Reward function
	action-space

INTRODUCTION

It is inarguable that in 2020 we are in an era of manufacturing efficiency and mass production that has never been seen before [1]. Modern manufacturing economies in the West and in Asia have automated traditional manufacturing processes such as stamping, forging, casting, machining, and assembly [2]. Whereas human laborers previously performed material handling and machine sequencing tasks, automated systems and robots have replaced the laborer because they are relatively inexpensive, repeatable, and do not tire. However, the human laborers they have replaced had superior human cognition; automation robots are capable of following a sequence of tasks, but rarely make decisions. The human ability to plan, analyze, and intervene, and thus intelligently modify a manufacturing processes by local material work, local heat transfer, and local material addition and subtraction has been replaced in favor of tight process operating windows and robotic, repeatable actions for superior efficiency.

Given this unstoppable trajectory of manufacturing automation for mass manufactured components, it is worth considering what the next big movement in manufacturing will be. The authors and others [3–5] believe that the next generation of manufacturing automation will enable to manufacturing machines to plan, analyze, and intervene; instead of automated robotic manufacturing, the next generation of machines will be autonomous. The machines will have the actuation dexterity, sensing capabilities, and cognition for planning, analysis, and intervention that far surpasses human capabilities.

This paper investigates cognition elements that are based on the methods of machine learning (ML). As defined by Murphy [6], "ML is a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty." The three main classes of ML are supervised learning (SL), unsupervised learning, and reinforcement learning (RL). Typically, ML is applied to a dataset $\mathbf{X}\ \mathbf{Y}$, $\mathbf{X}$ $T\ d$, $\mathbf{Y}$ $T\ m$, where the dataset is comprised of T observation pairs $\mathbf{x}_t\ \mathbf{y}_t$, $\mathbf{x}_t$ are the d independent parameters or inputs, $\mathbf{y}_t$ is the set of n dependent output variables. In the general supervised learning problem, the objective is to uncover patterns in . In the context of the general manufacturing problem, $\mathbf{x}_t$ is the set of independent tunable parameters for a part design or process variable and $\mathbf{y}_t$ is an output metric or metrics of interest. When m 1 the output is a scalar and when m 1 the output is a vector. In unsupervised learning, y_t is unavailable ($\mathbf{X}$) and the objective is to cluster the independent parameters. Although unsupervised learning may have applications to manufacturing, they are not studied here. Both supervised and unsupervised learning algorithms are often applied to an existing dataset, and we term these offline algorithms here, where it is assumed that a dataset has been collected on a manufacturing system. In contrast, the aim of reinforcement learning (RL) is to continually explore the domain of the independent variable space to maximize a reward (sometimes as a function of $\mathbf{y}$) or goodness of manufacturing performance. RL is an online algorithm where current data informs the next action, or movement, in the domain. In the standard ML problem, it is inexpensive to populate ; for example, classical ML problems such as the modified NIST data [7] and the Netflix Prize [8] have T values in the range of 10^5 – 10^9. In contrast, the general manufacturing process is relatively expensive, requiring materials and labor and occupying plant infrastructure, thus limiting the acceptable number of trials before production quality parts are produced to be on the order of 10^2.

For a manufacturing system to learn, the system must be automated, such that manufacturing or part design parameters can be automatically specified by the algorithm, thus $\mathbf{x}_t$ can be specified, and the system must have automatic metrology, to quantify the quality or performance of what is manufactured, and thus generate observation data y_t. Importantly, the system must have an ML engine for cognition, either to understand underlying patterns in dataset or perform actions in a RL framework. This paper explores such a system, particularly in the case of expensive data, where T is budgeted and there is an advantage to converging to a good performance in as few of observations, thus manufacturing operations, as possible. We explore both offline and online algorithms and compare part performance at different budgets, T. Some online algorithms have a budgetary advantage because the algorithms probabilistically travel along the path of highest reward and thus ignoring low-reward regions of the domain. The paper is organized as follows. The Methodology section presents a generic mathematical framework for manufacturing systems, definition of a performance index to measure the manufacturing performance, and application of both offline and online ML algorithms with manufacturing budgets. The Computational Emulation of Manufacturing System section provides a case study for hypothetical system and ML application for the manufacture of a Phononic Crystal (PnC) (Fig. 1); this manufacturing system, based on an additive manufacturing (AM) platform, can dexterously change the design or process parameters, $\mathbf{x}_t$, by changing G-Code instructions, has inline acoustic spectra sensing to generate y, and has a cognition element to autonomously manufacture, thus sample, in the design and process workspace to generate data and evolve part performance. The Machine Learning Application and Results Section describes the specifics of the panel of ML algorithms applied and compares the performance metric for different manufacturing budgets for each algorithm. The Conclusion section provides interpretations and suggests future research directions.

METHODOLOGY

In this section, a generic mathematical framework for a manufacturing system capable of learning from data or previous experiences is developed. Consider the general manufacturing system modeled by

