Deep Reinforcement Learning for the Design of Structural Topologies

Nathan Brown
nkbrown@clemson.edu

DEEP REINFORCEMENT LEARNING FOR THE DESIGN OF STRUCTURAL TOPOLOGIES

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
Of the Requirements for the Degree
Doctor of Philosophy
Mechanical Engineering

---

by
Nathan K. Brown
December 2023

ABSTRACT

Advances in machine learning algorithms and increased computational efficiencies have given engineers new capabilities and tools for engineering design. The presented work investigates using deep reinforcement learning (DRL), a subset of deep machine learning that teaches an agent to complete a task through accumulating experiences in an interactive environment, to design 2D structural topologies. Three unique structural topology design problems are investigated to validate DRL as a practical design automation tool to produce high-performing designs in structural topology domains.

The first design problem attempts to find a gradient-free alternative to solving the compliance minimization topology optimization problem. In the proposed DRL environment, a DRL agent can sequentially remove elements from a starting solid material domain to form a topology that minimizes compliance. After each action, the agent receives feedback on its performance by evaluating how well the current topology satisfies the design objectives. The agent learned a generalized design strategy that produced topology designs with similar or better compliance minimization performance than traditional gradient-based topology optimization methods given various boundary conditions.

The second design problem reformulates mechanical metamaterial unit cell design as a DRL task. The local unit cells of mechanical metamaterials are built by sequentially adding material elements according to a cubic Bezier curve methodology. The unit cells are built such that, when tessellated, they exhibit a targeted nonlinear deformation response under uniaxial compressive or tensile loading. Using a variational autoencoder for domain dimension reduction and a surrogate model for rapid deformation response prediction, the DRL environment was built to allow the agent to rapidly build mechanical metamaterials that exhibit a diverse array of deformation responses with variable degrees of nonlinearity.

Finally, the third design problem expands on the second to train a DRL agent to design mechanical metamaterials with tailorable deformation and energy manipulation characteristics. The agent’s design performance was validated by creating metamaterials with a thermoplastic polyurethane (TPU) constitutive material that increased or decreased hysteresis while exhibiting the compressive deformation response of expanded thermoplastic polyurethane (E-TPU). These optimized designs were additively manufactured and underwent experimental cyclic compressive testing. The results showed the E-TPU and metamaterial with E-TPU target properties were well aligned, underscoring the feasibility of designing mechanical metamaterials with customizable deformation and energy manipulation responses. Finally, the agent's generalized design capabilities were tested by designing multiple metamaterials with diverse desired loading deformation responses and specific hysteresis objectives. The combined success of these three design problems is critical in proving that a DRL agent can serve as a co-designer working with a human designer to achieve high-performing solutions in the domain of 2D structural topologies and is worthy of incorporation into a wide array of engineering design domains.

DEDICATION

To my parents, whose unwavering love and support have been my guiding light on this incredible journey

ACKNOWLEDGEMENTS

I cannot begin to thank my advisor Dr. Gang Li enough for his time, effort, and support during this process. Without his constant encouragement, enthusiasm, patience, and immense knowledge, I would not have been able to complete this journey. I am so grateful to have an advisor who cares so much about the well-being and success of his students and who allowed me to pursue research that aligned with my passions while also setting me up for future success.

I would also like to send my sincere thanks to Dr. George Fadel and Dr. Anthony Garland for their constant backing and guidance throughout this process, who, week by week, along with Dr. Li, helped develop my love for research and a desire to continue pursuing the answers to challenging problems. I will forever be thankful for the opportunity to work with these incredible researchers and look forward to continuing this relationship in the next stages of my career.

I thank Amit Deshpande, Sai Pradeep, Sam Biemann, and Tim Pruett for their immeasurable assistance in this work. Their enthusiasm towards this research was critical to its success.

To all my professors, students, and friends I've been lucky enough to get to know during my eight-year career at Clemson University, thank you for the support and memories and for making Clemson one of the most special places on Earth. To the MEGSC: Yasir, Chase, Malek, Omar, Daniel, Yitao, Milad, Irina, and Maher, your friendship has been one of the most meaningful parts of my time at Clemson, and for that, I will always be grateful. "No road is long with good company," and with you, this journey flew by.

Finally, to my parents and entire family, thank you for loving and strengthening me through each step of this journey. Without you, none of this would be possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION AND MOTIVATION

The engineering design process is primarily guided by a human designer taking a set of requirements and using past experiences or analytical equations to iteratively improve the detailed representation of the design. Human designers use models of the world and systems to understand how different designs will perform in different situations. In general, a model represents a partial and simplified view of a system. Therefore, creating multiple models is traditionally necessary to capture a more accurate and encompassing understanding of the system of interest [1]. These models include governing equations (PDEs) [2]–[4], state machines [5]–[7], surrogates [8]–[12], and many more. The human engineer generates candidate designs and evaluates the designs from these models, assesses the quality of the design, identifies where it succeeded or failed, and then modifies the design to improve the performance or meet requirements [13]. Essentially, engineering design is 1) specifying the objectives and constraints for a system, 2) identifying methods for evaluating the objectives and constraints, and 3) creatively designing and iterating on the design to better meet the objectives and constraints. A core challenge in the engineering design process is that humans have reduced efficiency in comprehending and acting on data produced from the system models. Additionally, any proposed actions or conclusions drawn by the engineering designer will inherently be biased by human intuition or previous experiences, potentially reducing the quality of the proposed design [14]. These limitations have prompted the development of automated design tools.

Design automation improves upon traditional engineering design approaches by introducing optimization methods to improve the conceptual designs of the human designer [15]. Optimization seeks to find the best solution given a model, a set of parameters, one or more objective(s), and constraints. In general, optimization-based design seeks to mimic the human design process by using computational tools to evaluate a design, using the output of the evaluation to inform how to update candidate design solutions to better meet the objective(s) and satisfy the constraints [16]. Two of the most common computational optimization methods include gradient-based topology optimization (TO) and size/shape optimization. The details of these optimization methods and their uses in applicable engineering design problems will be addressed in the Literature Review section of the manuscript.

Optimizers are not without limitations. Gradient-based optimization tools require an analytical relationship between the design parameters and the objective function(s) and constraint(s) to calculate the gradient(s) needed to determine the path toward an optimal solution. Often, such gradients do not exist for complex systems better described by state machines, flow charts, and physical models. Furthermore, if the objective and constraint functions can be developed, they must be continuous and smooth to increase the chance of convergence. Non-convex relationships between the functions and their corresponding parameters can make solving gradient optimization challenging. These relationships can cause functions to converge to local minima, far from the best design solution [17]. Additionally, nonlinear constraint functions can make finding feasible solutions difficult. To overcome these limitations, heuristic optimization approaches (e.g., particle swarm, genetic algorithms, ant colony) seek to apply general rules that will slowly push the candidate designs toward the global optima. However, these methods generally require many model evaluation calls and perform poorly in high-dimensional domains [18]. Due to the limitations of traditional optimization methods, deep machine learning techniques are becoming more regularly implemented as potential solutions to achieving optimized design solutions in various domains [19].

1.1 Machine Learning and Engineering Design

Integrating deep machine learning (DML) and other artificial intelligence (AI) techniques into the design process offers a valuable relationship between human experts and novel design tools to revolutionize the efficiency and effectiveness of the engineering design process. In deep learning, data is passed through sequential neural network (NN) layers to learn patterns between inputs and outputs. The patterns are established by calculating the difference between the NN output and the true value, called the loss. The loss informs how to change the weights of the NN to better predict the outputs. Within the engineering design domain, a DML model could theoretically approximate the complex mapping between a design and its optimal design modifications without formulating a gradient-based objective function. While DML methods offer an alternative to traditional gradient-based optimizers for engineering design, these methods are traditionally limited to supervised or unsupervised learning. These two subsets of DML rely on generating training datasets ahead of time so that the model can learn underlying trends or patterns to propose a solution [20]. While supervised [19], [21]–[26], and unsupervised learning [27]–[29] models have been successfully implemented as design optimization tools, three key challenges may limit the effectiveness of their implementation in a broader range of design domains: 1) Data sparsity, 2) The creativity gap 3) Usability and feasibility [19].

1. Data Sparsity: Implementing supervised and unsupervised learning models into the engineering design process can be hindered by the limited availability of large, well-annotated, public datasets of engineering design-relevant data. Even if data is available, it often does not cover the design space evenly. Limited diversity in training data can introduce the possibility of overfitting. Overfitting occurs when the DML model memorizes the relationship between specific inputs (design parameters) and outputs (proposed design solution) rather than learning generalized relationships between the

inputs and outputs. An overfit DML model can propose severely suboptimal design solutions when introduced to design parameters it has not experienced during training.

2. The creativity gap: In most DML applications for engineering design, the proposed solutions can closely mimic the training data, which is generally modeled around existing designs and products. Emulating existing designs is often undesirable and can fail to yield novel, improved solutions. Formatting the DML training sets around existing products will introduce an inherent bias from the human designer, potentially limiting the creativity and optimization performance of the DML model. The human designer will view certain features as critical for successful training based on intuition or previous experiences. The bias toward specific datasets may hinder the ability to achieve high-performing solutions.
3. Usability and feasibility: The proposed solutions of a DML model can only be deemed viable if they can be physically fabricated. Design solutions that are not feasible have limited benefit to the human engineer. Therefore, the dataset must be encoded as a data representation that contains enough parametric details to be deemed viable for physical fabrication. However, encoding these representations can be challenging and subject to inherent bias from the human engineer.

The three limitations may arise when implementing supervised or unsupervised DML methods in engineering design. The limitations stem from the models' reliance on generating training datasets beforehand. These datasets can be challenging to produce (data sparsity), inherently biased by the human designer (the creativity gap), and may yield impractical design solutions (usability and feasibility). For these reasons, there is justification for exploring DML for engineering design beyond the limitations of supervised and unsupervised models.

1.2 Reinforcement Learning and Engineering Design

An alternative branch of DML, which does not rely on generating a prefabricated training dataset, is deep reinforcement learning (DRL). Within the DRL paradigm, the goal is to train an agent to learn to complete a particular task or set of tasks within an interactive environment where the agent receives feedback (reward) on its actions based on how well it is performing the tasks [30]. The agent is essentially learning how to solve the optimal control problem for a given task. In recent years, DRL has shown groundbreaking achievements in the fields of game-play [31], [32], protein folding [33], robotic control [34], and nuclear fusion [35], among many others. DRL is also currently being used to aid in engineering design and optimization, including CAD model optimization [36], topology design [37], microchip floor planning [38], 3D shape modeling [39], and composite design [40].

DRL offers a unique advantage over the other DML methods for engineering design automation. Instead of configuring an effective training dataset, a designer can utilize DRL to collect the training data during the learning process by creating a DRL environment that captures the salient information about the design problem and corresponding domain. The DRL agent can now learn to satisfy a given design objective by interacting with the environment and receiving feedback on these interactions. Within the DRL setup, the designer's influence and biases are still present in how the environment is simulated. However, this dependency is significantly reduced. In addition, DRL enables improved domain explorations by quickly identifying suboptimal regions and directing domain exploration away from those regions.

A DRL environment is the "world" with which the agent can interact to take actions, observe states, and collect rewards. The agent's objective is to interact with the environment to learn the best sequence of actions (policy), π, which leads to the largest reward accumulation. The agent and environment interact through the transition of information, as seen in Figure 1-1.

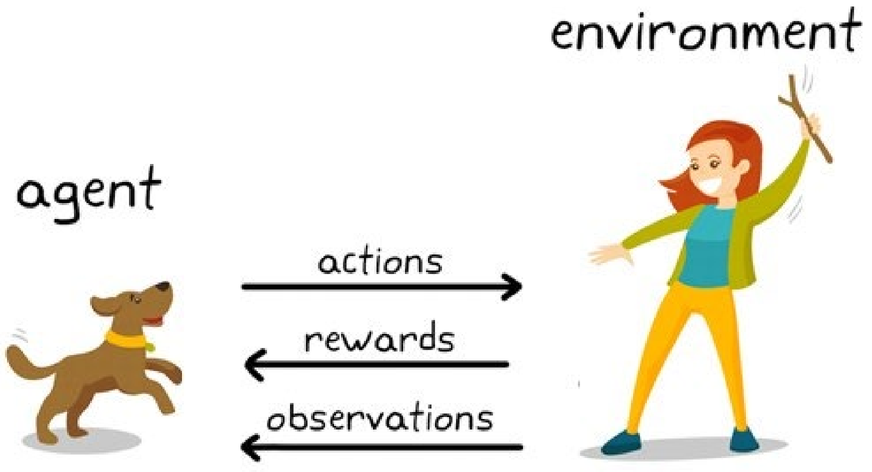


Figure 1-1. Information transfer of generic RL problem

Given a current observation, the agent performs an action in the environment. The environment returns a new observation and a reward that acts as feedback for the agent to determine if the action was good or bad. The agent will update its policy to maximize positive feedback based on this new observation and reward.

Using DRL, the engineering design problem is flipped from identifying optimal design solutions given a set of design inputs to the optimal control problem of finding a sequence of actions to adjust a design to optimize the design objective. This difference allows the agent to be creative and explore regions of the design domain that would not be obvious to a human designer. In addition, the optimal sequence may require passing through several sub-optimal designs, where a gradient optimizer might get stuck in a local minimum before reaching the best design. Finally, this approach allows for generalized design solutions that can rapidly adjust to changes in objectives or constraints.

The mathematical framework that helps define the transfer of information between the agent and a value-based environment is a Markov Decision Process (MDP) [41]. An MDP is a discrete-time stochastic process where state transitions and reward functions are solely dependent on the current state and action and are independent of the previous states and actions. An MDP is built upon four elements, state-space (S), action space (A), transition probability function (P), and reward function (R). The state-space of an environment represents all the combinations of observations an agent can experience while interacting with the environment. Depending on the complexity of the environment, the state-space can range drastically, potentially varying from a 4x4 grid for simple maze-solving applications [42] to the continuously changing urban roads and landscapes needed for autonomous driving [43]. The action space defines all the possible actions the agent can take at each time step to move from the current observation to the resulting next observation. These action spaces can be either discrete, such as moving the agent "left" or "right" in a video game, or continuous, such as the angular displacement of a steering wheel [44].

Formulating the reward function of an MPD can be difficult due to human subjectivity [45], characterization of complex goals [46], sparse rewards [47], and conflicting objectives [37]. The core challenge is to shape the reward function to encourage the agent to effectively and efficiently achieve the desired outcome. The reward function controls the feedback given to an agent after taking an action. Therefore, the agent seeks to maximize its reward and should learn what action sequences must be taken to accumulate the most reward while avoiding penalties. The reward function can be produced using nearly any combination of engineering analysis tools, such as PDE solvers, state machines, analytical equations, and even models based on real-world data.

The final component of an MDP is the transition probability function, also called the value function. The value function acts as the "brain" of the agent by determining the anticipated value, $Q(s_t, a_t)$, of taking an action, a_t, given the observation/state, s_t, at the current time step, t. The value can be thought of as *how valuable it is to perform a given action in a given state*. The notion of "how valuable" is in terms of expected future rewards. The future rewards are dependent on the sequence of future actions the agent will take, and this sequence of actions is known as the policy,

π. The overall objective of DRL is to learn the policy that will lead to the optimal action-value function, denoted Q_*, and defined as

$$Q_*(s, a) = \max_{\pi} Q_\pi(s, a) \tag{1}$$

for all states, s, within the state-space S and all actions, a, within the action-space A. For all state-action pairs (s, a), Equation 1 gives the expected return for taking action a_t in state s_t and, therefore, is dependent on the action reward, r_t, and the anticipated value of the future states $Q_*(s_{t+1})$, and can be rewritten as:

$$Q_*(s_t, a_t) = \mathbb{E}[r_t + \gamma Q_*(s_{t+1})] \tag{2}$$

where γ is a future value discount factor between zero and one. Equation 2 offers a generic representation of the Bellman optimality value function [41], and several extensions will be presented throughout this manuscript.

If $Q_*(s_t, a_t)$ is known for every state-action pair, the agent can easily execute the sequence of actions leading to the largest accumulation of rewards. However, the environments of most DRL problems have action and state spaces that are so large and complex that knowing the exact value of $Q_*(s_t, a_t)$ for each state-action pair is impossible. Therefore, $Q_*(s_t, a_t)$ is predicted using a deep neural network (DNN). A numerical representation of the agent's current state, s_t, is used as the input for the DNN, and the output is the predicted $Q_*(s_t, a_t)$ for each action, a_t within the action space.