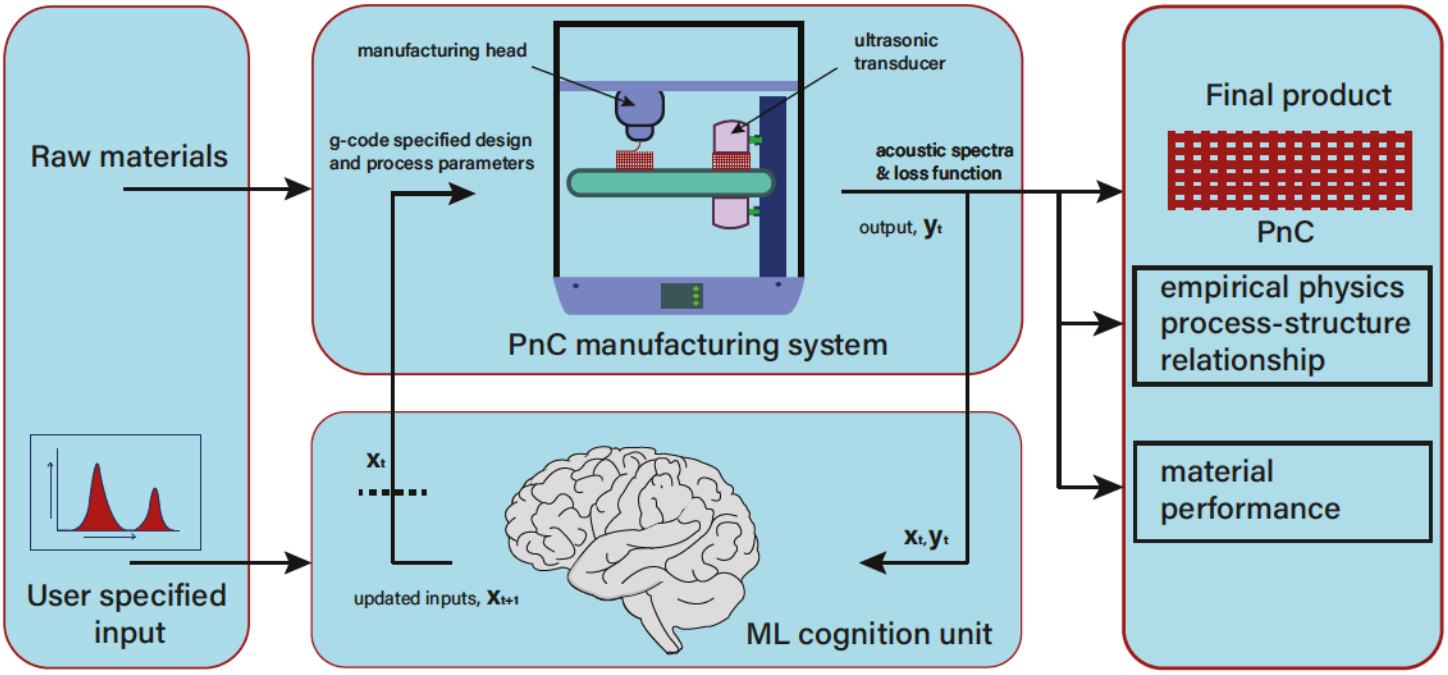

FIGURE 1. Autonomous Manufacturing System of Phononic crystals (PnC) with ML unit as the cognitive element (dotted line represents the beginning of a new timestep).

$$\mathbf{y}_t = f(\mathbf{x}_t) + \boldsymbol{\varepsilon} \tag{1}$$

where, $f(\mathbf{x}_t) : \mathbb{R}^d \to \mathbb{R}^n$ is the mapping between the controllable inputs, $\mathbf{x}_t$, and the true, or measured, output features of the system, $\mathbf{y}_t$, where $\mathbf{y}_t$ can be a function or scalar $(n = 1)$ or vector-valued function or vector $(n > 1)$, depending on the definition of an output feature of a specific system, and $t \in \mathbb{Z}_+$ is manufacturing observation index. $\boldsymbol{\varepsilon}$ is a stochastic variable that captures uncontrollable inputs, process variability, and measurement noise. We assume that the controllable inputs are bounded: $\mathbf{x}_t = [x_{t,1}, x_{t,2}, \ldots, x_{t,d}] \in \mathscr{X}$ where $\mathscr{X}$ is the corresponding vector space satisfying the inequality

$$\underline{\mathbf{x}} \preceq \mathbf{x} \preceq \bar{\mathbf{x}}$$

where $\underline{\mathbf{x}}$ is the minimum allowable limit of the input variables and $\bar{\mathbf{x}}$ is the maximum allowable limit of the input variables. Here, $\preceq$ is used to define the element-wise inequality: $\mathbf{x}_1 \preceq \mathbf{x}_2$ means that $[\mathbf{x}_1]_i \leq [\mathbf{x}_2]_i \; \forall i$.

Optimization problem

Considering the definition of the manufacturing system in Eqn.(1), our goal is to find $\mathbf{x}^* \in \mathscr{X}$ such that $|\mathbf{y}_d - \mathbf{y}^*| = |\mathbf{y}_d - \mathbb{E}[(f(\mathbf{x}^*)]|$ is minimal, where $\mathbf{y}_d$ is a desired output and $|\cdot|$ is a generic norm operator. Note that, in general, $|\mathbf{y}_d - \mathbf{y}^*| = 0$ is not feasible, as one can specify a desired output that is not achievable for a manufacturing system; for the case example in this paper, one could specify a desired transmission spectra that is an ideal filter (top-hat filter) with perfect transmission in the pass-band and perfect rejection in the stop-band, which is not realizable for physical systems. Additionally, it is extremely difficult to obtain optimality with a fixed budget due to the stochastic noise in the system. Hence, we wish to find a sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ with a total budget of T such that we can end up in the vicinity of $\mathbf{x}^*$.

Performance index

The generic optimization problem statement in the previous section is ill-posed as vector-valued output features, $\mathbf{y}_t$, create a multi-objective optimization problem, with the potential to have competing objectives. A common method for this issue is the scalarization of such a multi-objective optimization problem with the definition of a 'loss function' or 'cost function',

$$\mathscr{L}(\mathbf{y}) = [w_1, w_2, \ldots, w_n]\,[\mathscr{L}_1(y_1), \mathscr{L}_2(y_2), \ldots, \mathscr{L}_n(y_n)]^T : \mathbb{R}^n \to \mathbb{R}$$

where each individual loss function is a mapping $\mathscr{L}_i(y_i) > 0 : \mathbb{R} \to \mathbb{R}$ and each $w_i \geq 0$. Leveraging the loss function, our opti-

mization problem is formulated as

$$\begin{aligned} & \min_{\mathbf{x}} && f\ \mathbf{x} \\ \text{s.t.}\ & \underline{\mathbf{x}} && \mathbf{x}_t \quad \bar{\mathbf{x}} \\ & t && 0\ 1 \quad T \end{aligned} \tag{2}$$

Ideally the loss function is convex and zeroth order, first order or second order optimization algorithms are typically used to minimize the loss function. Non-convex loss functions pose a more complicated optimization problem than the ideal one as there might be multiple local optima with one or more global optima. If the loss function is not differentiable then gradient-based optimization algorithms are difficult to implement as there may be multiple points with discontinuities [9–11].

Optimization of the loss function: Online vs Offline

The solution to Eqn.(2) is the main problem statement of standard ML problem. Here, we investigate offline, SL-based learning methods as a gold-standard for comparison of the optimization problem. We define offline to be the optimization of Eqn.(2) when the dataset $\mathbf{X}\ \mathbf{Y}$ is available *a priori*; note that the acquisition of may be prohibitively expensive for manufacturing systems. We define online to be the optimization of Eqn.(2) with no *a priori* information and the information at a given operation observation t ($\mathbf{X}\ \mathbf{Y}$, $\mathbf{X}^{\ t\ d}$ and $\mathbf{Y}^{\ t\ m}$) directs the selection of the next input set $\mathbf{x}_{t\ 1}$ to observe.