To successfully implement a DRL agent into an engineering design task, these four components (S, A, R, P) of an MDP must be appropriately defined. Instead of producing datasets to train a supervised or unsupervised model to complete design tasks, the human engineer is now tasked with configuring these four elements to capture critical information about the design task. Properly defining these four components can create a rich environment for the agent to interact with. According to the *Reward is Enough* hypothesis, the ability of a DRL agent to learn to complete a task can be understood as learning to maximize reward within a rich environment by trial and error [48]. This hypothesis can be applied to teaching DRL agents to complete engineering designs tasks. The DRL environment must capture critical information about the design problem, including a reward function such that maximizing the reward leads to high-performing design solutions. The DRL agent would then attempt to learn the sequence of actions that maximize the reward within this environment. This sequence of actions would mimic an engineering design process that leads to high-performing design solutions. Altering the DRL environment to capture information about various design domains would allow the DRL agent to be a powerful design tool in arbitrarily complex domains.

This work attempts to take a step toward validating the concept that a DRL agent can learn powerful design strategies when interacting with an environment built to represent a particular design domain. While I agree with the hypothesis that DRL models can achieve high-performance design solutions in a wide range of scenarios, this work focuses on validating this hypothesis in the domain of 2D structural topology design, particularly compliance-minimization topology optimization and novel mechanical metamaterial unit cell design with targeted mechanical response characteristics. For the remainder of this manuscript, "structural topologies" can be defined as the layout of the solid and empty space within a continuous structure or subsystem. With this background information provided, the formal objective of this work can be presented as the following:

Demonstrate that a deep reinforcement learning agent can serve as a co-designer working with an engineering designer to achieve high-performing design solutions in various structural topology design domains.

The "co-designer" relationship between the DRL agent and human designer is established by the human designer defining the design objective(s) and constraint(s) and constructing the DRL environment, while the DRL agent uses this information to attempt to determine the set of actions to achieve a high performing design solution that satisfies the objective and constraints.

1.3 Research Questions and Hypotheses

This work's objective can be completed by answering the three research questions below. Each question attempts to validate that a DRL agent can successfully produce high-performing engineering designs in a series of structural topology-related design problems without the need to calculate objective or constraint gradients. The design domains will specifically target compliance-minimization topology optimization, mechanical metamaterial unit cell design with targeted nonlinear deformation response, and mechanical metamaterial design with targeted deformation and energy manipulation characteristics. The literature review has included a thorough investigation of the current design methods for these domains. The complexity of the design problem is expected to increase moving through the list of research questions. Each research question has been presented with a corresponding description of the design domain and a general hypothesis for how a DRL agent can successfully achieve this design task. While these research questions and associated solutions can help demonstrate a DRL agent as an effective co-designer within structural topology design problems, this work does not claim that DRL is the only or the optimal method for producing these designs.

1.3.1 Research Question 1

How can deep reinforcement learning be used to produce high-performance solutions for the 2D compliance minimization topology optimization problem?

The first question proposes a gradient-free alternative to compliance minimization topology optimization using DRL. Topology optimization (TO) is a common engineering-based design tool and attempts to optimally place material within a domain to minimize an objective while satisfying constraint(s) based on stress and weight [49]. For pseudo-density-based approaches, TO works by calling a finite element analysis of a structure, calculating the gradient of the objective function with respect to the design variables (the pseudo densities), and updating the design variables until convergence, as in the Solid Isotropic Material with Penalization method (SIMP).

While gradient-based TO methods are well-developed and highly efficient, the sensitivity analyses of the objective functions and constraints can be challenging, especially with the introduction of nonlinearity. In addition, they can be sensitive to initial conditions, which results in the optimizer converging to different designs under changing initial conditions. Genetic algorithms have also been used as a gradient-free approach to solving TO problems [50]–[52]. However, these methods are limited by significant increases in computational cost with larger domains, inefficient domain exploration, and the inability to generalize to new constraints and objectives without retraining. Therefore, this work focuses on gradient-free alternatives, specifically using DRL to solve the compliance minimization TO problem.

Various DML methods, primarily relying on supervised and unsupervised learning algorithms, have been developed to address this problem and will be discussed in the literature review. However, these methods can be inherently limited by the need for constructing topology datasets beforehand to effectively train the ML models. These datasets can be difficult to produce or be inherently biased by the human who developed them. Therefore, I hypothesize that deep reinforcement learning will be shown as a gradient-free compliance minimization TO solver that does not rely on the prefabrication of topology datasets.

To investigate the idea of using DRL in this design domain, the four components of the previously discussed MDP will need to be defined. The four components of the MDP must capture critical information regarding the compliance minimization TO problem to ensure the DRL agent can learn the best sequence of actions to achieve the optimal topology designs. The DRL environment can be configured in several ways, and this dissertation does not claim that the selected methods offer an optimal approach but instead present one or many potential solutions for applying DRL to solve compliance minimization TO. The proposed method will focus on discretized topologies where the domain is viewed as a combination of individual elements. Each element can be viewed as either material or voided. Therefore, each element is treated as a binary design variable, and the complexity of the design problem is dependent on the number of elements used to define the topology shape.

The compliance minimization TO problem is commonly addressed by eliminating minimally load-bearing elements as they offer little structural rigidity to the base topology. These minimally load-bearing elements can be found by determining the stress distribution of the topology. Therefore, the state space of the DRL environment should capture the topology's stress distribution and boundary conditions, commonly used to solve gradient-based compliance minimization TO problems. Given the discretized representation of the topologies, the stress and boundary conditions should be defined at the elemental level. The action space will be simplified as a discrete space allowing the agent to toggle any element variable from "material" to "void." Limiting the agent to only voiding a single element per action step may hinder the efficiency of the DRL agent, but it can also reduce the complexity of the problem. The reward function should mimic the objective of the compliance minimization problem. Therefore, the reward function should assign the agent larger rewards for designing topologies with smaller volume fractions and compliance. Finally, the transition probability function needs to provide a method for the DRL agent to know which element it should remove, given the current observation. The agent's actions should depend on both the individual observation values of each component and the relationship between these values throughout the topology. Since spatial awareness is expected to be necessary, a convolutional neural network, a class of artificial neural networks commonly applied to analyze visual imagery [20], should be used to determine the best action for each observation.

DRL algorithms that define transition probability functions can be divided into value-based or policy-based functions. Value-based functions calculate the predicted value of taking each action given the current observation. Alternatively, policy-based functions calculate the probability of taking each action given a current observation. Policy-based methods are better suited for DRL environments with continuous action spaces and thrive with small policy changes that can result in repeated actions given similar observations. On the other hand, value-based functions are better suited for large policy changes. The proposed DRL environment has a discrete action space where voiding a single element can result in drastic policy changes. For example, the value associated with voiding a particular element can significantly decrease as it is altered from a minimally load-bearing material element to a voided element. I hypothesize that a value-based function can capture this change in value and policy functions better than a policy-based method. The value-based double deep Q-learning algorithm will be utilized for this reason and its success with similar design problems [50].

With the components of the MPD and corresponding DRL algorithm produced, it is hypothesized that a DRL agent will be able to design high-performance 2D topologies based on the compliance minimization TO objective without the need to calculate objective gradients or rely on prefabricated topology datasets.

1.3.2 Research Question 2

How can deep reinforcement learning design novel mechanical metamaterial unit cells with targeted nonlinear deformation characteristics under uniaxial tension or compression?

The second research question targets the more complex design domain of mechanical metamaterials. Mechanical metamaterials are artificial materials with unique global properties due to the structural geometry and material composition of their unit cell [53]. These unique properties include mechanical metamaterials with negative Poisson's ratio [54] or energy absorption through bi-stability [55], thermal metamaterials with extreme thermal expansion [56], and fluidic metamaterials exhibiting maximum permeability and prescribed flow symmetries in bulk materials [57]. While metamaterials can achieve unique properties, the second research question specifically targets designing mechanical metamaterials that exhibit targeted deformation behaviors. This research attempts to design mechanical metamaterials with linear elastic constitutive properties that mimic the deformation behavior of nonlinear materials such as elastomers or foams. Since metamaterials are made up of unit cells, the overall response of the metamaterial is a function of the base constitutive material and the geometries of the unit cells. The objective is to achieve bulk non-linear behavior in the metamaterial by carefully adding geometric nonlinearities to the unit cell. The geometric nonlinearities are associated with the unit cell pattern and result in large overall deformation with small linear strains within the constitutive material. The most popular methods for designing metamaterials with targeted deformation responses are topology optimization (TO), size/shape optimization, and synthesis methods. These three methods will be investigated in the literature review.

TO for metamaterial design is similar to the compliance minimization TO but the objective is changed to determine the material densities that minimize the error between the desired and resulting deformation responses. Gradient-based TO methods have achieved impressive results when designing mechanical metamaterials, which will be discussed in the literature review. However, gradient-based TO methods are not without flaws. The sensitivity analyses of the objective functions and constraints can be challenging, especially with the introduction of nonlinearity. Additionally, the relationship between design variables and objective/constraint functions must be differentiable and mathematically described so that the gradient can be calculated on all design variables before an update of the design variables in the negative gradient direction is performed. Due to the challenges of gradient-based TO, researchers have implemented size/shape optimization and unit cell synthesis as alternative methods for metamaterial unit cell design.

Size/shape optimization alters the geometric dimensions of an initially parameterized design to optimize the design objective(s) while satisfying constraint(s). This technique relies on analytical tools such as partial differential equation solvers [58], finite element analysis [59], analytical equations [60], and models based on real-world data [61] to determine the performance of a design, compare that performance to the objective, and change the geometric parameters to optimize the objective. The major limitation of size/shape optimization is the reliance on a human user designing and parameterizing the initial unit cell. This bias can limit the explorable design domain.

A unit cell synthesis approach is a process of iteratively selecting and placing elemental components in a unit cell and then carrying out tessellations of the unit cell into a metamaterial. The unit cells are designed at the local level but, when tessellated, result in a metamaterial with material properties unattainable by the constitutive material alone. Unfortunately, these methods commonly rely on a human designer to propose initial designs which can be optimized using size or shape optimization. The reliance on a starting design and corresponding parameters can limit the exploration of the design space, leading to sub-optimal solutions. Additionally, the human-designer-based synthesis method offers limited automation and generalization capabilities.

Due to the limitations of traditional metamaterial design approaches, I hypothesize that this design task can be reformulated as an MDP, allowing a DRL agent to design mechanical metamaterial unit cells with targeted nonlinear deformation characteristics. This hypothesis is built on the fact that unit cell design approaches are iterative processes that optimize local material configuration to deliver a particular material response. DRL agents exhibit this same sequential building process and configure a combination of components by accumulating rewards in an interactive environment. The DRL method will also ensure the thorough and efficient exploration of the complex metamaterial design environment.

Similar to RQ1, the four components of the MDP must be selected to capture relevant information about this new design domain. Properly defining the four components of the MDP will allow a DRL agent to design unit cells that, when tessellated, exhibit targeted nonlinear deformation characteristics under uniaxial tension or compression.

The DRL unit cell design process should sequentially alter the state of elements within the topology to achieve these characteristics. Similar to RQ1, the design domain is viewed as discrete grids where each element is viewed as either arbitrarily defined as “material” or "void." To improve upon the methods of RQ1, the material alteration process was changed to sequentially adding materials to a blank starting discrete domain compared to removing/voiding elements from a starting solid-block topology. The material elements were added according to paths defined by cubic Bezier curves. These curves introduce novel configurations of material elements that individually exhibit geometric nonlinearities. These material element configurations ensure a diverse array of metamaterial designs, each with a unique deformation response. I do not claim that the cubic Bezier curve approach is the only solution to this design problem or necessarily optimal. However, the findings shown in this work validate the effectiveness of this method.

Due to the anticipated complexity of the proposed unit cell designs, a dimension reduction tool, such as a variational autoencoder or principal component analysis, should produce a low-dimensional representation of the domain. This 1D low-dimensional representation can be used in place of the 2D unit cell topology to represent the design configuration in the DRL environment.

An additional improvement on the methods of RQ1 is introducing a pre-trained surrogate model to predict the proposed unit cells' responses. This method is considerably more efficient than relying on individual *Abaqus/CAE* (Dassault Systemes, Velizy-Vallacoubly, France) FEA calls during each step of the DRL training process. I hypothesize that these improvements should allow the four components of the MDP to capture critical design domain-relevant information.

The state space must capture the current unit cell design and the desired non-linear deformation response. Therefore, the state-space observations combine the 1D VAE latent space representation of the 2D Boolean array of material/void elements in the current unit cell topology and a 1D array of user-specified force values defining a force-displacement curve. The action space used to design these unit cells defines the coordinates of the starting, ending, and two intermediate control points of the cubic Bezier curve and the thickness of the line. This action space is continuous to reduce the high dimensionality that may arise if a discrete action space were to be implemented.

The reward function should evaluate how close the current deformation response of the unit cell is to the desired unit cell. The DRL agent will receive more reward for actions leading to proposed unit cells that closely match the desired deformation response. Finally, the transition probability function needs to give the DRL agent a way to design the high-performing unit cells given the desired loading response. Due to the required continuous state and action space, the deep deterministic policy gradient (DDPG) algorithm will allow the agent to properly explore this complex design environment and learn a generalized design strategy that leads to high-performing designs.

With the components of the MDP and corresponding DRL algorithm produced, it is hypothesized that a DRL agent will be able to design 2D metamaterial unit cells that exhibit targeted nonlinear deformation characteristics under uniaxial tension or compression.

1.3.3 Research Question 3

How can deep reinforcement learning design mechanical metamaterial with tailorable deformation and energy manipulation characteristics under dynamic compressive loading?

Research question three expands on the previous research question by addressing the additional objective of energy manipulation via hysteresis control. Mechanical hysteresis, often called elastic hysteresis, is the energy loss during mechanical loading and unloading brought on by internal material friction. During mechanical testing, the loading and unloading paths of a material exhibiting hysteresis will diverge. The area between these curves represents the energy loss.

Traditionally, engineers use monolithic viscoelastic materials, like elastomeric rubber, for energy absorption applications. However, viscoelastic material performance is dependent on the operating environment. Extreme conditions can cause material degradation in these elastomers, affecting the material performance [62]. For energy-restoring applications, engineers use linear elastic materials (ceramics, metals) or closed foam cells (engineered thermoplastic polyurethanes). The high stiffness and rigidity of linear elastic materials can make them impractical for high-compliance applications such as soft robotics or wearable technologies. Alternatively, the more compliant closed foam cells can be costly and challenging to manufacture [63].

Mechanical metamaterials offer an alternative for mechanical energy manipulation beyond the range of traditional monolithic materials. Through careful design of their unit cell and proper selection of a constitutive material, mechanical metamaterials can achieve unprecedented control over mechanical energy propagation, transformation, and absorption. The previously mentioned TO and size/shape optimizations have been implemented to design metamaterial with unique deformation and energy characteristics. However, they suffer from the limitations discussed in RQ2, specifically the need for a differentiable relationship between the objective and design constraints, reliance on the initial human-proposed design, and lack of generalizability.

Therefore, RQ3 expands on the work of RQ2 by training a DRL agent to design mechanical metamaterials with tailorable deformation and energy manipulation characteristics. Due to the similarities between RQ2 and RQ3, I hypothesize that some of the fundamental components of the DRL approach (state-space, action-space, transition probability function) that were used to satisfy RQ2 will require slight alterations to yield successful results for RQ3.

Compared to the arbitrary design domain of RQ2, RQ3 will focus on a physically validated design domain of metamaterials built with the constitutive material of thermoplastic polyurethane (TPU). TPU is a thermoplastic elastomer that offers the mechanical performance characteristics of rubber (high resilience, compliance, abrasion resistance, and flexibility) but can be processed like thermoplastics through additive manufacturing [64]. Like RQ2, the agent can be prompted to iteratively build metamaterials exhibiting a desired deformation response. However, an additional objective is introduced, prompting the agent to achieve designs that minimize or maximize hysteretic energy loss under cyclic compressive loading. Therefore, the reward function from RQ2 must be altered to account for strategic energy manipulation.

To provide real-world applicability, the agent was prompted to design TPU metamaterials that maximize or minimize hysteresis while exhibiting the deformation response of the expanded thermoplastic polyurethane (E-TPU) material, Infinergy® (BASF, Ludwigshafen, Germany). This design task physically validates mechanical metamaterials as a potential material solution in various industries, including footwear, wearable technologies, and medical equipment. Proving applicability is essential, but a critical advantage of DRL methods is their generalizability. Therefore, the agent should be tested on a wide array of desired deformation and energy manipulation responses.

I believe that successfully answering these three research questions and, in turn, establishing three DRL environments for which an agent can produce high-performing design solutions for these unique topology design problems will satisfy the objective of proving DRL to be a high-level co-designing tool that can address trade-offs and produce superior solutions in an array of structural topology domains.