Offline methods Most of the popular machine learning algorithms are offline methods and hence not suitable to implement in a problem setting with limited or expensive data. Although we will implement commonly used supervised learning techniques to show the best case scenario, the performance of supervised algorithms become poor with fewer training samples. There are also gradient based and gradient free optimization methods which may reach the global/local minimum with reasonably good accuracy, but these algorithms have a major limitation. Gradient based optimization methods can achieve convergence for convex objective functions fairly well, but fail to ensure optimality for non-smooth (smooth in the sense of Lipschitz), non-convex objective functions. Gradient free optimization methods are computation intensive and need large datasets to obtain optimality with convergence. As any manufacturing system is a real-world process and incorporates various inherent noise to the system, it is highly unlikely that we will end up with a smooth convex objective function to optimize and as mentioned in the previous section, we might get stuck at one of the many local minima.

Online methods We can extend the optimization problem to the online setting without any prior knowledge, unlike most of the offline optimization problem where a training set, $\mathbf{X}\ \mathbf{Y}$ is available beforehand. Online optimization problems can utilize either one new input-output relationship (with knowledge of already experienced values) or they can utilize small batches of data (often called batch optimization). A convenient approach often followed for this purpose is the "Response Surface Method" (RSM), which is sometimes used analogously with global metamodel optimization [12]. In case of global metamodel optimization, a global model is fit to the dataset, while in case of RSM, local models are fit into small batches of data. The fundamental idea behind RSM is simple and sketched in algorithm 1. Starting with a randomly selected state we fit a local model (unlike a global metamodel) to a sampled region around the starting state at every iteration and use search methods like Stochastic Gradient Descent (SGD) [13] to find the minimum of that region. This minimum acts as the starting state for the next iteration. By state, we define the vector of all input variables ($\mathbf{x}$) at a single point. From the pseudocode we can view RSM as a batch online method that takes a small batch of data at each iteration.

Algorithm 1 RSM pseudo code

Require: design space , dimension of sampled region (e.g. length $_{rsm}$ for square region in 2D design space), maximum iteration number: $iter_{max}$, random initial state $\mathbf{x}_0$

1: **while** $step$ $iter_{max}$ **do**
2: state $\mathbf{x}_0$
3: create sampled region S around $\mathbf{x}_0$ (e.g. square region with length = $_{rsm}$)
4: randomly choose n points from S
5: S_f new region by fitting n points using lower order polynomials
6: $\mathbf{x}_{opt}$ global minimum on S_f using SGD
7: $\mathbf{x}_{next}$: $\min_{\mathbf{x}_{next}}$ $\mathbf{x}_{next}$ $\mathbf{x}_{opt\ 1}$
8: $\mathbf{x}_0$ $\mathbf{x}_{next}$
9: **end while**

Reinforcement learning (RL) has the correct properties to be a good candidate in our problem setting due to its inherent online nature of learning [14]. Reinforcement learning has been proved to be an effective sequential decision making process under uncertainty where an agent (multiple agents in some cases [15]) interacts with its dynamic environment to achieve a goal and obtains rewards as the outcome of this interaction. This learning scheme typically works in the context of incomplete Markov Decision Process (MDP) [16]; incomplete in the sense of unavailability of all the state-transition probabilities . Formally

a finite MDP can be expressed as the tuple , where, is the finite set of states, is the finite set actions, is the state transition probabilities, is the reward function and is the discount factor 0 1 . The behavior of the agent is described by policy $a\,\mathbf{x}$ $Pr\,A_t$ $a\,\mathbf{x}_t$ $\mathbf{x}$ $\mathbf{x}$ which is basically a mapping from the state-space to the action-space, : . The reinforcement learning agent follows policy to move from current state $\mathbf{x}$ to next state $\mathbf{x}$ and receives reward $r\,\mathbf{x}\,\mathbf{x}$ a $\mathbf{x}\,\mathbf{x}$ a . The goal of reinforcement learning is to estimate the "value function", V that estimates the future expected rewards (often discounted) while the agent is in the current state.

$$V\ \mathbf{x} \quad G_t\,\mathbf{x}_t\ \ \mathbf{x} \quad \sum_{k\ 0}{}^{k}R_{t\ k\ 1}\,\mathbf{x}_t\ \ \mathbf{x}\ \ \mathbf{x} \tag{3}$$

where, return G_t $R_{t\ 1}$ $R_{t\ 2}$ ${}^{2}R_{t\ 2}^{2}$ and $r\,\mathbf{x}_t\ a_t\ \mathbf{x}_t$ R_t. Usually action-value function $Q\,\mathbf{x}\ a$ is used instead of value function, $V\ \mathbf{x}$ to define the value of taking action a in current state $\mathbf{x}$ while following policy

$$Q\ \ \mathbf{x}\ a \quad G_t\,\mathbf{x}_t\ \ \mathbf{x}\ a_t\ \ a \tag{4}$$

$$\sum_{k\ 0}{}^{k}R_{t\ k\ 1}\,\mathbf{x}_t\ \ \mathbf{x}\ a_t\ \ a\ \ \mathbf{x} \tag{5}$$

Formally this problem becomes finding the optimal policy that maximizes $Q\ \ \mathbf{x}\ a$,

$$Q\ \ \mathbf{x}\ a \quad \max Q\ \ \mathbf{x}\ a\ \ \mathbf{x} \tag{6}$$

COMPUTATIONAL EMULATION OF A MANUFACTURING SYSTEM

The goal of the case study is to evaluate the methods described in the previous sections on simulation of a manufacturing process: the manufacture of acoustic metamaterial crystals with a fused deposition (FDM) based AM system. These metamaterials, also commonly known as phononic crystals (PnCs), exhibit special acoustic bandgap properties corresponding to the geometric parameters [17, 18]. PnCs can be fabricated via additive manufacturing and tested using piezoelectric transducers as shown in Fig. 2(a) and 2(b). The bandstructure of PnCs can be parameterized by their geometric design parameters, such as the filament diameter and lattice constant (Fig. 2(a)), and desired bandpass can be obtained by proper tuning of those parameters. In the following sections we provide a brief overview of the bandstructure analysis of PnCs with respect to geometric parameters and the development of suitable learning framework for this specific manufacturing system.