1.4 Manuscript Organization

The remaining chapters of this manuscript are presented accordingly: Literature Review, Framework and Implementation, General Framework for Deep Reinforcement Learning in Engineering Design, and Conclusion. In the *Literature Review*, I begin by discussing how DRL has been applied to achieve superior design and optimization results in a series of design domains. Each presented study in this manuscript offers a unique advantage for applying DRL to the engineering design process. Next, the *Literature Review* discusses the methods currently implemented to solve design and optimization problems in topology optimization and metamaterial design. The advantages and limitations of each method are presented to justify the investigation of DRL as an engineering design alternative.

Separate *Framework and Implementation* chapters are devoted to solving each research question. Within these chapters, the individual engineering design problem is introduced, the DRL environment is established, and the agent is tested on achieving high-performing solutions. Regarding the first research question, I discuss the method for reformulating the 2D compliance minimization TO problem as an MDP. Within this chapter, I break down the details of the four components of the MDP and justify my selected DRL algorithm. The DRL agent training and testing results are also presented, including a comparison to a gradient-based TO solver. Finally, I conclude this chapter with specific shortcomings of the DRL method.

In the next chapter, I discuss how a DRL environment was built to design mechanical metamaterials with customizable nonlinear deformation responses. I discuss the four MPD components and provide details on the additional DML tools (VAE and surrogate model) that were used to improve the performance and efficiency of the DRL method. The section thoroughly validates the agent's performance against a diverse array of desired nonlinear deformation responses. The chapter concludes with design domain limitations, several of which are addressed with RQ3.

The final *Framework and Implementation* chapter investigates the DRL method for designing mechanical metamaterials with tailorable deformation and energy characteristics. There is considerable overlap between RQ2 and RQ3. Therefore, the reader will regularly be directed to the RQ2 chapter for detail and justification for how the DRL environment was built. However, due to the additional objective of energy manipulation control, critical changes to the DRL environment are discussed. Much of the chapter discusses the methods and results of physically manufacturing and experimentally testing metamaterial samples. From these results, a discussion is born regarding the applicability of additively manufactured metamaterials in domains such as footwear, wearable technologies, and medical devices. The chapter ends with an analysis of the generalized design strategy of the DRL agent while simultaneously addressing certain limitations in the material design domain and DRL algorithm.

With the three research questions answered, I offer a general framework for how various engineering problems can be reformulated as deep reinforcement learning problems. Specifically, I offer recommendations for how the design domain, state-space, action-space, reward function, and value function should be established depending on the objectives and constraints of the design problem. This manuscript will conclude with a summary of the completed work and confirmation that all research questions have been answered.

CHAPTER TWO

LITERATURE REVIEW

2.1 Reinforcement Learning and Engineering Design

The initial section of this literature review will present the work completed to incorporate deep reinforcement learning (DRL) into the engineering design process. This review will show the unique advantages of DRL for achieving high-performing design solutions in several domains. This review will also show the novelty of our research and the evident gap in applying DRL to the design of structural topologies.

DRL is currently used to aid in design and optimization for various design domains. Yonekura and Hattori [65] offered a simple framework for incorporating DRL into the design optimization process by determining the optimal airfoil angle of attack, given contour images of the flow field around the airfoil. The DRL agent sequentially alters the angle using a discrete action space to minimize an objective function. The authors test their methods under various airfoil parameters and show that the DRL agent can generalize to unforeseen circumstances.

Settaluri *et al.* [66] incorporated DRL to achieve superior design solutions in the highly complex domain of analog circuit design. Traditional circuit design approaches can be limited due to information gaps, sample inefficiencies, or a lack of generalizability. The authors successfully trained a DRL agent to select optimized circuit parameters given target design specifications. The DRL agent produces diverse results around the feasible domain, showing a similar intuitive understanding as a circuit designer. The results show that the DRL agent could satisfy the target specifications for 96.3% of the test designs. The proposed method was approximately 40x faster than a genetic algorithm alternative that could achieve similar results. This paper shows that DRL agents can develop effective and efficient design strategies that mimic the human design process while significantly reducing time constraints.

Lin *et al.* [39] show another example of DRL agents mimicking human designers by sequentially altering the size and orientation of 3D shapes to achieve a given design. The authors developed a DRL model that created a mesh model of arbitrary objects in two steps: 1) approximating the object's shape using a set of primitives and 2) editing the meshes of the primitives to create a detailed geometry. This approach mimics the basic design sequence of a human designer. By taking actions and collecting rewards in the interactive environment, the agent first learned to parse a target shape into primitives and then edit to satisfy the detailed geometry. More reward was assigned as the DRL agent altered the design to mimic the desired shape better. The DRL agent was trained using a combination of imitation and reinforcement learning. The DRL agent applied the two-step design process to model 3D shapes such as planes, guitars, and automobiles. The results show that a DRL agent can exhibit human-like design sequences in a controlled environment.

Liu *et al.* [67] offer an investigation into incorporating DRL into the design of metamaterials, specifically thermal metamaterials. The authors attempt to achieve thermal transparency using metamaterials with periodic lattices. This paper reframes this common inverse design problem into a DRL problem. The DRL agent is trained to sequentially alter design parameters to realize thermal transparency. The DRL framework relies on FEA simulations of the heat fluxes on the metamaterials as the state spaces. When tested, the DRL agent achieved thermal transparency better and more efficiently than the author's previously proposed autoencoder-based approach. This paper showed that DRL could solve inverse design problems for metamaterials.

Rajak *et al.* [68] showed an example of incorporating DRL into the mechanical metamaterial domain through the design of 2D kirigami. The authors trained a model to design kirigami structures based on a desired stress-strain response. Incorporating other ML solutions to solve this problem would be difficult because the gradient of the model could not be determined due to the discrete nature of the search space. Furthermore, a genetic algorithm would be infeasible due to the computational expense and domain diversity. The authors of this paper trained a DRL agent to design 2D materials by selecting the length and location of seven cuts to achieve a manually specified stress-strain curve under uniaxial tension. During training, the agent experienced approximately 1.45% of possible designs in the state space but could generalize to various desired responses. The authors show that DRL should be used to design a wide range of metamaterials to achieve targeted stress-strain responses when black-box function optimization may be infeasible.

The presented works show that a DRL environment can be configured to solve various design and optimization problems. However, beyond [68], the presented domains and objectives do not align with those presented in the research questions of this proposal. The topology optimization and mechanical metamaterial design domains have not been thoroughly explored using DRL techniques. This apparent gap in the literature highlights the novelty of this research. The remaining sections of the literature review will offer a more in-depth breakdown of the topology optimization and metamaterial design problems, the traditional approaches for solving them, and novel techniques to solve them more efficiently or effectively.

2.2 Topology Optimization Design Domain

2.2.1 Gradient-Based Topology Optimization

Topology optimization (TO) is a common engineering optimization-based design tool traditionally solved using gradient-based methods. TO attempts to optimally place material within a domain to minimize an objective while satisfying constraints [49]. TO has been used extensively and has undergone tremendous development since it was first proposed by Bendsøe and Kikuchi

[69] in 1988. For pseudo-density-based approaches, TO works by calling an FEA of a structure, calculating the gradient of the objective function with respect to the design variables, and updating the design variables, as in the previously mentioned SIMP method [70]. This process is repeated until convergence. In the level-set approach method of TO, the optimizer uses FEA results to implicitly find the gradient of moving the object's material outer boundary with respect to the objective function and then updates the boundary according to the new topology. This process is also repeated until the topology converges. The reader is directed to [49], [69], [70] for a more in-depth discussion of traditional TO methods and underlying functions. While gradient-based TO methods are well developed, the sensitivity analyses of the objective functions and constraints can be challenging, especially with the introduction of nonlinearity. Additionally, the nonconvexity of most TO problems can make efficient convergence difficult [26]. Finally, TO is not generalizable, meaning that for each new objective or constraint, the time-consuming optimization process must be rerun. Therefore, extensive research has been conducted to implement ML methods as an alternative solution for designing optimal topologies.

2.2.2 Incorporating Machine Learning into Topology Design

Various methods have been proposed for combining topology optimization and ML, particularly DML. Researchers have shown that a DML approach could learn the mapping between topology optimization inputs, such as boundary conditions, and the optimized topologies. The computational burden of generating the data and the training process is non-trivial, but after proper training, the model can provide a gradient-free, on-demand topology design tool [26], [71]–[74].

Ulu *et al.* [73] explored the feasibility and performance of a data-driven approach for TO. Ulu trained an ML model using a set of optimal topology examples in a lower-dimensional space. The model successfully learned a mapping between load conditions and optimal topologies. The authors proposed a sequential design approach where the topologies predicted by the ML model could serve as adequate initial conditions for subsequent gradient-based TO methods. The authors prove that incorporating ML in a sequential design approach can more efficiently produce optimized designs.

Sosnovik and Oseledets [74] used deep Convolutional Neural Networks (CNNs), an NN commonly used to analyze images, to accelerate optimal topology design. The proposed method passed the pixel-wise images of partially formed topologies from the SIMP method into the CNN, which outputs the pixel-wise optimal proposed topology. The mapping between partially formed and optimal topologies allowed the SIMP method to produce partial topologies instead of fully converging to the optima of the design domain, significantly decreasing computational cost. These results showed that a deep neural network constructed entirely of convolutional filter and pooling layers could learn fundamental topology design patterns.

Lei *et al.* [23] used a moving morphable component (MMC)-based explicit framework for training dataset generation, leading to a near-instantaneous 2D topology design process. The MMC approach uses morphable components as the basic building blocks of a topology, reducing the dimensionality of the design domain. Compared to other works, Lei's method can rapidly predict the optimal topology once the objective and constraint functions are defined. Additionally, the authors hypothesize that this method can be incorporated into the learning process of engineering designers to help establish general engineering design intuitions for optimal structures under different loading cases.

The above works show a variety of methods for integrating DML into the engineering design process. The proposed methods present alternatives to traditional gradient-based TO solvers and do not require a direct link between design parameters and an objective function to design optimal topologies. The data sets used to train these models are comprised of pre-optimized topologies created from gradient-based TO solvers. While the results from the ML methods using pre-optimized datasets are promising, there are drawbacks.

Producing a large set of optimized topologies under different loading conditions is time-consuming and introduces the possibility of model overfitting if the diversity of the training data is low. Overfitting occurs when the ML model memorizes the relationship between specific inputs and outputs rather than learning a generalized relationship between inputs (the loading conditions) and outputs (the optimal topology). An overfit ML model may generate suboptimal outputs when a loading case not used for training is introduced. Developing these data sets can also require domain expertise from a human designer to extract critical features from the data to serve as inputs to help guide the learning process [75]. For these reasons, there is justification for exploring ML-based topology design beyond the limitations of supervised and unsupervised ML.

To the best of the authors' knowledge, there has only been one attempt to use a DRL agent to sequentially design novel optimal topologies. Hayashi and Ohsaki [51] incorporated DRL and graph embedding for binary truss topology optimization. The binary design of the truss structure views each segment as material or void, with only the material segments defining the structure's topology. The DRL agent was trained to remove individual truss segments sequentially from a starting truss structure until the structure's topology was no longer a single continuous structure. The reward function aligns with the objective of compliance minimization TO by rewarding the agent for only keeping a minimum number of highly-load bearing segments to define the structure. During training, the DRL agent experienced relatively few loading case environments, but during testing, the DRL agent generalized the design strategies and found optimal truss topologies for a wide range of load cases. These results demonstrate DRL's generalization capabilities when applied to truss-based design. However, a significant limitation in these results is the truss-based discretization of the design domain. The design complexity of an optimized truss topology is severely limited compared to an elemental discretization of a design domain, where each element can serve as an independent design variable.

The work completed answering RQ1 attempts to bridge this gap by proposing a DRL-based topology design method that performs sequential interactions with a 2D discrete grid design domain. The design variables of this discretized design domain are viewed as discrete quadrilateral elements, each defined by four nodes within the design domain. The elements and their corresponding nodes are mapped directly to the elements and nodes found in an FEA solver. While there are several ways to discretize a design domain, I believe the simplicity of the 1:1 mapping between parameter elements and FEA elements makes it attractive for our environment setup. For brevity throughout this manuscript, the term "elements" will describe the design variables that may be augmented to best satisfy the design objective. A DRL agent sequentially removes these elements under a load condition to satisfy a multi-objective reward function. The reward function is built to mimic the objectives of compliance minimization TO problems. The details of this work can be found in Chapter 3.

2.3 Metamaterial Unit Cell Design

The next phase of the literature review investigates a small subset of the metamaterial design and optimization process. Metamaterials are a class of artificial materials named due to their designed purpose of achieving specific global properties different from those of their constitutive material. The unique properties arise from strategically designing the base unit cell's structural geometry and material composition. Several researchers have used novel methods to design and optimize metamaterial unit cells to achieve unique material properties.

Steckiewicz and Choroszucho [76] used particle swarm optimization (PSO) to synthesize an electric metamaterial cloak. The internal geometry of the porous unit cells was adjusted using four sizing parameters to minimize the cloak's conductivity. The conductivity was sequentially calculated using FEA after each parameter alteration, significantly increasing the computational cost. The authors show that the metamaterial synthesis method using PSO yields effective electric cloak realization. This paper offers a simplified approach to unit cell design, as the design space is limited to four bounded geometric parameters. Therefore, the final proposed solutions are inherently biased by initial design and parametrization. The author states that future work should focus on analyzing different unit cell geometries, but the design space exploration will always be limited so long as a parameterization approach is used.

Wang *et al.* [77] increase the available design domain complexity by introducing a genetic algorithm (GA). The authors attempt to use this well-known optimization method to develop high-absorption, wideband, and multi-band absorbers using metamaterials. The fitness function of the GA was given according to the relative frequency width, relative average reflectivity magnitude, and amplitude of the reflectivity curve. The metamaterial design space was defined by the unit cell size, the thickness of the substrate, and the geometry of the material components. The GA approach fulfills various structural design requirements, and the test results prove the GA to be a viable tool for metamaterial design. The elemental discretization of the design domain improves the complexity and diversity of the proposed solutions. Regardless of the improved design space exploration, the GA approach is significantly more computationally expensive and still lacks generalization capabilities without retraining.

Nanda *et al.* [76] used DML for inverse metamaterial design. The method took an input of a parameterized material response and output the unit cell design or parameters needed to exhibit the desired response. The model was trained to output a single radius value for the inner ring of a split-ring resonator, given an input of the desired permeability. This method produces a robust sample of permeability responses, ensuring the DML model can learn the underlying inverse relationship between the output radius and material response. The author vocalizes their concerns due to the failure of convergence and the inability to estimate multiple variables. Limiting unit cell design to a single variable reduces the practicality of the proposed design space. Although the authors did not yield viable results using the DML method, several other studies have investigated data-driven and AI approaches for designing metamaterials [78]–[81]

The previously discussed studies offered unique approaches to designing metamaterial with unique material properties. However, none of the presented works attempted to design metamaterials to achieve targeted nonlinear deformation characteristics, as is the objective of RQ2. After thoroughly investigating the literature, five studies were found that directly pursue this design objective.

Wang, Sigmund, and Jensen [82] incorporate TO into the metamaterial design process to achieve nonlinear properties under finite deformation. The objective of the TO was to minimize the error between the current and desired properties. The metamaterials were designed to mimic linear stress-strain curves in the first study, while the second study attempted to achieve specific Poisson ratios. Both truss-based and grid-based materials were designed under the various desired properties. The results show that the TO method could achieve metamaterials with linear elastic constitutive material that exhibit rubber-like responses. The grid-based study, where only the desired Poisson ratio was prescribed, yielded satisfactory results under symmetrical and asymmetrical design limitations. This study shows that a TO approach can design mechanical metamaterial unit cells to achieve desired nonlinear characteristics when the optimization objective aims to minimize the error between current and desired material responses. While these results are promising, this study was limited to tensile load where the desired stress-strain response had limited nonlinearity.

Satterfield *et al.* [83] and Kulkarni *et al.*[84] evaluated metamaterial design for targeted nonlinear compressive deformation characteristics using unit cell synthesis. Both studies used a series of basic functional geometries to design an initial unit cell design and then applied additional optimization to achieve the desired response. Satterfield used zeroth, first, and second-order connection configurations between oval, fixed, and cantilever beam structures, all having predictable nonlinear responses. The nonlinearity of these individual components could be combined in a unit cell design to achieve the desired nonlinear response. Satterfield proposed a "Cantioval" unit cell design with three adjustable size parameters that underwent size optimization. After size optimization, the "Cantioval" design was validated as a viable solution to exhibit the single targeted response.

Kulkarni expanded on the results of Satterfield by proposing a new "Canti-Duo" design with three adjustable parameters that underwent a multi-objective optimization routine. One objective minimized the difference between the current and desired stress-strain response, while the additional objective aimed to minimize the maximum Von mises stress under compression. This additional objective can maintain the proposed design solution within the elastic limit of the base material. The authors used a genetic algorithm to thoroughly explore the design space and achieved a solution that exhibited a similar response to the desired stress-strain curve while ensuring the maximum stress remained below the yield point of the constitutive material.

While these two studies offer promising methods for designing targeted nonlinear deforming metamaterials, there has been no investigation into the generalization of these techniques. These two unit-cell synthesis methods aim to design the unit cell for a single response. If a new response were desired, a novel starting design and accompanying size optimization would be needed. Given the ever-changing nature of material needs, this manual process is not practical.

Incorporating TO methods, Zeng *et al.* [85] designed metamaterials with tunable elastic-plastic responses under large compressive displacements. Finally, Behrou *et al.* [86] used TO to design metamaterials with tunable levels of tensile deformation nonlinearity under various displacement values. These TO methods are limited by the same reliance on the differentiable objective and constraint functions and a lack of generalizability, as previously discussed.

Expanding beyond mechanical metamaterial design for tailorable deformation responses, I found no previous literature explicitly attempting to design deformation and energy-manipulating metamaterials. That being said, mechanical metamaterials are commonly used for energy manipulation. Mechanical metamaterials offer an alternative for mechanical energy manipulation beyond the range of traditional monolithic materials. Through careful design of their unit cell and proper selection of a constitutive material, mechanical metamaterials can achieve unprecedented control over mechanical energy propagation, transformation, and absorption. A common implementation of metamaterials for energy manipulation is for the design of structures with negative or near-zero effective stiffness (auxetic materials) [53], [87]–[89]. These counterintuitive structures have been implemented for energy absorption [90], shock mitigation [91], and vibration isolation [92]. Additional implementations include tailored wave propagation characteristics [93]–[95], frictional unit cells for energy dissipation [96], [97], and tailoring structural compliance while limiting energy loss [83], [84].

The strategic design of the base unit cell plays a pivotal role in attaining these distinctive energy manipulation properties. Previous investigations into energy-manipulating mechanical metamaterials commonly rely on human intuition paired with size/shape optimization [91], [93]–[97], or topology optimization [98], [99]. These design methods lack generalizability, reliance on an initial human design, and limited domain exploration.

Therefore, there is a clear gap in the literature for developing a generalized metamaterial design solution that can cover detailed and diverse design solutions capable of achieving targeted nonlinear deformation and energy manipulation properties. Chapter Four discusses the work done to utilize DRL to design mechanical metamaterials with customizable nonlinear deformation responses in tension and compression. Chapter Five will discuss the advancements to ensure the DRL agent could design for deformation and energy manipulation characteristics. Combining Chapters 3-5 will show that DRL can be a high-performing design tool in complex topology design problems.

CHAPTER THREE

RESEARCH QUESTION 1: FRAMEWORK AND IMPLEMENTATION

3.1 Methods: Topology Optimization as a Deep Reinforcement Learning Problem

To answer RQ1, the discretized TO problem needs to be reformulated as a sequential DRL task, specifically an MDP. If the TO problem can be expressed in terms of the four components of an MDP, state space (S), action space (A), transition probability function (P), and reward function (R), then a DRL agent should have the necessary tools to design optimized topologies. Formally, the TO problem can be viewed as finding the material distribution that optimizes an objective function, F, subject to a volume constraint $G_0 \leq 0$ and possibly M other constraints $G_i \leq 0, i = 1 \dots M$. The material distribution is described by the density variable ρ that can take the value 0 (void) or 1 (solid material) at any point in the design domain Ω. The mathematical form of this optimization problem is shown in Equation 3 [49]:

$$\begin{gathered}\min_{\rho} F \\ F(\boldsymbol{u}(\rho), \rho) = \int_{\Omega} f(\boldsymbol{u}(\rho), \rho) dV \\ s.t.\ :\ G_0(\rho) = \int_{\Omega} \rho dV - V_0 \leq 0 \\ :\ G_i(\boldsymbol{u}(\rho), \rho) \leq 0, j = 1, \dots, M \\ :\ \rho(x) = 0\ or\ 1, \forall\ \boldsymbol{x} \in\ \Omega\end{gathered} \tag{3}$$

where the state field $\boldsymbol{u}$ satisfies a linear or nonlinear state equation. For this application, the objective function, $F(\boldsymbol{u}(\rho), \rho)$, is compliance, where minimizing compliance leads to maximizing the stiffness of the structure. The objective function is dependent on the density of the material at each location, $\boldsymbol{u}(\rho)$. The objective function is also dependent on predetermined constraints, $G_i(\boldsymbol{u}(\rho), \rho)$, which can be subject to change for different applications. Finally, the design space, Ω, indicates the allowable volume, V, within which the design can exist.

The grid-based environment defining the MDP is represented in Figure 3-1 and consists of an N-by-N grid of discrete elements that can be toggled on or off, representing material (light grey) or voided (white), respectively. The material elements are assigned an arbitrary modulus of elasticity of 1 and Poisson's ratio of 0.33. For convenience and to avoid meshing, the voided elements are assigned a modulus of 1^{-4} rather than removing them from the design. Boundary conditions are applied to the nodes of certain bounded elements (black), while loaded elements (dark grey) have a force value applied to their nodes.

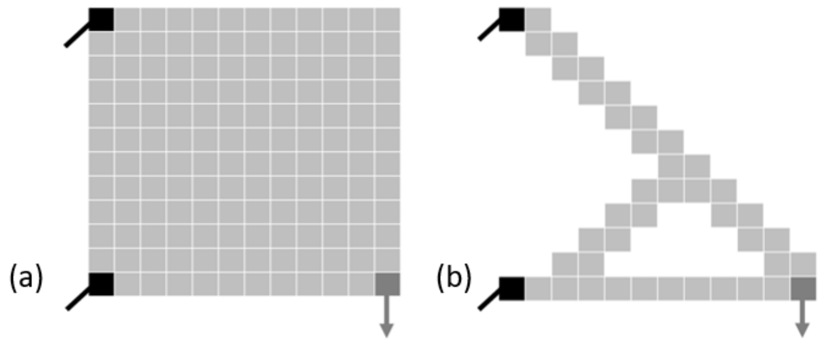


Figure 3-1. 2D representation of a grid-based (a) starting topology of a 12x12 cantilever beam (b) corresponding optimal topology

The details of the agent's actions within the environment are in the following sections, but a brief overview helps guide the description. The environment starts as a solid material block (Figure 3-1a), and the agent must decide which elements to remove to achieve the design objective. After removing an element, FEA provides the stress distribution throughout the structure and feedback on the current performance of the design. The agent uses this stress information to make intelligent decisions about which elements to remove until convergence.

3.1.1 DRL Environment: State Space

The state space, S, of an environment represents all combinations of observations an agent can experience while interacting with the environment. The agent's current observation depends on the boundary conditions, loading conditions, and current topology. The individual observations that comprised the state space were built as NxNx3 arrays, where N represents the number of design variables in one dimension, resulting in N^2 total design variables.

The first channel of the observation array is the normalized *inverse* Von Mises Stress, σ_{VM} of each element and describes the current stress distribution within the topology. The Von Mises stress is commonly used in engineering design to describe an object's current stress state [100], [101]. The Von Mises stress for each 2D element can be found under any load case using Equation 4:

$$\sigma_{VM} = \sqrt{\sigma_x^2 + \sigma_y^2 - \sigma_x \sigma_y + 3\tau_{xy}} \tag{4}$$

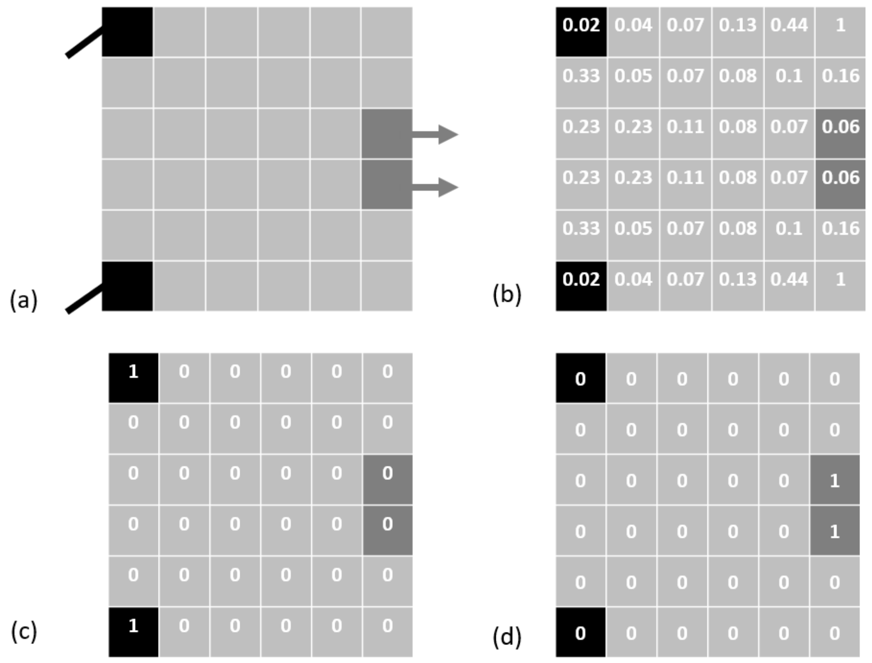
where σ_x, σ_y, and τ_{xy} are found using a 2D plane stress FEA solver, based off the work of Li [102]. Equation 4 is only used for material elements within the topology. Voided elements are assigned a value of 0 to isolate the location of the voided elements. The *inverse* Von Mises stress of each element was used to magnify the difference between minimally loaded and voided elements to help the agent's DNN decipher between similar elements. This difference should prompt the agent to learn that removing minimally load-bearing elements will lead to higher rewards while removing voided elements will lead to a penalty. The inverse Von Mises stresses are normalized about the maximum value of the first channel to prevent unbounded stress values [103]. Therefore, the final representation of the first observation channel can be viewed in Equation 5:

$$O_{1,a,b} = \left(\frac{\sigma_{VM,a,b}}{\sigma_{VM,max}}\right)^{-1} \tag{5}$$

where $O_{1,a,b}$ is the first channel observation of the element at row a and column b, given the Von Mises stress at the element, $\sigma_{VM,a,b}$, and $\sigma_{VM,max}$ is the max stress in the structure.

The second channel of the observation is the Boolean representation of which elements are fixed with boundary conditions such that fixed elements are assigned a value of 1, otherwise 0.

Finally, the third channel provides the loaded element equivalent of the second channel, where loaded elements are assigned a value of 1 and otherwise 0. Figure 3-2 shows an example of a simple 6x6x3 observation under a topology with multiple loaded elements.



0.02	0.04	0.07	0.13	0.44	1
0.33	0.05	0.07	0.08	0.1	0.16
0.23	0.23	0.11	0.08	0.07	0.06
0.23	0.23	0.11	0.08	0.07	0.06
0.33	0.05	0.07	0.08	0.1	0.16
0.02	0.04	0.07	0.13	0.44	1

1	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
1	0	0	0	0	0

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	0	1
0	0	0	0	0	0
0	0	0	0	0	0

Figure 3-2. Observation of a starting 6x6 topology (a) Schematic (b) First channel: Elemental normalized inverse Von Mises stress (c) Second channel: Boolean representation of bounded elements (d) Third channel: Boolean representation of loaded elements

3.1.2 DRL Environment: Action Space

The action space defines all the possible actions the agent can take at each time step. Actions define how an agent can interact with an environment, taking the agent from its current observation to the next. Within the TO environment, an action corresponds to toggling an element from material to void, leading to a new topology. Within an NxN topology environment, the action space size is N^2. The action of toggling an element from material to void can be represented as changing an element's modulus of elasticity from 1 to 1^{-4}.

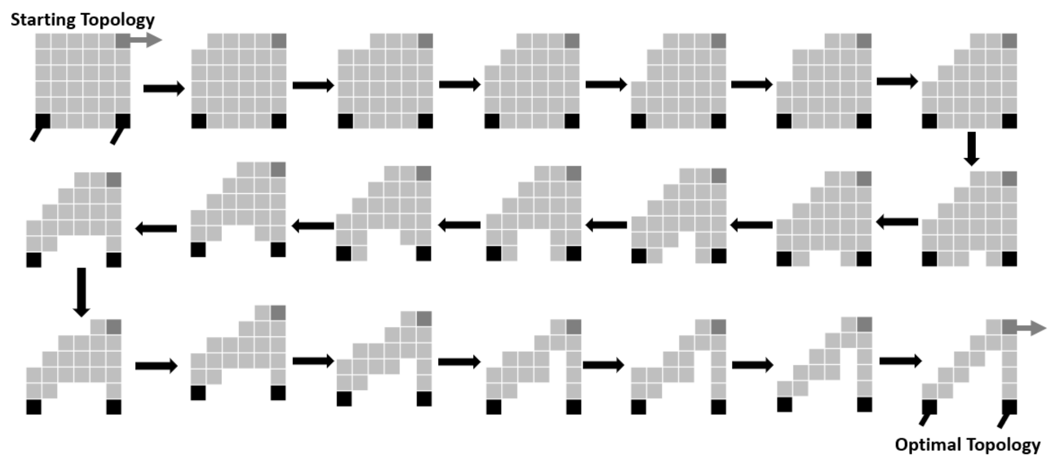


Figure 3-3. Element removal sequence of 6x6 topology

Designing topologies (Figure 3-3) is accomplished by sequentially selecting elements for removal from a starting solid block topology until a termination criterion is reached or the agent attempts an illegal action. The illegal actions, illustrated in Figure 3-4, include trying to remove a bounded, loaded, or previously voided element or an element that would lead to a non-singular body. A non-singular body arises when the topology is held together by a hinge point. This single-node connection at a hinge point does not represent a feasible part of the design solution as it artificially inflates the stiffness of the topology while not representing a physical material connection. Other discretization of the design domain, beyond single element representation, could remove this issue.

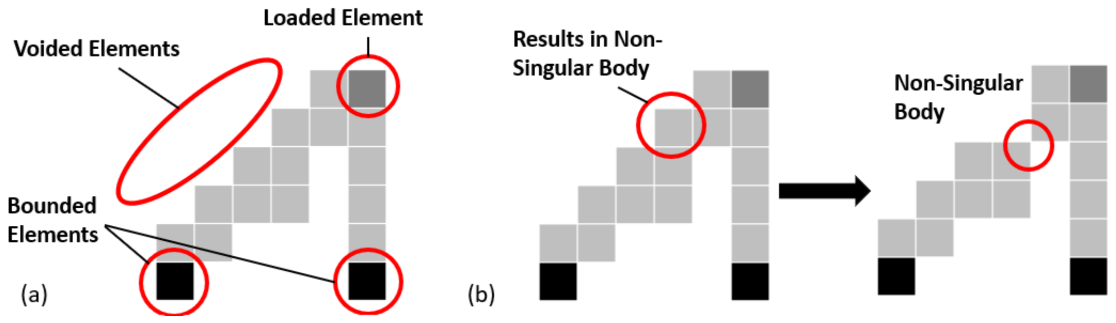


Figure 3-4. Illegal actions (a) Removing voided, loaded, and bounded elements (b) Action results in a non-singular body

3.1.3 DRL Environment: Reward Function

The DRL agent seeks to maximize its reward and should learn what action sequences must be taken to accumulate the most reward while avoiding penalties. In the presented environment, the reward function should be built to minimize structural compliance under specific volume or stress constraints, thus mimicking the objectives of compliance TO.

Minimizing compliance is accomplished by minimizing the increase in strain energy of the current topology compared to the starting solid block topology. The strain energy of a topology is a standard measurement in traditional TO methods to assess the compliance of a topology [100], [101]. The strain energy of a topology increases as more elements are removed, so the agent should receive more reward if it removes elements that lead to a minimal increase. The quadratic mathematical function used to represent this proposed reward function, r_t, at time step, t, is shown in Equation 6:

$$r_t = \left(\frac{c_s}{c_t}\right)^2 + \left(\frac{\alpha_t}{N^2}\right) \tag{6}$$

where c_s is the strain energy of the initial solid block topology, c_t is the strain energy of the current topology at the time step, t, α_t is the number of voided elements at t, and N^2 is the total number of elements.

Equation 6 shows that the agent is rewarded more if the topology exhibits minimal increases in strain energy while reducing volume fraction by voiding more elements. The quadratic representation of the function was selected through experimentation to help magnify the optimal rewards during the later stages of the design process when the removal of various elements could lead to a similar strain energy increase. This addition helped distinguish the optimal action from a group of similar yet slightly inferior actions. While other reward function alternatives are possible, the effectiveness of the function in Equation 6 is demonstrated in the following sections. A graphical representation of Eq. 6 can be found in Figure 3-5.