Bandstructure analysis of PnC

In this study, the band structure calculations are performed on a representative unit cell, which is the building block of the PnC and repeats periodically along the three spatial directions $x\ y\ z$. Fig. 2(a) shows the unit cell used in this study. A computational engine is developed that uses the acoustics module package of COMSOL® as its core to solve the elastic wave propagation problem by Finite Element Method (FEM) at discrete parameter values of the lattice constant, l_{xy} 700 705 1035 m, and filament diameter, d 300 305 635 m, for a total of 4624 unique design parameter combinations. More specifically, these band structures are calculated through corresponding numerical eigenfrequency analysis. The traditional frequency-wave vector k dispersion relationship is extracted from the eigenfrequency analysis by sweeping the wavevector k in the first symmetric Brillouin zone $X\ S\ Y$ $Z\ U\ R\ T\ Z\ Y$ $T\ U$ $X\ S$ R. The structure of the Irreducible Brillouin Zone (IBZ) comes from the symmetry of the unit cell. As this study mainly focuses on the learning algorithms we will skip the details of the IBZ and the eigenfrequency analysis of PnCs; interested readers are suggested to look into [19, 20]. We consider only the Z direction as experimental phononic studies typically use longitudinal waves for PnC characterization as shown in Fig. 2(b). The set of dominantly longitudinal band structures is synthesized into the transmission spectra by labeling as bandpass in frequency regions in which there is transmission and bandstop in frequency regions in which there is no transmission and then Gaussian kernel based filtering to smooth the spectra. The outcome spectra from this method is qualitatively comparable to the experimentally derived spectra from [21]. A representative desired spectra and a typical actual spectra from the simulation are shown in Fig. 3. These spectra are characterized with respect to the desired peak amplitudes ($y_{d_{c1}}$ $y_{d_{c2}}$), actual peak amplitudes ($y_{a_{c1}}$ $y_{a_{c2}}$), desired peak frequencies ($f_{d_{c1}}$ $f_{d_{c2}}$) and actual peak frequencies ($f_{a_{c1}}$ $f_{a_{c2}}$).

Learning problem formulation

As mentioned in the previous sections, our goal is to implement and analyze the performance of various machine learning algorithms - both online and offline. One advantage with this computational study is that we can consider supervised algorithms as a hypothetical, best-case scenario of having all the data *a priori*.

In our study we impose some constraints to mimic the actual manufacturing systems. It is assumed throughout the study that our system is only capable of manufacturing PnCs with filament-to-filament distance (l_{xy}) and filament diameter (d) within a certain range. We can describe the manufacturing system in Eqn.(1) with input vector $\mathbf{x}$ $l_{xy}\ d\ ^{T}$, output function y is the spectra, is the noise, with the noise distribution considered to be un-

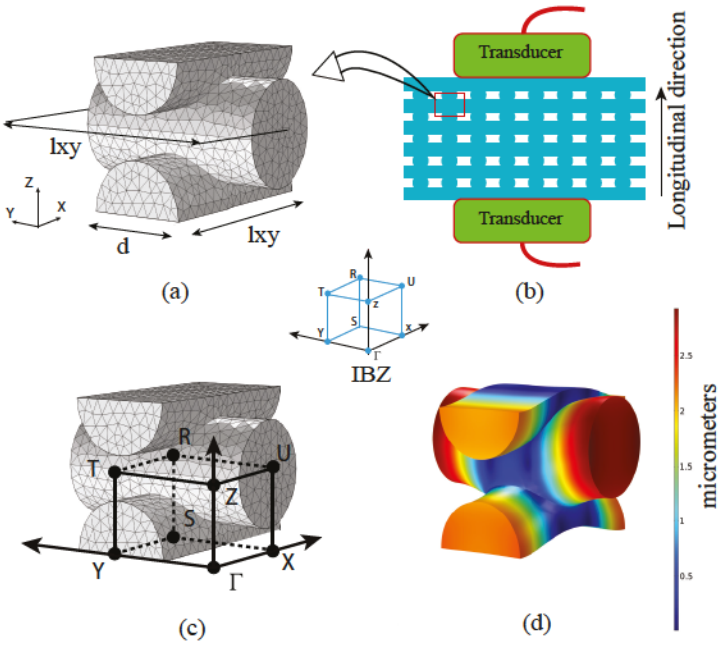

FIGURE 2. Unit cell description and PnC characterization: (a) Unit cell with mesh and design parameters l_{xy} and d, (b) PnC Characterization using ultrasonic transducers, (c) IBZ structure of the unit cell, (d) mode shape showing dominant longitudinal deformation.

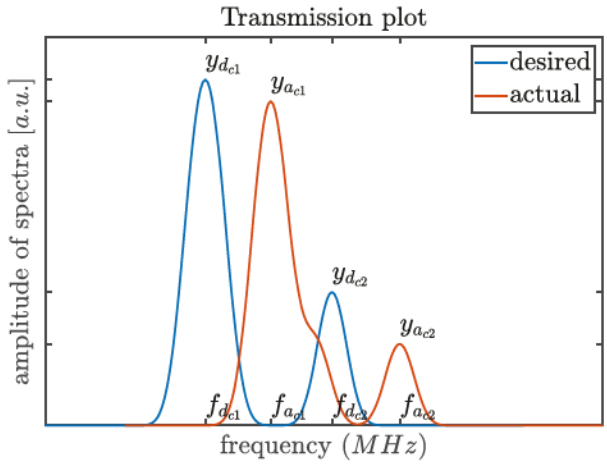

FIGURE 3. Typical desired and actual spectra: Desired spectra and actual spectra characterized by peak amplitudes and peak frequencies.

known. The key idea is that we assume we do not have a model of the system which encourages us to consider model-free online learning methods later in this section. Ultimately we can express the optimization problem for this computational manufacturing system as the following:

$$\begin{aligned} &\min_{l_{xy},d} \ \mathbb{E}[\mathscr{L}(y_a(l_{xy},d),y_d)] \\ &\text{s.t.} \quad \begin{bmatrix}700\mu m\\ 300\mu m\end{bmatrix} \preceq \begin{bmatrix}l_{xy}\\ d\end{bmatrix}_t \preceq \begin{bmatrix}1035\mu m\\ 635\mu m\end{bmatrix} \\ &\quad t \in [0,1,\ldots,T] \end{aligned} \tag{7}$$

To assess the performance as the budget changes we consider t as a variable. We construct the loss function $\mathscr{L}(y_d,y_a)$ from the mathematical comparison of user specified desired spectra (y_d) and actual spectra obtained from the computational eigenfrequency analysis (y_a). We design the following loss:

$$\mathscr{L}(y_d,y_a) = \mathscr{L}_1 + a\mathscr{L}_2 + b\mathscr{L}_3 \tag{8}$$

where,

$\mathscr{L}_1 = \sum_{i=1}^{m} |y_{ai} - y_{di}|$

$\mathscr{L}_2 = \frac{|f_{ac1}-f_{dc1}|}{f_{dc1}} + \frac{|f_{ac2}-f_{dc2}|}{f_{dc2}} + \frac{|f_{ac1}-f_{ac2}|}{|f_{dc1}-f_{dc2}|}$