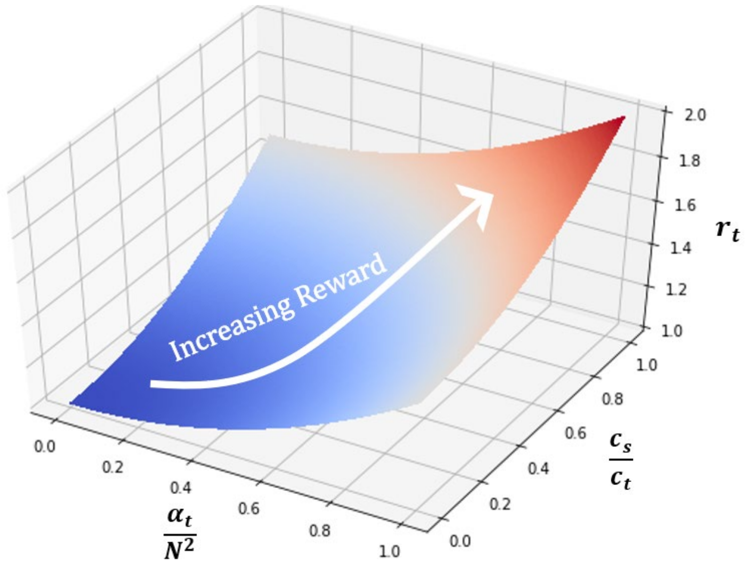


Figure 3-5. Graphical representation of the multi-objective reward function

The reward function of Equation 6 was only used if the agent took an action deemed "legal." If one of the illegal actions of Figure 3-4 is taken, the agent is penalized -1. This penalty teaches the agent to avoid taking illegal steps as it diminishes the accumulated reward of a designing episode.

Combining the positive reward function and negative penalty means the reward between two sequential actions could differ significantly. For example, the agent could be nearing the final design steps of an optimal topology and accumulating significant reward when suddenly it makes an illegal move and is penalized. This phenomenon where the last action's reward differs substantially from the previous rewards is known as reward saltation [42]. Positive reward saltation occurs if the current reward is significantly larger than the previous reward, whereas negative reward saltation would be the opposite. Reward saltation in an environment can lead to training inefficiencies and convergence to a locally optimal policy. Therefore, to address reward saltation in the proposed reward function, a magnified saltatory reward (MSR) algorithm, proposed by *Hu et al.* [42], was introduced to augment the reward function.

The MSR algorithm aims to magnify the reward if positive saltation occurs and reduce the current reward if negative saltation occurs. Therefore, the algorithm causes the agent to be more cautious when encountering risk, such as nearing an optimal topology, and more decisive when encountering high-yield choices, such as the initial stages of topology design. The MSR algorithm is applied to every time step to produce an augmented reward value, r_t^*, using the following series of equations [42]:

$$r_t^* = r_t + (f(x) - \lambda)|r_t + \varepsilon| \tag{7}$$

$$f(x) = \tan^{-1}\left(x\frac{\pi}{2\eta}\right) \tag{8}$$

$$x = \rho + \lambda \tag{9}$$

$$\lambda = sign(r_t - r_{t-1}) \tag{10}$$

$$\rho = \frac{(r_t - r_{t-1})}{\min\left(|r_t + \varepsilon|, |r_{t-1} + \varepsilon|\right)} \tag{11}$$

where $r_t$and $r_{t-1}$are the current and previous rewards, respectively, ε is a negligible non-zero value needed for numeric stability in Equation 11 if both rewards were zero, $f(x)$ is the monotonically increasing bounded function used to define the allowable range of reward changes, and η is a parameter used to determine if the difference between the current and previous rewards represents significant reward saltation and justifies magnifying or reducing the current reward.

3.1.4 DRL Environment: Transition Probability Function

Given the state at the current time step, s_t, the agent needs to select the action within the action space, a_t, that will lead to the largest value, $Q(s_t, a_t)$, where $Q(s_t, a_t)$, is defined as the total expected future reward if the action, a_t, is chosen. The transition probability function, also called the value function, maps state-action tuples, (s_t, a_t), to values for each action. If $Q(s_t, a_t)$, is known for all actions when the agent experiences a particular state, then selecting the $argmax(a_t)$ is the best action. However, even for a simple representation of the TO environment with 6x6 topologies, the number of possible states for a single load case is large ($\approx 6.87 \times 10^{10}$), which makes the value function complex and challenging to model. Therefore, $Q(s_t, a_t)$ is approximated using a DNN. The value function is found using the Double Q-learning algorithm [104], as seen in Equation 12:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma max_a Q'(s_{t+1}, argmaxQ(s_{t+1}, a_t)) \tag{12}$$

where $r(s_t, a_t)$ is the reward for taking the action, a_t, at the current state, s_t, and γ is the discount factor, quantifying the importance of future rewards. Double Q learning uses two DNN models, a main and an auxiliary model. The main model is used for action selection by predicting the action with the largest value, $Q(s_t, a_t)$ and the auxiliary model predicts the value of the next state if that maximum value action was taken, $max_a Q'(s_{t+1}, argmaxQ(s_{t+1}, a_t))$. The introduction of the auxiliary model helps predict the next state value to address the overestimation of future value, commonly seen in the generic Q learning algorithm. This small change in Double Q learning has been shown to improve algorithm stability [104].

The main and auxiliary models consist of four convolutional neural network (CNN) layers, each with a varying number of 3x3 convolutional filters. The results for each layer are passed through a ReLU activation function before proceeding to the subsequent layer. The purpose of a

CNN is to extract relevant features from the state representation matrix and use those features to make decisions about the value of each possible action. The CNN architecture, represented in Figure 3-6, shows the agent's NxNx3 observation of the current state is the input and is passed through four convolutional layers comprised of 16, 8, 4, and 1 convolutional filters, respectively, to produce an output of the predicted value of taking each action. The action associated with the highest value is selected, and that element is removed, leading to the next observation.

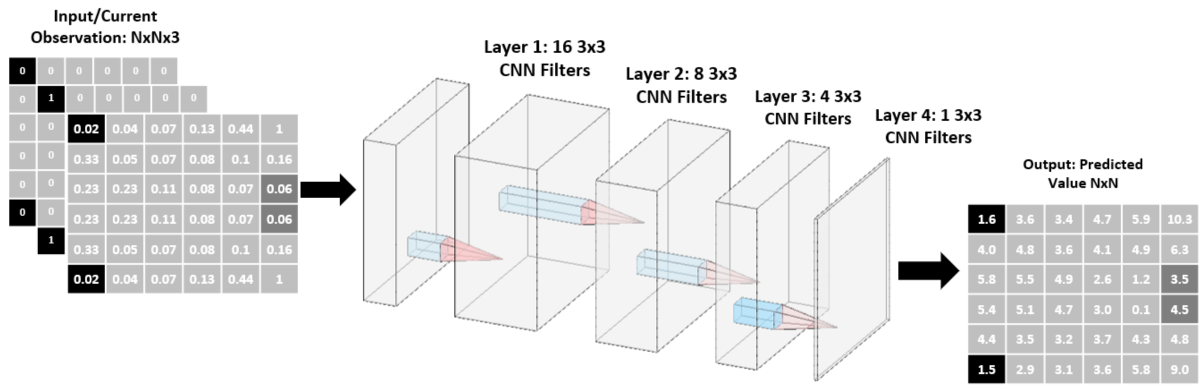

Input/Current Observation: NxNx3
Layer 1: 16 3x3 CNN Filters
Layer 2: 8 3x3 CNN Filters
Layer 3: 4 3x3 CNN Filters
Layer 4: 1 3x3 CNN Filters
Output: Predicted Value NxN

Figure 3-6. Deep convolutional neural network architecture

3.2 Methods: Training The Agent

Section 3.1 defined the TO problem as an MDP; therefore, the DRL agent should be able to learn to satisfy the compliance minimization TO problem. The agent's performance depends on its ability to approximate the value function in Equation 12. The agent must undergo episodic interactions with the proposed environment to collect experiences and update its DNN weights for better value approximations. The DNN weights are updated to minimize the error between the agent's prediction of $Q(s_t, a_t)$ and the actual value, which is realized after the agent has taken the corresponding action, a_t. An episode in this environment is defined as a sequence of element removal actions starting at the initial solid block topology and continuing until one of the illegal moves of Figure 3-4 is made. During training, so long as the agent does not attempt to remove a bounded, loaded, or voided element, the agent can continue to take actions to remove elements from the topology until a non-singular body is ultimately produced. This approach increases the accumulated experiences to update the DNN weights. This approach contrasts with specifying stress-based or volume fraction termination criteria, which will be implemented for testing.

All training was completed on a 6x6 topology size to limit the computational cost of running the FEAs needed to produce the first observation channel at each step. The DRL agent is trained to design topologies with randomly generated load cases to ensure thorough state-space exploration. The load cases were introduced for each episode by randomly assigning two elements on the exterior edge of the topology to act as bounded elements. A bounded element was represented by providing a static displacement boundary condition on the two exterior nodes of the selected element. If a corner element was selected, the single exterior corner node was assigned the boundary condition.

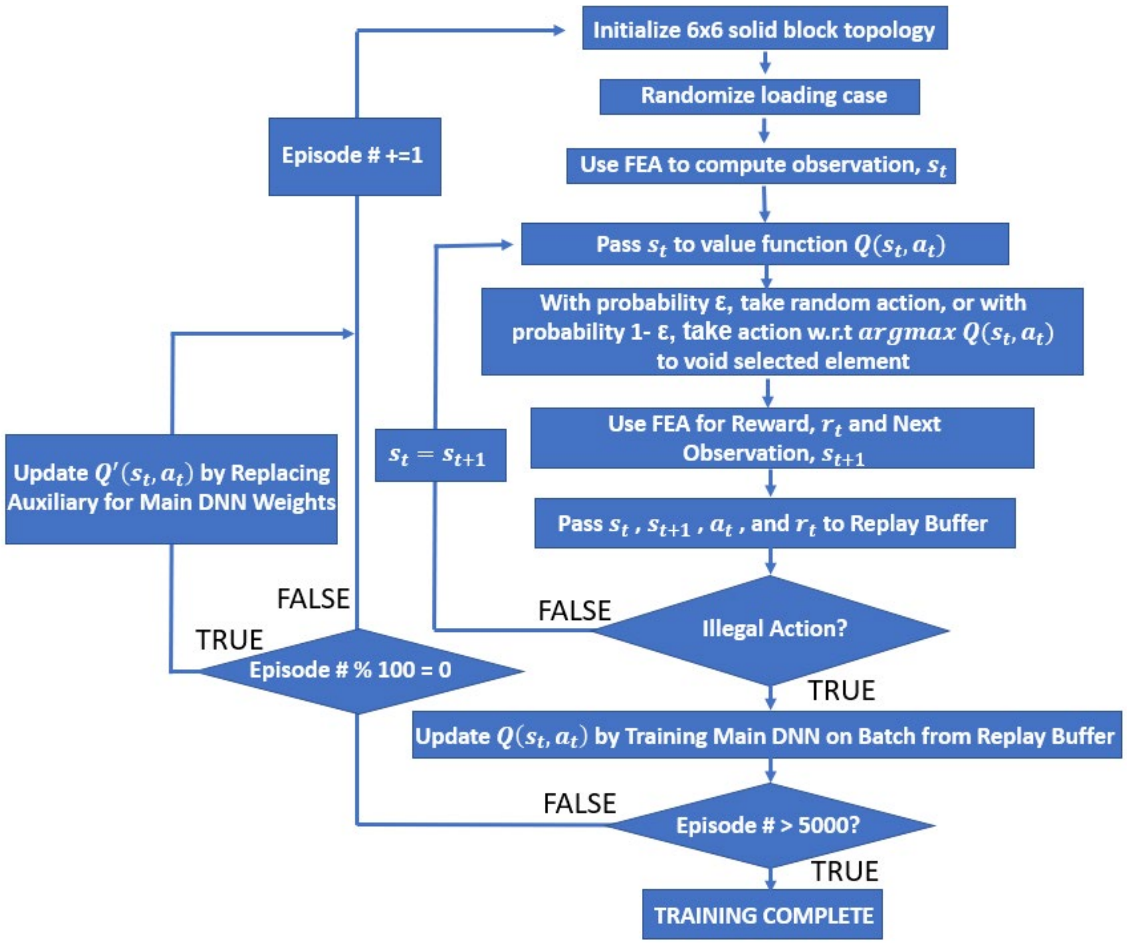
A single element was randomly selected to serve as the loaded element. The element was randomly assigned in the 6x6 topology so long as it wasn't previously selected as a bounded element. The direction (horizontal or vertical) and type (tensile or compressive) of the load were also randomly selected. If the selected element was exterior, the load was distributed between the two exterior elemental nodes. If a corner element was selected, the load was applied to the single exterior corner node. Finally, if the element was in the interior, the load was randomly distributed between two elemental nodes defining one of the edges.

A fundamental dilemma of DRL training is the trade-off between exploration and exploitation. The agent explores the environment by taking random actions to compile diverse experiences to build a sufficiently accurate mapping between actions and values. However, to ensure efficiency, the agent should not strictly rely on random actions and instead exploit the value mappings it has already learned. Therefore, to balance proper exploration and exploitation, random behavior should be more frequently used when the value predictions are poor at the beginning of training. As training continues and the value predictions improve, random behavior should be replaced by exploiting the maximum value action, $argmax(Q(s_t, a_t))$. This method is called the ε -greedy method [44]. The probability of taking a random action depends on a decaying ε value. At the start of the training process, the ε value is 1 and decays for each episode following Equation 13:

$$\varepsilon = max\left(1 - E \cdot \varepsilon_{decay}, \varepsilon_{min}\right) \tag{13}$$

where ε_{decay} is the rate at which ε decreases per episode ($\varepsilon_{decay} = 3.5e - 4$) until ε reaches its minimal allowable value ($\varepsilon_{min} = 0.01$), and E is the current episode number. ε_{min} is greater than zero to ensure exploration continues during the later stages of training. These values were selected after experimenting with several combinations of ε_{decay} and ε_{min}. I do not claim that the two selected values are necessarily optimal. However, I believe a sufficient balance between exploration and exploitation is met.

Whether randomly or strategically taken, the action will remove the corresponding element leading to a new topology and corresponding new observation. Based on the results of the observation FEA, the agent will receive a reward for taking that action. After each action, the previous observation, new observation, action, and reward are recorded in a replay buffer. Using topological symmetry, the agent stores the equivalent x and y-axis flipped observations, corresponding flipped actions, and reward in the replay buffer. This technique allowed four experiences to be captured from a single FEA call for each step and is essentially a simple data augmentation method. Once an episode is complete and the buffer size is at least 128, a batch of 128 past experiences is randomly selected to train the DNN. Randomly selecting a batch from the replay buffer ensures the agent is trained on independent and identically distributed (i.i.d) data to help promote generalization. The main model weights were updated using the stochastic gradient descent ADAM optimizer with a learning rate of $2.5 * 10^{-3}$. The auxiliary model weights are replaced with the current main model weights after every 100 episodes. A flowchart of a training episode is shown in Figure 3-7.



Initialize 6x6 solid block topology
Randomize loading case
Use FEA to compute observation, s_t
Episode # +=1
Pass s_t to value function $Q(s_t, a_t)$
With probability ε, take random action, or with probability 1- ε, take action w.r.t $argmax\ Q(s_t, a_t)$ to void selected element
Use FEA for Reward, r_t and Next Observation, s_{t+1}
$s_t = s_{t+1}$
Update $Q'(s_t, a_t)$ by Replacing Auxiliary for Main DNN Weights
Pass s_t , s_{t+1} , a_t , and r_t to Replay Buffer
FALSE
TRUE
Episode # % 100 = 0
FALSE
Illegal Action?
TRUE
Update $Q(s_t, a_t)$ by Training Main DNN on Batch from Replay Buffer
FALSE
Episode # > 5000?
TRUE
TRAINING COMPLETE

Figure 3-7. Episodic Training Flowchart

3.3 Methods: Testing the Agent Using Progressive Refinement

The trained agent was tested to validate its accuracy and generalizability. The agent is trained to design 6x6 topologies to limit the computational cost of running the FEA calculations to produce the environmental observations. Designing an optimal topology at the coarse 6x6 level does not have sufficient detail to be beneficial because of the limited degrees of freedom. For this reason, testing on a finer mesh size will result in a more meaningful evaluation of the agent's abilities. The DRL agent can interact with topologies larger than it is trained on because the value function DNNs are built solely on convolutional filters. Convolutional filters are not size-dependent and can be passed along any input size. Therefore, without retraining, the agent could interact directly with a larger topology size (e.g., 24x24 or beyond). However, the computational cost of interacting with a larger topology from starting solid block to optimal topology is significantly larger than interacting with a 6x6 topology. Therefore, the authors propose implementing progressive refinement to help improve the computational efficiency of the DRL agent.