$\mathscr{L}_3 = \frac{|y_{ac1}-y_{dc1}|}{y_{dc1}} + \frac{|y_{ac2}-y_{dc2}|}{y_{dc2}}$

Here, $\mathscr{L}_1$ compares the amplitude of the desired and actual spectra, $\mathscr{L}_2$ is a metric of the difference between peak amplitude frequency of the actual and desired spectra, normalized by the desired peak amplitude frequency, to quantify the position of the peak within the transmission bandwidth, $\mathscr{L}_3$ is a metric of the difference between peak amplitude of the actual and desired spectra, normalized by the desired amplitude, to quantify the transmission level within the transmission bandwidth. Fig. 4

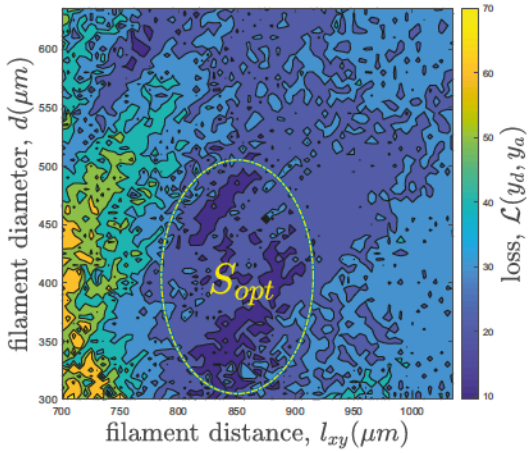

FIGURE 4. Map of loss over the domain of $\mathscr{X}$: same color means equal loss.

shows the map of the above loss function over the domain (design space). The loss function is non-smooth and non-convex for the computational dataset described in the bandstructure analysis section. The non-smoothness occurs because the simulation results are found to be highly sensitive for small change in dimensions of the PnCs. This sensitive loss mimics the experimental loss function as both of them associate noise of very random nature. Nonetheless this contour gives us a global minimum which shows close proximity with the experimental result of [21]. Due to the presence of many local optima, we choose to define an optimum region S_{opt} which we treat as the most promising region of (dark blue region with boundary in Fig. 4). This region, S_{opt} corresponds to low values of loss (y_d y_a 25).

MACHINE LEARNING APPLICATION AND RESULTS

Methods

In this section we present the details of each algorithm applied to the dataset and the important results from the study.

Supervised learning algorithms In this study, some of the widely used supervised algorithms like Polynomial Regression, Decision, Tree, Kernel Ridge Regression, Support Vector Regression (SVR) [6, 22] and multilayer Neural Networks (NN) [23] are applied. Each supervised algorithm is performed for various budgets (T 5 125 250 500 1000 2000 3000 4000) to incorporate the worst case and best case scenario for the amount of data available. The sampled data is randomly selected from the entire dataset of n 4624 observations and each algorithm is independently run 15 times to understand the range in algorithmic performance. Evaluating the performance of each supervised algorithm is straightforward. After each algorithm fits the data we determine the global minimum of loss obtained from the supervised model and calculate the corresponding optimum input variables, $\mathbf{x}$ l_{xy} d ^{T}. In case of multiple points with lowest value of loss we randomly choose one point which is compensated by running each algorithm multiple times. Before implementing any algorithm, the whole dataset is split into 90% training data and 10% test data to check the accuracy of the model on unseen data. Most of the algorithms have hyperparameters (e.g. polynomial regression quality is dependent on polynomial degree and decision tree is dependent on tree depth) which directly impact model accuracy. To overcome this issue of manually choosing the hyperparameters, 10 $fold$ cross validation [22] (when T 10) is used for polynomial regression, decision tree, kernel ridge regression and SVR; for example: polynomial degree are varied between 0 21 while tree depth are varied from 3 20 for better accuracy. In case of kernel ridge regression, radial basis functions are chosen as the kernel. We define the prediction error of the loss obtained from each algorithm as $_{TR}$ for training data and $_{TE}$ for test data where the errors are mean squared error (MSE) expressed as below:

$$MSE \quad \hat{} \quad \frac{1}{n_s} \sum_{i\ 1}^{n_s} \quad i \quad \hat{}_i^{\,2}$$

where n_s is the total number of samples used to calculate the MSE, $_i$ and $\hat{}_i$ are the actual loss and predicted loss of the ith sample, respectively.

Response surface method As previously mentioned we use response surface method (RSM) to fit a local metamodel to a sampled dataset obtained from the full dataset (pseudo code 1). In this study we have a 2D design space , so we opted for rectangular sampling topographies. Various combinations of sampled square length, $_{rsm}$ and sample number, n are tested in this study, while the intuition behind these values is that higher values of $_{rsm}$ and n will have a larger region, S to fit. A larger S will increase the amount of data to feed the algorithm at each iteration but will ensure better fit as it is difficult to get a good fit with very low number of data samples. To balance these issues $_{rsm}$ 10 and n 100 have been used in this study. Initially lower order polynomials (degree = 2,3) are used to fit the data and at later steps of iterations (30% of total iterations) higher degree of polynomials (degree = 4) are used to fit the data.

Reinforcement learning algorithms Our goal is to find the state which maximizes the expected future rewards as described in (6). Different types of reward functions, r $\mathbf{x}$ a $\mathbf{x}$ are applied in this context and the following function is defined as the suitable reward due to its ability to incorporate the loss at each state:

$$r \ \mathbf{x} \ \mathbf{x} \quad a \quad f \quad M \quad \mathbf{x} \ \mathbf{x} \qquad (9)$$

where, M is an arbitrary high constant value. The purpose of M is to build an inverse relationship between the loss and reward while keeping the reward as a positive value. It is clear from Eqn.(9) that the current reward comes from the value of loss at the current state, which makes it a noisy reward due to the random noise existent in the original system. While many reinforcement learning problems deal with episodic tasks and constant binary rewards, in this case, it is hard to do so. Due to lack of *a priori* information about the position of minimum loss on we can not define a terminal state in our problem like many other episodic implementations of RL. Instead we have a continuing task problem. In this study we consider an on-policy algorithm called "Differential SARSA" [16], which uses undiscounted average reward for continuing tasks while using the temporal difference (TD) update

rule for updating the Q-value. The differential return (G_{td}) and TD update rule with TD error (δ) can be expressed as follows:

$$\begin{aligned} G_{td} &= R_{t+1} - \bar{R} + R_{t+2} - \bar{R} + \dots \\ \delta &= R - \bar{R} + Q(\mathbf{x}', a') - Q(\mathbf{x}, a) \\ \bar{R} &= \bar{R} + \beta\delta \\ Q(\mathbf{x}, a) &\leftarrow Q(\mathbf{x}, a) + \alpha\delta \end{aligned} \tag{10}$$

In this study $\varepsilon - greedy$ policy is used with slightly higher exploration factor ($\varepsilon = 0.2$). Like most RL problems balancing the exploration-exploitation is a bit of a challenge and due to the high non-convexity and non-smoothness of the loss function and limited budgets, our agent is often stuck at local minima in the no exploration case. We tried different action spaces and eventually adopt the "king's move" as possible actions of the agent from each current state, $\mathbf{x}$. The "king's move" allows the agent to move in eight directions from any state with certain step size ($\Delta = 5\mu m$). It adds more degrees of freedom for the agent to explore the search space than an agent which is allowed to move only in four directions. The agent starts from a random state within design space $\mathscr{X}$. If the agent hits the boundary of $\mathscr{X}$, it gets a zero reward and chooses a random state within $\mathscr{X}$ as the next state. Other hyperparameters for learning have been set as: $\alpha = 0.5, \beta = 0.3$.

Results

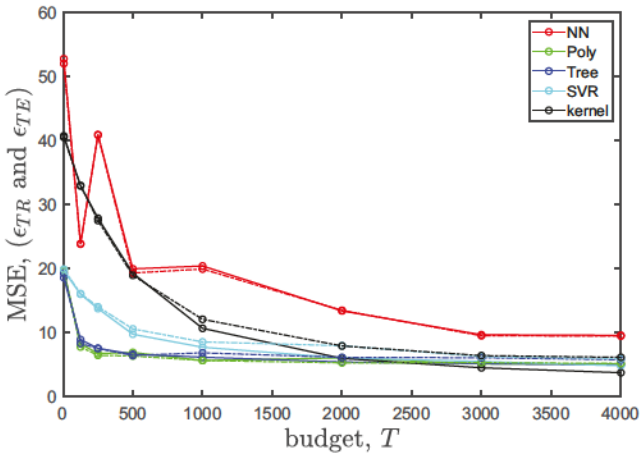

FIGURE 5. Train and test error: Mean squared error of SL algorithms (solid lines represent training MSE (ε_{TR}) and dashed lines represent test MSE (ε_{TE}).

For supervised algorithms training error (ε_{TR}) and test error (ε_{TE}) with respective to budgets are shown in Fig. 5. Both ε_{TR} and ε_{TE} decrease with higher training samples as expected. Close values of ε_{TR} and ε_{TR} ensures the confidence of the models in avoiding issues like overfitting. Although one can find optimum points (points with minimum loss) for higher training data, the optimum point is not near the optimum region S_{opt} most of the times. Even if we use large amount of training data (≥ 3000 samples) it is quite difficult to predict the optimum loss for any supervised algorithm as described in Fig. 6. The optimum loss is

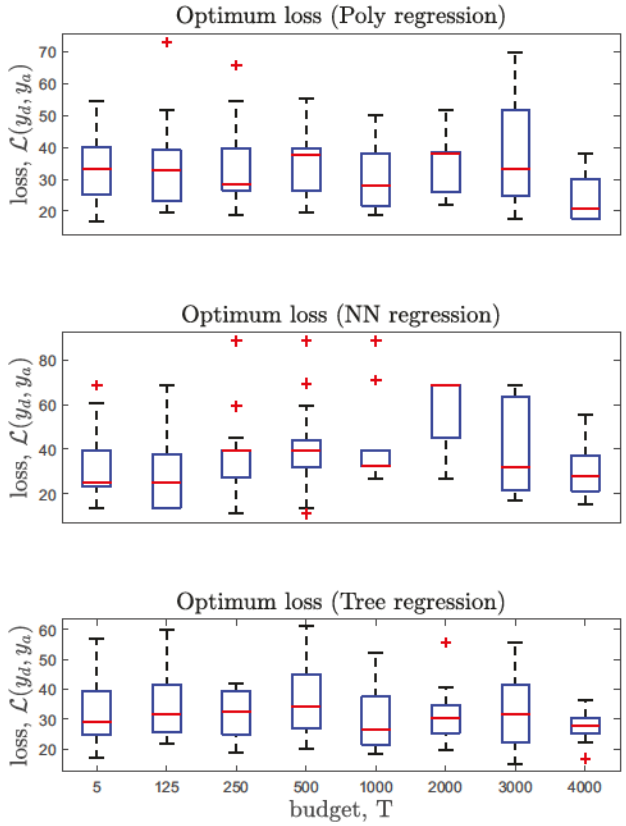

FIGURE 6. Optimum value of loss provided for selected supervised algorithms: Poly regression means Polynomial regression, NN regression means Neural network regression and Tree regression means Decision tree regression

not consistent with the number of budgets and it varies within a large range of values for the same amount of budget. Only 4000 training data show consistent low values of the loss and almost all the supervised algorithms suffer from this issue. The random noise in the manufacturing system makes it hard for the super-

vised algorithms to find the optimum points near S_{opt}. Note that the models would have been able to provide better results if the probability distribution of the noise was known (e.g. Gaussian noise). An example fit of the data is shown in Fig. 7 where mul-

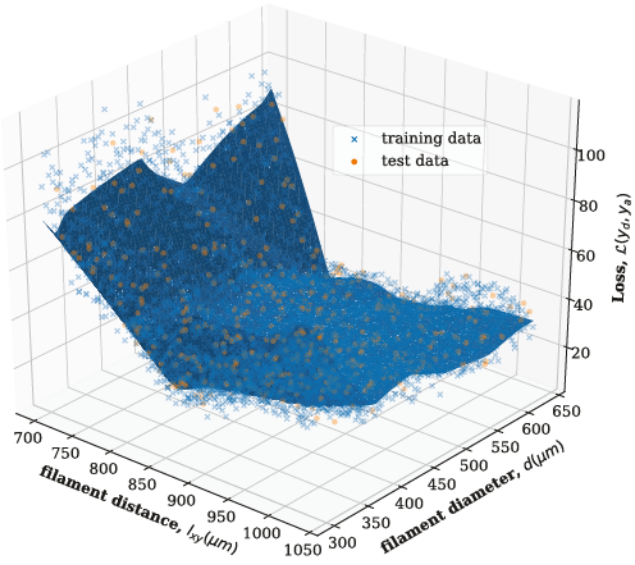

FIGURE 7. **Example fit of data**: Loss function fit by a multilayer neural network.

tilayer neural network regression is implemented.