Progressive refinement is the process of sequentially increasing the complexity of an object from "coarse" to "fine." Regarding topology design, progressive refinement increases the topology detail over sequential steps instead of all at once [105]. As a result, the number of elements used to define an equivalent topology shape increases through intermediate steps, as seen in Figure 3-8. In this work, progressive refinement was carried out from a starting 6x6 topology to an intermediate 12x12 topology to a final 24x24 topology which was used to define the agent's proposed optimal topology for each load case. Figure 3-8 shows that the bounded and loaded elements are transferred between the mesh refinements. This transfer alters the proportion of elements that are applying the boundary and loading conditions and may alter the stress distribution and should be addressed in future work.

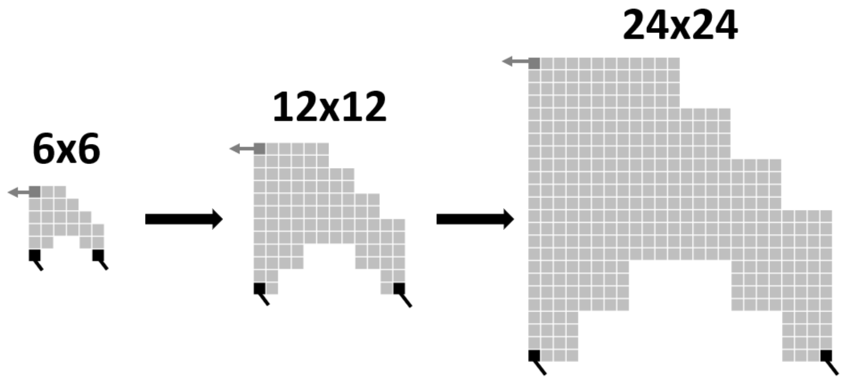


Figure 3-8. Two-step progressive refinement from 6x6 to 24x24 equivalent topologies

Progressive refinement improves computational efficiency by allowing the agent to start taking actions and removing elements in the coarse 6x6 topology until an intermediate volume fraction (VF) or stress-based termination criterion is met. The intermediate termination criteria for the 6x6 and 12x12 sizes are with respect to a global termination criterion for the 24x24 size, as seen in Equations 14 and 15. The agent begins by removing elements from the 6x6 topology until the initial intermediate termination criterion is met. Next, the agent interacts with an equivalent 12x12 topology and continues to remove elements until a second intermediate termination criterion is met. Finally, the agent interacts with the equivalent 24x24 topology until the final, user-specified termination criterion is satisfied. An example of this sequence is represented in Figure 3-9, where the final termination criterion is $VF_{24x24} = 0.25$. It should be noted that Equations 14 and 15 would need to be adjusted if the final topology refinement were to extend to 48x48 or 96x96 and could be addressed as future work.

$$VF_{6x6} = 1 - \left(\frac{1 - VF_{24x24}}{2.5}\right) \tag{14}$$

$$VF_{12x12} = 1 - \left(\frac{1 - VF_{24x24}}{1.5}\right) \tag{15}$$

The sequential decrease in allowable VF follows the work of Kim and Weck [105], who found that the optimization method works better by gradually tightening the criteria with increasing refinement.

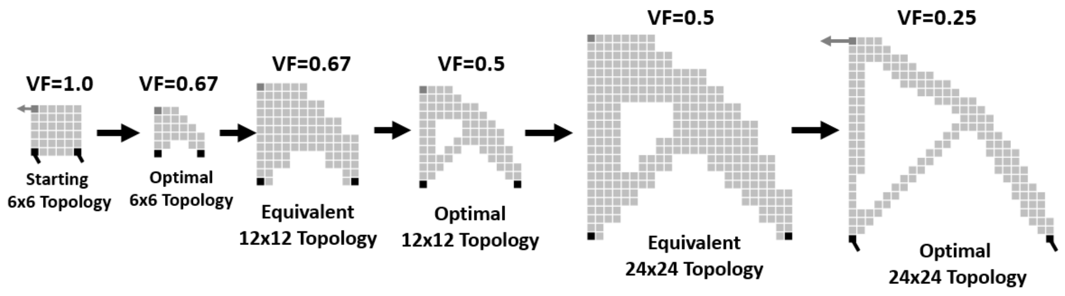


Figure 3-9. Sequential element removal using progressive refinement

Starting progressive refinement with a 6x6 topology significantly improves the computational efficiency. This method reduces the actions the agent needs to take because removing a single coarse element from the 6x6 topology is equivalent to removing the 4 or 16 corresponding elements from the 12x12 or 24x24 topologies, respectively. Therefore, the agent interacts with the intermediate 6x6 topology to produce a general, non-detailed shape that can be formed into an equivalent 12x12 topology. The agent can begin to produce low-level details at the 12x12 topology but will ultimately achieve its sufficiently detailed final design at the 24x24 topology size. I hypothesize that progressive refinement could be incorporated for topology sizes larger than 24x24, but due to the computational cost of running the FEA, the 24x24 size was deemed viable for testing purposes.

3.4 Results

3.4.1 Training Results

The training was run for 5000 episodes with ε reaching ε_{min} shortly after 2800 episodes. The training took approximately 1.5 hours using a PC with Intel(R) Core(TM) i7-10510U CPU @

1.80GHz and 16 GB of RAM. The program was implemented within a Python 3.7 environment. Figure 3-10 shows a moving average of the training reward over 100 episodes. At the same time, ε decreases as a function of the episode number until it reaches the minimum ϵ_{Min}. After ϵ_{Min} is reached, the reward starts to generally converge towards a maximum. The average reward does not converge to a single value because the maximum reward the agent could accumulate during an episode depends on the load case and the corresponding number of elements removed. The increase in reward in Figure 3-10 indicates the agent has learned to take more strategic actions that lead to designing topologies that better satisfy the design objectives.

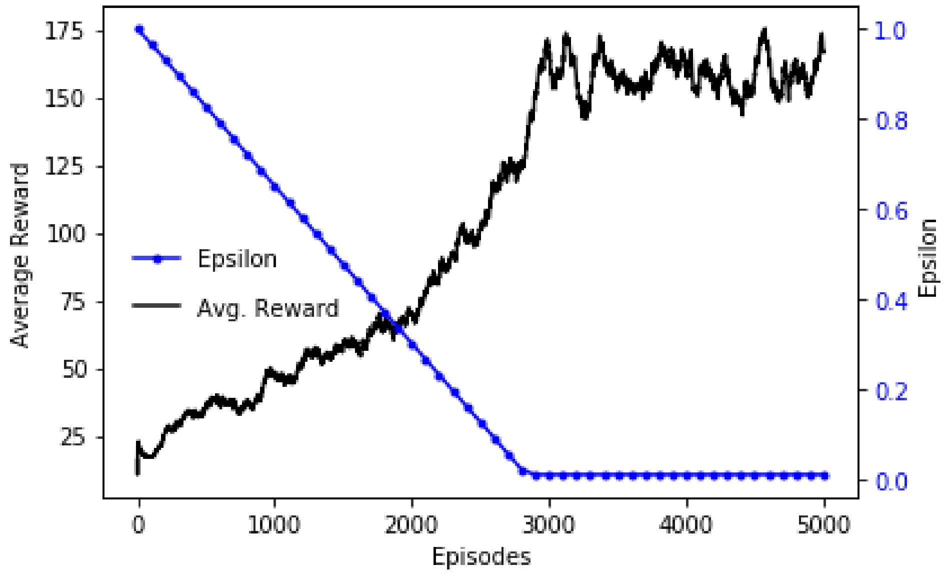


Figure 3-10. Average reward and epsilon-decay during training

3.4.2 Test Case Results

Several test cases are presented to assess the performance of the algorithm. Each test case introduces a unique combination of loaded and bounded elements. These elements are represented in the second and third channels of the observation array for 24x24 topology size. However, to account for progressive refinement, the equivalent elemental regions in the 6x6 and 12x12 topology observations were also treated as loaded or bounded elements.

The test cases were run using a user-specified final volume fraction (VF) or allowable stress increase (SI) termination criteria. User-specified termination criteria ensure that the presented method can adapt to satisfy various design constraints. A user-specified VF termination criterion specifies the final desired VF for the 24x24 Topology, where the intermediate VFs are defined using Equations 14 and 15 in Section 3.3. A stress-based termination criterion is reached when the ratio of the current p-norm stress divided by the initial p-norm stress exceeds a user-specified value. The p-norm stress, σ_P can be calculated using Equation 14:

$$\sigma_P = \left(\sum_{i=1}^{N^2} \sigma_{VM,i}^{p} \right)^{\frac{1}{p}} \tag{14}$$

where $\sigma_{VM.i}^{p}$ represents the Von-Mises stress at element i raised to the power of the smoothing parameter constant, p. The p-norm stress eliminates the local stress concentrations and replaces them with a single approximation of the maximum global stress [106]. A larger smoothing parameter better represented the global stress value and was set to 10 for this application. A larger allowable SI would lead to a topology with a small final VF and a more significant increase in strain energy, as seen in Figure 3-11. A test case could implement one or both termination criteria to satisfy a designer's constraints.

..

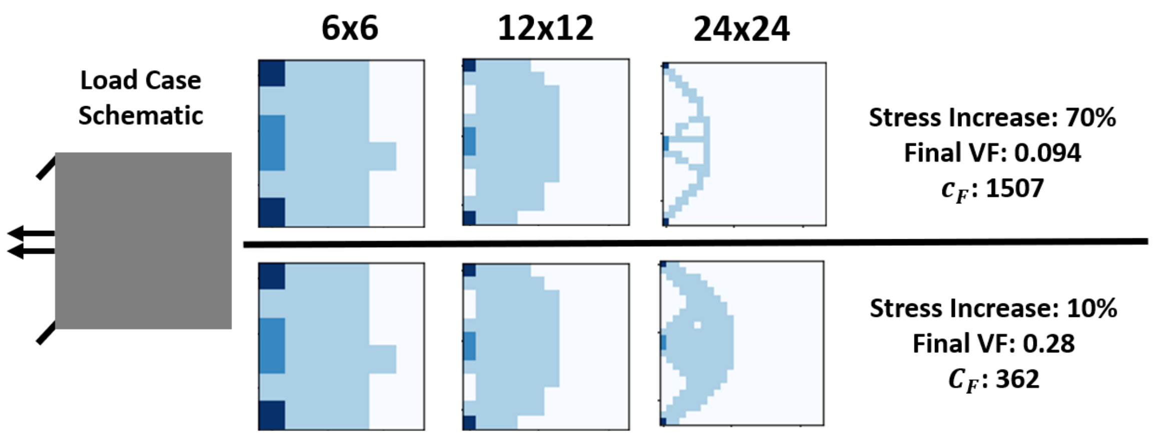
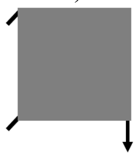
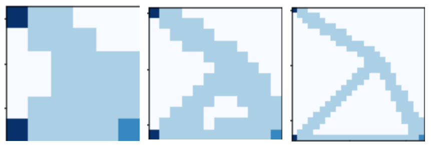
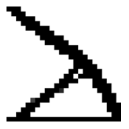
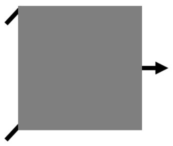
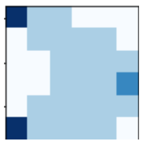
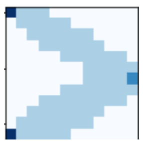
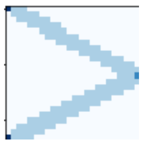
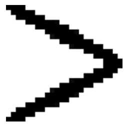
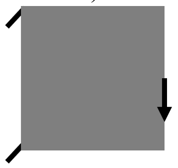
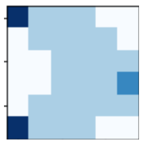
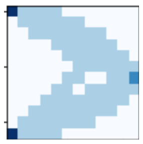
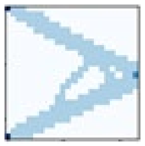
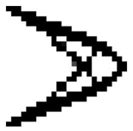
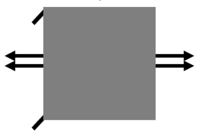
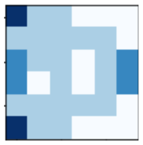
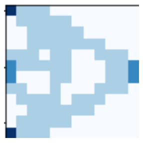
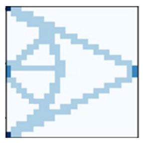
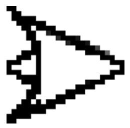
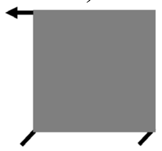
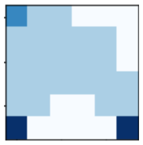
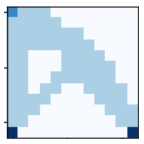
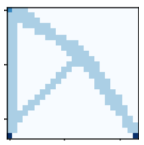
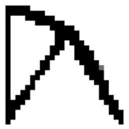
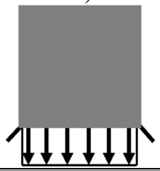
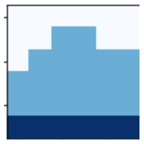
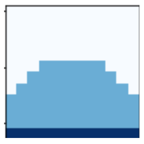
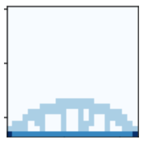
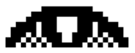


Figure 3-11. Allowable stress increase termination comparison of final strain energy (c_f) [J] and volume fraction (VF)

A series of test cases with common load cases were run, and the results are shown in Table 3-1. The final 24x24 topology has been included, along with the intermediate 6x6 and 12x12 topologies. The final topology for each load case was determined using a user-specified final VF or SI. The final proposed 24x24 topologies from the DRL designer were compared to a more traditional, gradient-based TO algorithm produced by Sigmund using the 99-line SIMP TO algorithm with a penalty factor of 3 and filter radius of 1.5 to ensure the topology is built using discrete elements [107].

Table 3- 1. Test case result comparison of final strain energy [J], c_F, and final volume fraction, VF_F, using volume fraction, VF, or stress increase, SI, termination criteria

Load Case Schematic	6x6 |12x12 | 24x24 Topologies	Termination Criteria	VF_F	c_F	Gradient-Based Equivalent		
					Topology	c_F	VF_F
A) 		VF=0.25	0.25	864		964	0.25

B) 				VF=0.25	0.25	193		211	0.25
C) 				SI=50%	0.31	768		806	0.25
D) 				VF=0.25	0.25	1853		929	0.25
E) 				VF=0.25	0.25	879		1000	0.25
F) 				SI=30%	0.16	1021		1245	0.2

Each proposed topology satisfies the specified termination criteria. The diversity of load cases is used to show the generalization capabilities of the DRL agent. Under these unique load cases, the DRL agent can consistently produce topologies similar to a more traditional gradient-based TO algorithm. In all load cases except D, the DRL agent produces a topology with lower stain energy than the gradient-based solver. Each proposed topology from the DRL agent represents a feasible design solution. The connections between individual elements build a manufacturable and viable topology and directly result from the DRL agent terminating an episode as soon as an illegal action is taken.

It should be noted that each design episode conducted by the DRL agent can require up to 11, 28, and 144 FEA calls for the 6x6, 12x12, and 24x24 topologies, respectively. Therefore, the computational efficiency of the DRL method is not a key advantage but can depend on a user's computational resource availability. However, when the computational time of the FEA is not considered, the time difference between a traditional gradient-based method and the proposed DRL method is trivial. Alternatively, a user could implement a pre-trained surrogate model for near-instantaneous stress distribution calculations, significantly reducing the computational burden of the proposed method.

3.5 Discussion

3.5.1 Learning a Generalized Design Strategy

Table 3-1 shows that the DRL agent has learned a generalized design strategy by satisfying the TO-based multi-objective reward function. For the agent to learn a design strategy, it must exhibit strategic thinking. Strategic thinking can be defined as using tools to navigate comprehensive potential approaches to find the one that best suits an objective [108]. For the DRL agent, the set of tools is the action space to void elements within the topology to maximize the reward function. As the agent must use the same value function for different load cases, the design strategy is general enough to account for various load cases, many of which the agent may not have experienced during training. A primary concern within DRL is that the agent will overfit by memorizing the best actions from the training experiences but fail to generalize to future, unseen experiences. The authors addressed these concerns with the methods used to test the agent.

The first indication of generalization is that the agent only interacted with 6x6 topologies during training, but during testing, the agent removed elements on 6x6, 12x12, and 24x24 topologies. Thus, the agent applied the same design strategy learned during the 6x6 training to the larger environments. The second indication of generalization can be seen in Load Cases D and F, as multiple elements have been assigned as loaded elements. During training, only a single element was randomly selected to serve as the loaded element. Therefore, the agent could not have experienced these load cases during training and could not be memorizing from past experiences. These two load cases indicate that the agent can generalize to pick the best design strategy under unique, unseen load cases.

DRL methods can often be subject to the black box problem, meaning the underlying strategy of the agent used to relate the inputs and outputs is left a mystery [44]. Understanding the black box strategy of a DRL agent can help human users guide choices in the future. Concerning engineering design, an expert designer can understand the DRL agent's strategy by tracking the observation-action pairs that lead to the maximum reward. This approach should allow the human designer to gain a deeper understanding of the design task and produce superior design solutions that may not have been previously apparent.

The best design strategies of the DRL agent in the TO environment should be built around maximizing the multi-objective reward function under varying load cases. Therefore, the agent should be targeting the removal of minimally load-bearing elements. Removing a minimally load-bearing element reduces the volume fraction of the topology and allows a minimal increase in strain energy. This strategy is commonly adopted in other sequential-based TO solvers [51], [109], [110]. Isolating minimally load-bearing elements can be discovered by viewing the elemental stress distribution. Fortunately, the first observation channel is each element's proportional inverse Von-Mises stress. This representation leads to minimally load-bearing elements having the smallest Von-Mises stress and the largest first-channel observation values. Therefore, to infer that the DRL agent has adopted design strategies that target the removal of minimally load-bearing elements, the elements with the largest first channel observation should not be present in the final topologies.

Figure 3-12 shows the first channel observations for a sequential 6x6 topology design without progressive refinement. The agent routinely selects to remove elements with larger proportional inverse Von Mises stresses. While the agent does target larger values, it does not strictly remove the element with the largest value, corresponding to the minimally load-bearing element. Instead, the agent adopts a strategy that also accounts for the locational relationship of elements in the topology. Examples of this strategy can be seen several times in Figure 3-12, such as steps 2, 11, and 20. Step 20 is particularly interesting because if the agent removed any of the top three minimally load-bearing elements, the resulting topology would not be a singular body, terminating the episode. Instead, the agent selects to remove one of the few elements that will still lead to a singular body. This addition is important because if the agent simply removed the minimally load-bearing element at each step, the DRL application would be deemed trivial, as this strategy could be hard-coded for any topology with any load case. Therefore, the agent has learned a design strategy that targets the removal of minimally-load bearing elements while accounting for these elements' locational distribution. The same agent is used to design at the 6x6, 12x12, and 24x24 topology sizes and, therefore, should adopt the same strategy for future generalized topology designs at increasing mesh complexity.

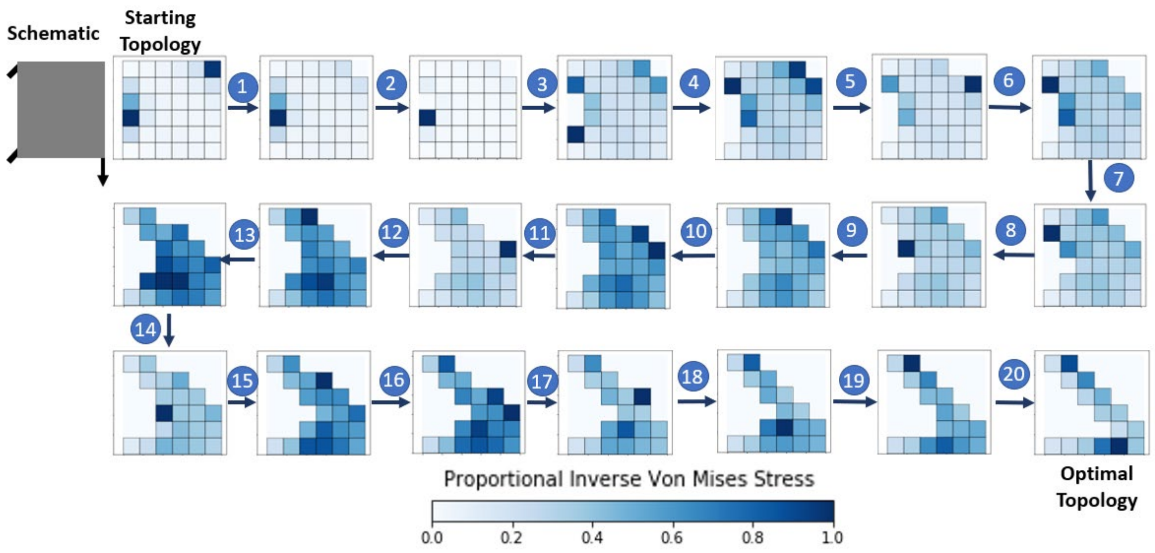


Figure 3-12. First layer observation of a 6x6 topology design episode

3.5.2 Deep Reinforcement Learning Shortcomings

Common to many TO methods [49], [100], [106], [111], the DRL agent's proposed topology appears to be sub-optimal under certain load cases. The proposed DRL topology algorithm outputs a poor-performing topology when the agent takes an illegal step early during the design process and cannot reach the user-specified termination criteria. The authors determined that the DRL algorithm outputs poor-performing topologies when the load case contains several bounded and loaded elements randomly distributed around the starting solid block topology or when multiple bounded or loaded elements are clumped together, shown in Figure 3-13. These examples represent irregular load cases that would only be introduced given a highly particular design objective.

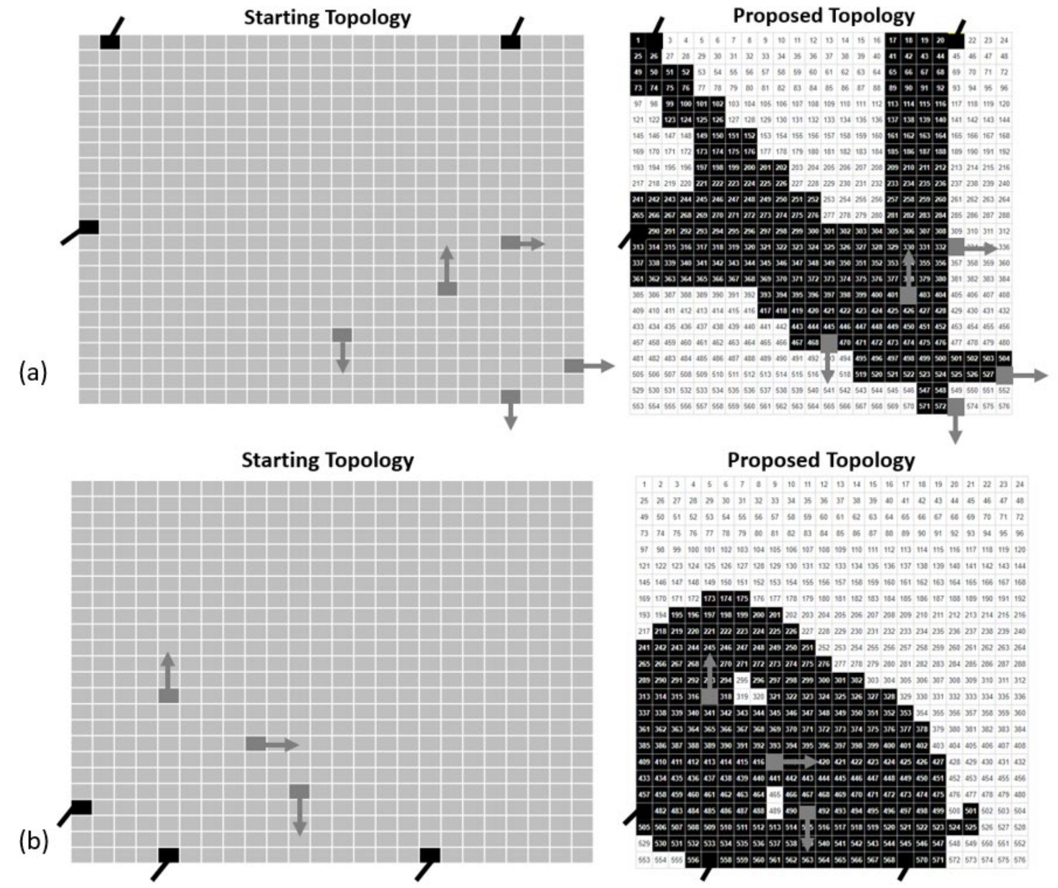


Figure 3-13. Sub-optimal proposed topologies for (a) dispersed and (b) clumped loading

The hypothesis for the sub-optimal topology in Figure 3-13a is that when the bounded and loaded elements are distributed throughout the topology, fewer elements can be legally removed at the 6x6 and 12x12 stages. Therefore, the agent is more likely to take an illegal action at the 6x6 or 12x12 topologies before reaching the corresponding intermediate volume fraction. As a result, the agent must successfully take more actions at the 24x24 topology, increasing the opportunity for sub-optimal solutions. A proposed solution is to adjust the intermediate volume fractions depending on the distances between the bounded and loaded elements. More considerable distances between elements would require higher intermediate volume fractions to avoid prematurely eliminating necessary clumps of elements. In comparison, closely packed elements would require smaller intermediate volume fractions to ensure the agent takes advantage of removing large groups of elements on the 6x6 and 12x12 topologies.

Figure 3-13b shows a challenging loading case with closely packed groups of bounded and loaded elements. Incorporating progressive refinement into the design process relies on producing the corresponding topologies and bounded and loaded elements at different topology sizes. The initial elements for each load case are selected at the 24x24 topology size but correspond to equivalent elements on the smaller topology sizes. A problem arises when the load case elements are closely packed together. Independent elements on a 24x24 topology can have the same equivalent element on the 6x6 or 12x12 topologies. Therefore, unique elements on the 24x24 topology can now be lost within the coarse topologies, leading to a difference in load cases between the topology sizes in a single design sequence. Load cases with clumped elements do not guarantee a severely sub-optimal design but have a higher probability than the more uniformly distributed load cases in Table 3-1 of Section 3.4.2. Given the reliance on progressive refinement and the equivalent bounded and loaded elements, the DRL topology designer should target more traditional load cases with a relatively even distribution of notable elements.

This research question, published in the following article [37], was limited to the engineering design task of 2D grid-based topology design. However, I believe DRL can be incorporated into many engineering design applications. I hypothesize that if a DRL environment, like the one proposed in this work, can be formulated to represent a specific design domain, then a DRL agent can identify high-performance designs regardless of the domain. These design domains could include but are not limited to metamaterial design, sequential generation of 3D geometries (similar to parametric CAD software), multi-material topology design, and systems engineering. The following sections will discuss additional design domains that can incorporate DRL to achieve superior design solutions. The following two chapters will discuss the use of DRL for mechanical metamaterial design with customizable deformation and energy manipulation properties. This design task is considerably more challenging when compared to the well-established compliance minimization problem. Therefore, the forthcoming chapters will further prove DRL as a powerful engineering design tool.

CHAPTER FOUR

RESEARCH QUESTION 2: FRAMEWORK AND IMPLEMENTATION

The next phase of this research focuses on the complex design domain of mechanical metamaterials. The objective of RQ2 targets the designing of mechanical metamaterials with tailorable nonlinear deformation responses through local optimization of the unit cell. The unit cell design process is a sequential design problem that requires thorough state-space exploration. The investigation of the literature shows a clear gap in providing a generalized method for designing metamaterial unit cells that exhibit targeted nonlinear deformation responses. The increased complexity of the metamaterial design domain makes it challenging for traditional design and optimization methods to efficiently and effectively find high-performing, viable solutions. I hypothesize that RQ2 can be answered by reframing the unit cell design process as a DRL task. The same DRL components of RQ1 must be addressed but altered to capture the increased complexity of the new design domain. Therefore, this chapter will discuss the domain changes and new DML tools introduced to the DRL task to successfully design mechanical metamaterials with customizable deformation responses.

4.1 Methods: Representing the Metamaterials

4.1.1 Discrete Grid Design Domain

The metamaterial unit cells are represented within a 2D grid design domain. The design variables of this discretized domain are viewed as discrete quadrilateral elements, each defined by four nodes. The material distribution of the unit cell is defined by the material density of each element, which takes the value of 1 (material) or 0 (void). The material elements are linear elastic and assigned a reference modulus of elasticity of 1 and Poisson's ratio of 0.33.

A 60x20 matrix of discretized elements represents the design domain of the unit cell. The material distribution of the unit cell is sequentially altered by adding material elements to the 60x20 matrix according to cubic Bezier functions and mirroring about the X and Y axes, as seen in Figure 4-1.

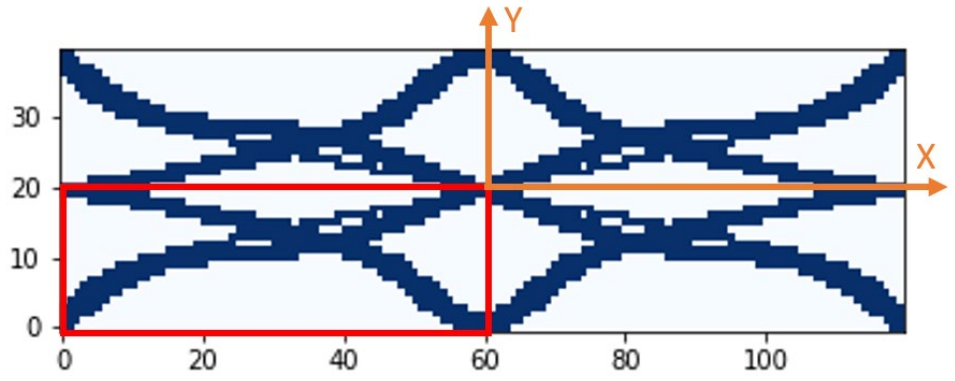


Figure 4-1. A randomly generated unit cell of discrete elements (material in blue, void in white) with design domain (Red) mirrored about the y and x axes

This work only investigates uniaxial compression and tension in the Y-direction. Therefore, mirroring the 60x20 design domain about the Y-axis ensures symmetric deformation. Additionally, mirroring the domain about the X-axis results in an increased geometric nonlinearity that results in more unique nonlinear and orthotropic deformation responses [112].

Altering the material distribution of the unit cell according to cubic Bezier curves offers a repeatable method that can produce a diverse range of geometric nonlinearities. Bezier curves are commonly used in computer graphics [113] but have proven to be a robust engineering design tool for airfoil profiles [114], design for manufacturing [115], and metamaterial design [116]. A set of discrete control points define the smooth, continuous Bezier curve. The formula for the x, $B_x(t)$, and y, $B_y(t)$, coordinates of the Bezier Curve are given by Equation 15:

$$\begin{aligned} B_x(t) &= \sum_{i=0}^{n} \binom{n}{i} P_{i,x}(1-t)^{n-i}t^i \\ B_y(t) &= \sum_{i=0}^{n} \binom{n}{i} P_{i,y}(1-t)^{n-i}t^i \end{aligned} \qquad for\ t{:}\,[0,100] \qquad (15)$$

where $P_{i,x}$ and $P_{i,y}$ are the x and y coordinates of the *i-th* discrete control point, and n is one less than the number of control points. Cubic Bezier curves (Figure 4- 2) are defined by the coordinates of an initial point, P_0, final point, P_3, and two intermediate points, P_1 and P_2, therefore n=3.

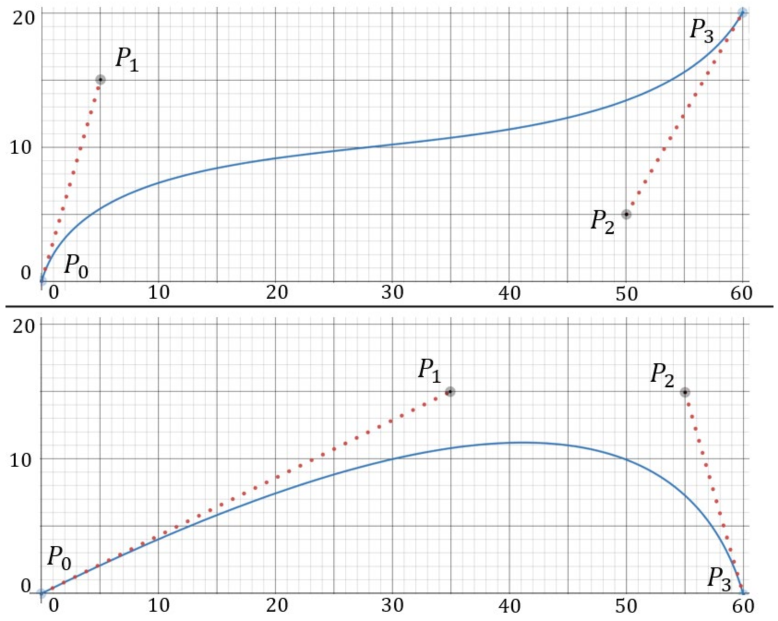


Figure 4- 2. Sample Cubic Bezier curves (blue) defined by four control points within a 60x20 domain

The material distribution within the design domain is assigned by defining the four control points of the Bezier curve, where each control point corresponds to a specific node in the domain. P_0 and P_3 can only be placed in one of the four corner nodes, precisely nodal coordinates (0,0), (60,0), (0,20), or (60,20), to ensure the material distribution results in a fully connected body when the domain is mirrored. P_0 and P_3 may not share the same nodal location. P_1 and P_2 can be placed at any nodal coordinates and may share a nodal location. The cubic Bezier curve can be produced once the four control points are defined. The design domain elements that intersect with the continuous line of the Bezier function are assigned material elements, while all other elements remain void. An additional thickness variable, T, has been introduced such that a thickness of 1 only assigns the directly intersected elements as material elements, while a thickness of 2 also assigns the intersected elements' immediate neighbors as material elements. A randomly generated design sequence for a unit cell is shown in Figure 4- 3.