On the other hand, Fig. 8 shows result from implementation of RSM with lower order polynomials. The RSM algorithm is able to minimize the loss fairly well with an overall budget of T = 1500. In most cases the value of the loss converges to a small value ($\mathscr{L}(y_d, y_a) < 25$) although in some cases the algorithm fails to converge to a lower value of loss than the initial loss. As shown in Fig. 8, 5 attempts out of 7 are able to converge to $\mathscr{L}(y_d, y_a) < 25$. The average reward SARSA is able to reach the "optimum region" (S_{opt}) within a limited number of budgets (T = 200) although the algorithm is not invariant to starting state. In some cases starting from a random state the agent fails to reach within the close proximity of the region S_{opt}. This is not surprising because of the noisy reward obtained by the agent at each timestep. Unlike typical deterministic reward functions, the reward function used in this study is a function of the noisy loss. As the algorithm is fed noisy reward it can not make a better decision following the deterministic policy $\pi(\mathbf{x}|a)$ in an otherwise well defined problem formulation. Nonetheless average reward SARSA is often capable of reaching within the close proximity of S_{opt} with small budget, while data intensive supervised techniques are not capable of producing a similar result with com-

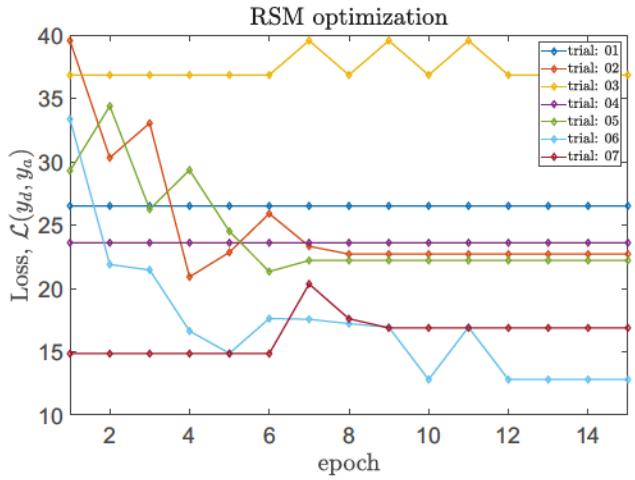

FIGURE 8. **Loss from RSM:** Loss values obtained from multiple RSM runs, trial 3 shows a non-convergent behavior (failed to find S_{opt}) unlike other trials.

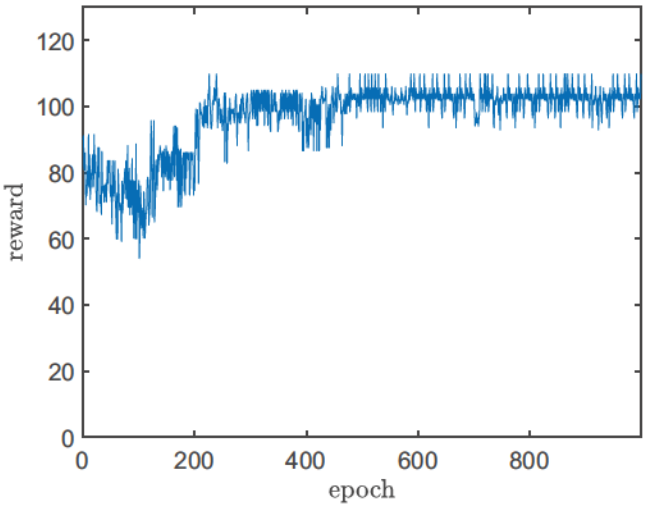

FIGURE 9. **RL for larger timesteps:** Reward obtained from RL algorithm (differential SARSA) with a large budget (T = 1000). Occasional drops in the reward are a result of exploration done by the agent.

paratively higher budgets. Fig. 9 shows a special scenario where the RL agent is allowed to operate for an extended period of timesteps. This figure states an interesting fact where the agent is capable of finding the maximum reward most of the times once the $Q-value$ space has been mapped after about 500 timesteps.

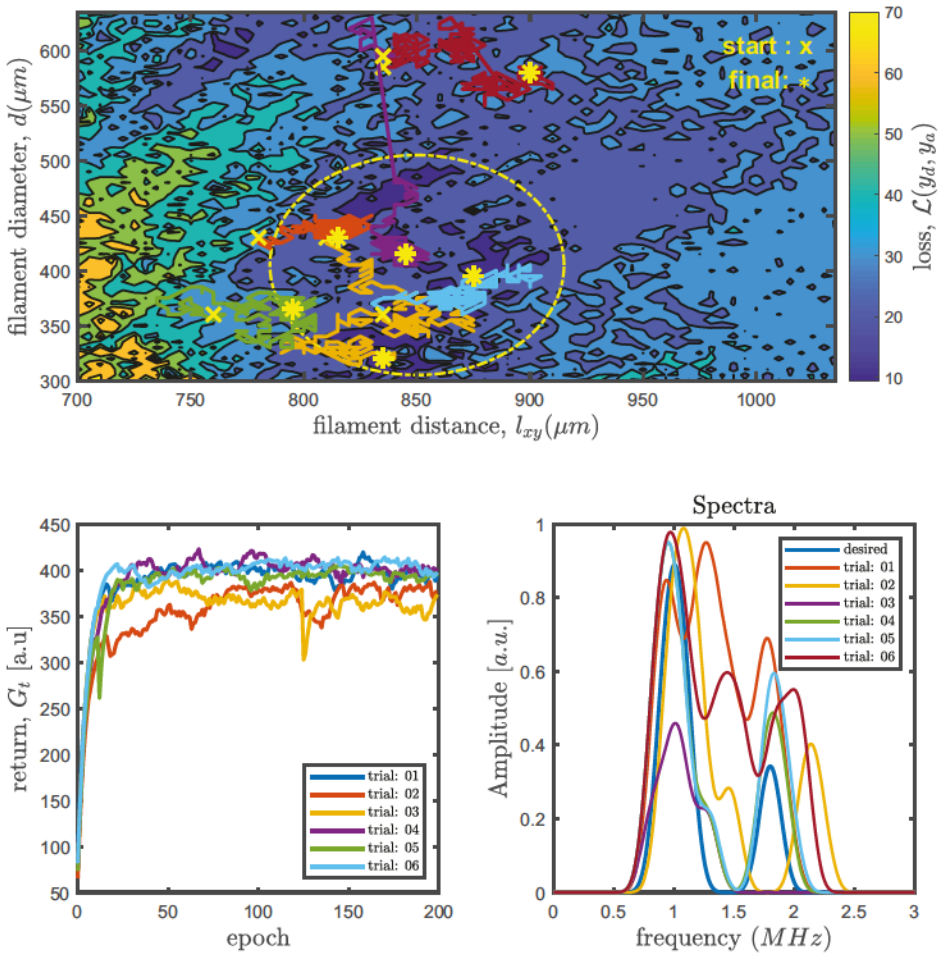

FIGURE 10. Differential SARSA (undiscounted average reward) results: (a) Agent from different random initial states often end up in the desired region within the allocated low budget. Path of the agent has been shown from random start state to final state, (b) return obtained by the agent over the timesteps, (c) A comparison of desired and actual spectra corresponding to agents final position.

CONCLUSION

Here, we study the general concept of manufacturing systems that learn to manufacture parts with a specified part performance. We focused on the cognition element of such a system, investigating a panel of supervised and reinforcement learning algorithms for their ability to optimize for part performance, with the constraint of a manufacturing budget limiting the number of manufacturing runs allowable. Reinforcement learning algorithms were observed to provide good performance even when there is a fixed manufacturing budget on the order of 100 observations. Critically, reinforcement learning algorithms aggressively pursue high-reward parameter sets, effectively ignoring low-reward parameter sets and thus not wasting manufacturing budget in low-reward regions of the parameter space. Future work will investigate multi-agent reinforcement learning algorithms that can coordinate to cull low-reward agents to further reduce the manufacturing budget and application on a manufacturing test bed to test algorithm efficacy with real process and measurement variability.

ACKNOWLEDGMENT

This work has been supported by NSF Award CMMI-1727894. The authors would like to acknowledge the computa-

tional facility support from the Simulation Innovation and Modeling Center at the Ohio State University. The authors would also like to acknowledge Brian Lezzi and Zahra Afkhami from University of Michigan for their suggestions.

REFERENCES

[1] Mariagrazia Dotoli, Alexander Fay, Marek Miśkowicz, and Carla Seatzu. An overview of current technologies and emerging trends in factory automation. *International Journal of Production Research*, 57(15-16):5047–5067, 2019.

[2] M. G. Mehrabi, A. G. Ulsoy, Y. Koren, and P. Heytler. Trends and perspectives in flexible and reconfigurable manufacturing systems. *Journal of Intelligent Manufacturing*, 13(2):135–146, Apr 2002.

[3] Pavel Nikolaev, Daylond Hooper, Frederick Webber, Rahul Rao, Kevin Decker, Michael Krein, Jason Poleski, Rick Barto, and Benji Maruyama. Autonomy in materials research: a case study in carbon nanotube growth. *npj Computational Materials*, 2:16031, October 2016.

[4] Kristofer G Reyes and Benji Maruyama. The machine learning revolution in materials? *MRS Bulletin*, 44(7):530–537, 2019.

[5] Daehn, Glenn, John Allison, Elizabeth Bilitz, David Bourne, Jian Cao, Kester Clarke, Johnnie DeLoach Jr., Ed Herderick, John Lewandowski, Tony Schmitz, Howard Sizek, and A. Erman Tekkaya. Metamorphic Manufacturing: Shaping the Future of On-Demand Components. Technical report, The Minerals, Metals & Materials Society, March 2019.

[6] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[7] Patrick J. Grother. NIST Handprinted Forms and Characters, NIST Special Database 19., 1995. type: dataset.

[8] James Bennett and Stan Lanning. The Netflix prize. In *KDD Cup and Workshop*, San Jose, CA, 2007.

[9] Yurii Nesterov. *Lectures on convex optimization*, volume 137. Springer.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[12] Lauren A Hannah. Stochastic optimization. *International Encyclopedia of the Social & Behavioral Sciences*, 2:473–481, 2015.

[13] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[14] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

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

[16] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[17] Alena Kruisová, Martin Ševčík, Hanuš Seiner, Petr Sedlák, Benito Román-Manso, Pilar Miranzo, Manuel Belmonte, and Michal Landa. Ultrasonic bandgaps in 3d-printed periodic ceramic microlattices. *Ultrasonics*, 82:91–100, 2018.

[18] Manvir S Kushwaha, Peter Halevi, Leonard Dobrzynski, and Bahram Djafari-Rouhani. Acoustic band structure of periodic elastic composites. *Physical review letters*, 71(13):2022, 1993.

[19] Wahyu Setyawan and Stefano Curtarolo. High-throughput electronic band structure calculations: Challenges and tools. *Computational materials science*, 49(2):299–312, 2010.

[20] Franziska Warmuth and Carolin Körner. Phononic band gaps in 2d quadratic and 3d cubic cellular structures. *Materials*, 8(12):8327–8337, 2015.

[21] Xiaochi Xu, Chaitanya Krishna Prasad Vallabh, Zachary James Cleland, and Cetin Cetinkaya. Phononic crystal artifacts for real-time in situ quality monitoring in additive manufacturing. *Journal of Manufacturing Science and Engineering*, 139(9):091001, 2017.

[22] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[23] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. MIT press, 2017.

Appendix A: result from supervised algorithms

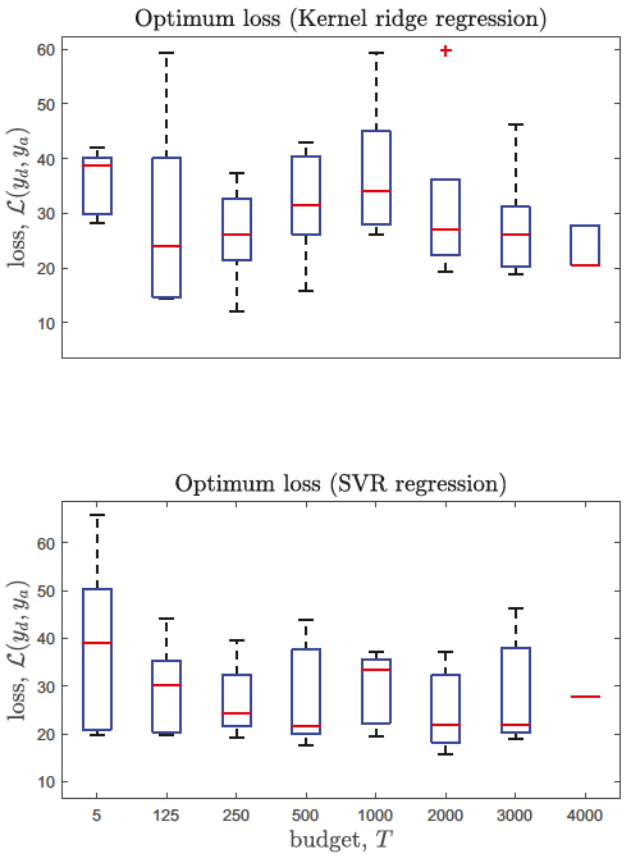

FIGURE 11. Optimum Loss for the remaining supervised algorithms.