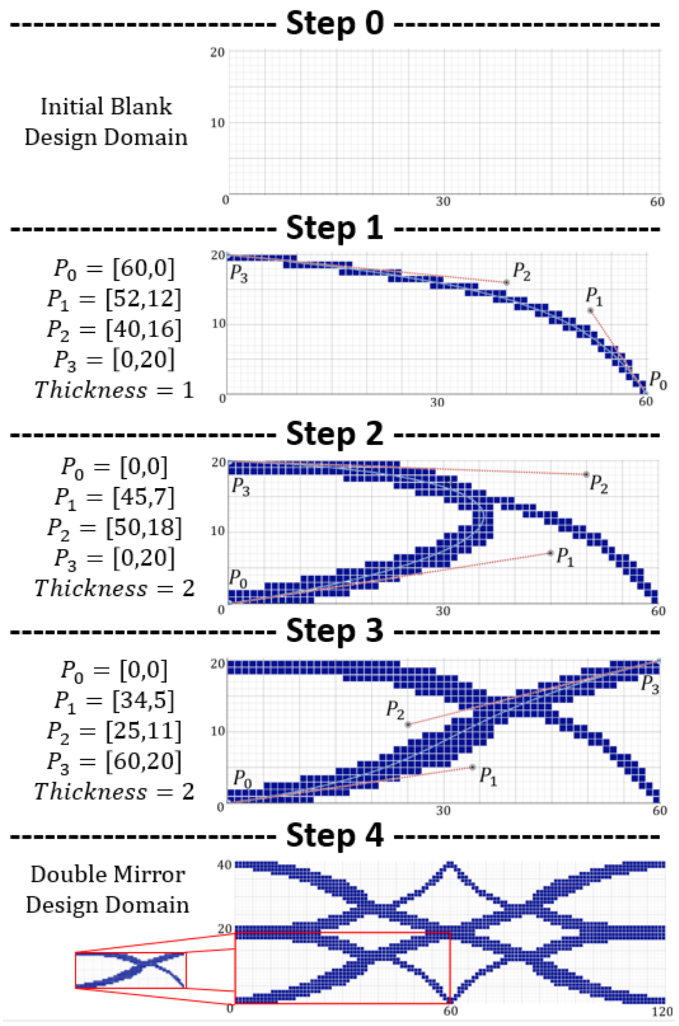


Figure 4- 3. Randomly generated metamaterial unit cell according to cubic Bezier curve material addition

The Cubic Bezier curve method is not the only approach for designing unit cells in this discretized domain. However, this method describes the unit cell as a series of continuous, seven-value arrays (starting/ending points, X/Y coordinates of intermediate points, thickness value). These array representations can be seamlessly integrated into the DRL environment, allowing the agent to directly control and modify the unit cell design by selecting and manipulating the array. Additionally, this method can produce more than 3.9×10^7 unique Bezier curves, ensuring the agent can thoroughly explore the unit cell's design domain without being limited by traditional geometric parametrization.

4.1.2 Tessellation into Metamaterial

The geometric nonlinearities of the unit cell define the deformation nonlinearities of a mechanical metamaterial. Therefore, once a base unit cell geometry is designed, the metamaterial is formed by tessellating the unit cell multiple times in the X and Y directions. The number of tessellations can vary depending on a user's application. However, the unit cell must undergo a sufficiently large number of tessellations to achieve homogenized behavior and minimize the effects of boundary conditions. Unfortunately, many unit cell designs must be tessellated 10-20 times before homogenization is met [62], [117]. As a result, designing, simulating, and optimizing the resulting metamaterials may be infeasible due to extreme computational costs.

The metamaterials were designed using a limited 3x3 tessellation of the base unit cell to reduce computational costs. The metamaterial uses a layer shift of half a unit cell length, allowing for a more uniform load distribution and a diverse range of nonlinear deformation responses [83]. Finally, the mesh is refined nine-fold to improve FEA accuracy. The tessellation process can be seen in Figure 4-4. This tessellation size does not guarantee a homogenized behavior. This unhomogenized assumption and the use of an arbitrary linear elastic material property show that the DRL agent is tasked with designing within a limited design domain.

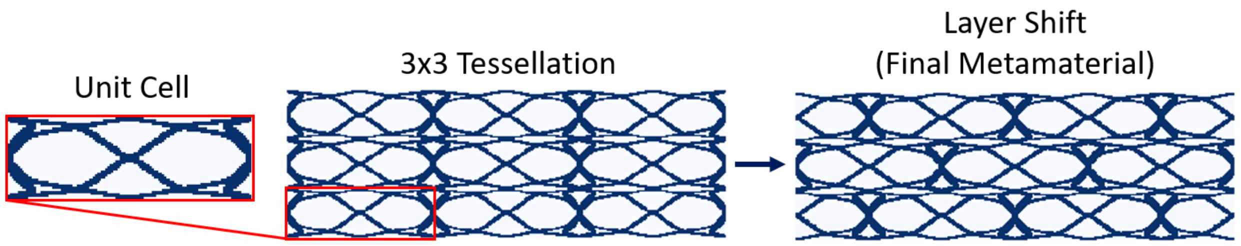


Figure 4- 4. Unit cell tessellation procedure to produce 3x3 metamaterial

4.1.3 Metamaterial Evaluation

The metamaterial's mechanical properties rely on the complex interactions between individual unit cells. The metamaterials are designed to achieve a target tensile or compressive nonlinear deformation behavior, which will differ from the constitutive material. Therefore, the effective deformation of the metamaterial must be determined. An FEA was performed to determine the force-displacement behavior of the metamaterials in compression or tension. The tensile and compressive force-displacement curves were determined by prescribing a 20% metastrain boundary condition on the top surface nodes of the metamaterial. Metastrain can be defined as the percentage of bulk deformation of the metamaterial [83], as seen in Equation 16:

$$Metastrain = \frac{\delta}{H}(100) \tag{16}$$

where δ is the top surface nodal displacement, and H is the original height of the metamaterial, as seen in Figure 4-5. Additional boundary conditions were applied to Y and X axes nodes to only allow vertical and horizontal displacement, respectively.

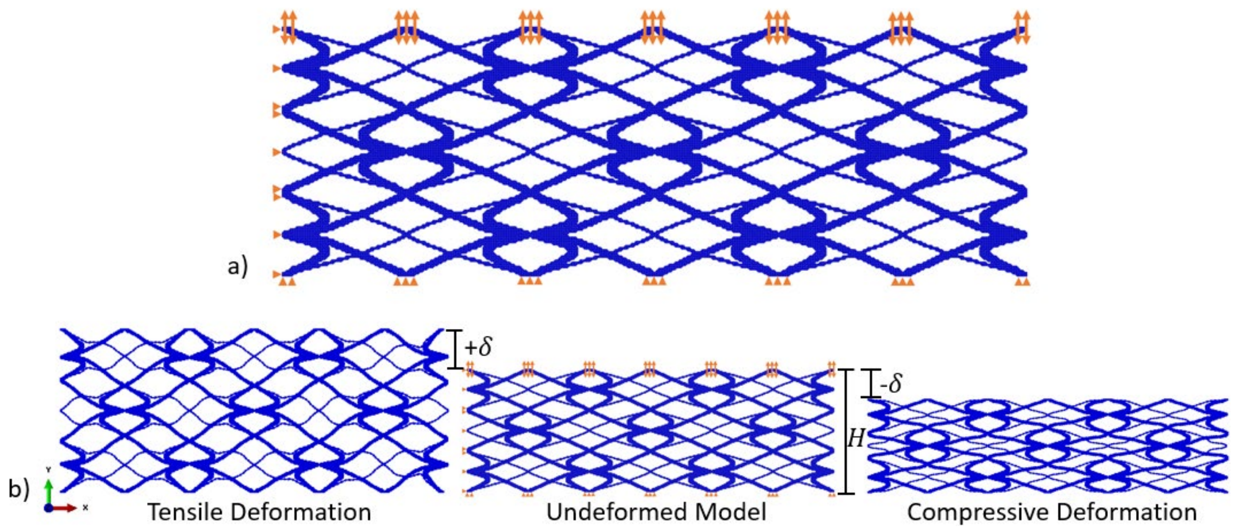


Figure 4- 5. Metamaterial FEA model with a) boundary conditions (orange) and b) tensile and compressive loading deformation

The FEAs were run in *Abaqus/CAE* as quasi-static evaluations using the reference material properties previously described, and internal contacts were not considered. Additionally, the potential for plastic deformation due to stress concentrations is not considered. However, the permanent distortion of plastic deformation should be considered in future work if the objective is to achieve targeted nonlinear deformation responses in cyclic loading.

The 20% metastrain boundary condition was applied in 11 equidistant steps, and the resultant vertical force of the top nodal layer of the metamaterial was recorded at each step. The force increases between successive displacement steps result in a nonlinear force-displacement curve. Applying the displacement in equidistant steps makes comparing the resultant force for different metamaterials at the same displacement value easier.

4.2 Methods: Metamaterial Unit Cell Design as a Deep Reinforcement Learning Problem

The cubic Bezier curve-based method for designing discrete unit cells needs to be reformulated as a sequential DRL task, precisely, a Markov Decision Process (MDP). Therefore, if the Bezier curve design approach can be expressed in terms of state space (S), action space (A), transition probability function (P), and reward function (R), the proposed metamaterial design method can be an MDP.

Designing grid-based metamaterial unit cells to achieve a desired nonlinear deformation response can be viewed as finding the material distribution that minimizes the objective function, F, of Equation 17. The material distribution is described by the density variable, ρ, which can take the value 0 (void) or 1 (solid) at any element, x, in the design domain, Ω. The mathematical form of this optimization problem is shown below:

$$\min_{\rho} F$$
$$F(\rho) = \sum_{i=1}^{11} \left(f_i^t - f\left(\int^{\Omega} u(\rho(x)) dx \right)_i \right) \tag{17}$$
$$s.t. \rho(x) = 0 \; or \; 1, \forall x \in \Omega$$

where f_i^t is the target resultant force at the *i-th* deformation step and $f\left(\int^{\Omega} u(\rho(x)) dx \right)_i$ is the resultant force of the current material distribution in the design domain, $\int^{\Omega} u(\rho(x)) dx$, at the *i-th* deformation step.

The details of the agent's interactions with the environment are in the following subsections, but a brief overview helps guide the description. First, the agent starts a design sequence with a wholly voided domain and receives information about the desired force-displacement curve. Given the material distribution and the desired force-displacement curve, the agent adds material to the design domain according to a cubic Bezier curve. The force-displacement response of that tessellated unit cell is determined, and the agent is rewarded or penalized depending on the difference between the desired force-displacement curve and the resulting curve of the designed unit cell. The agent uses this feedback to make intelligent decisions about designing unit cells according to different desired responses.

4.2.1 Encoding the Design Domain as a Low-Dimensional Latent Space

The 60x20 grid design domain is comprised of 1200 discrete variables. These variables represent the 1200 unique features needed to describe a unit cell. Unfortunately, DRL algorithms can be susceptible to the curse of dimensionality [118]. The curse of dimensionality states that as the number of features used to describe an input increases, the difficulty of the problem and the amount of data needed for sufficient learning exponentially increase [119]. Therefore, feature reduction for DML/DRL problems is a common area of research [120]–[122]. One of those methods is latent space representation using variational autoencoders.

Variational Autoencoders (VAEs) are an unsupervised learning technique that uses an encoder neural network architecture, g_ϕ, to develop a continuous compressed encoding, z, from input data, x. The encoding is trained by attempting to produce a reconstruction, x', by passing z through a decoder neural network architecture, f_θ, with a minimal difference to x. The low-dimensional encoding, traditionally called a latent space, forms a bottleneck in the middle of the VAE, forcing the encoder to extract the most critical features of the input and organize them into a continuous and meaningful embedding. Therefore, this latent space can be externally used as a reduced-order representation of the input state. Traditional autoencoders learn how to compress and reconstruct the original inputs by attempting to minimize the loss function, L, in Equation 18

$$L(\theta, \varphi) = \frac{1}{n}\sum_{i=1}^{n}\left(x^i - f_\theta\left(g_\phi(x^i)\right)\right)^2 \tag{18}$$

where x^i is the *i-th* variable of the input and $f_\theta\left(g_\phi(x^i)\right)$ is the *i-th* variable of the decoder's reconstruction. However, VAEs vary from traditional autoencoders because they do not directly form a latent space. Instead, VAEs output the parameters of a latent space distribution, i.e., the mean and standard deviation. This distribution helps regularize the latent space and provides data generation capabilities to the VAE. The VAEs have an additional loss term that encourages the latent space to have a target distribution (in this work, a Normal distribution):

$$\begin{gathered} reconstruction\ \ loss = L(\theta, \varphi) = \frac{1}{n}\sum_{i=1}^{n}\left(x^i - f_\theta\left(g_\phi(x^i)\right)\right)^2 \\ similarity\ \ loss = KL\ Divergence = D_{KL}\big(\mathcal{N}(\mu_x, \sigma_x) \parallel \mathcal{N}(0, \boldsymbol{I})\big) \\ loss = reconstruction\ \ loss + Z * similarity\ \ loss \end{gathered} \tag{19}$$

where $D_{KL}\big(\mathcal{N}(\mu_x, \sigma_x) \parallel \mathcal{N}(0, \boldsymbol{I})\big)$ measures the statistical distance between the current latent space distribution, $\mathcal{N}(\mu_x, \sigma_x)$, and a normal distribution, $\mathcal{N}(0, \boldsymbol{I})$ and Z is a constant scaling value ($Z = 3x10^{-3}$) to ensure a balanced contribution from the two loss terms.

The VAE was trained to define the 1200 variable design space of the unit cell as a 24-dimension latent space. This VAE is trained by passing the 60x20 design domain of a unit cell through the encoder deep neural network architecture. The outputs of the encoder network are the 24-dimension mean and standard deviation defining the latent space distribution. This distribution is sampled to produce a single 24-dimension latent space. The latent space is then passed through the decoder deep neural network layer to recreate the original 60x20 design domain. Using 24 dimensions to define the latent space achieves a balance between low dimensionality and generation quality. Based on an empirical study, a higher dimensionality provides a minuscule improvement (~1%) in reconstruction accuracy. The architecture details of both neural networks can be found in Figure 4- 6.

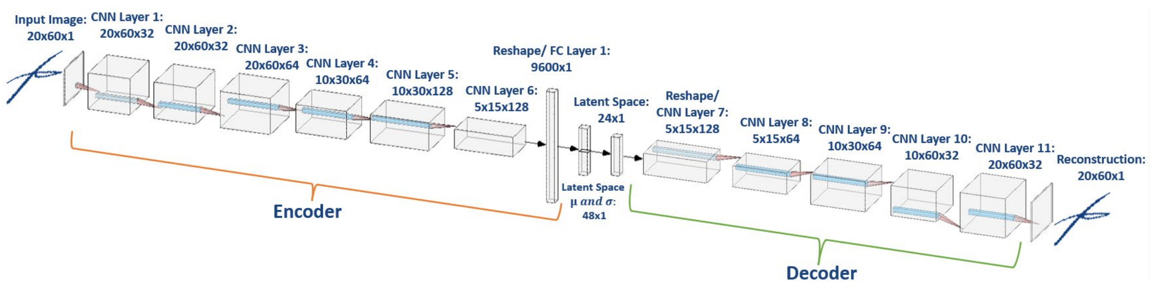


Figure 4- 6. Variational Autoencoder neural network architecture

The VAE was trained using a series of 100,000 60x20 design domains with material distributions defined by between one to six randomly generated cubic Bezier curve(s). Material distributions defined by more than six Bezier curves left few voided regions, which resulted in unit cells with undesirably stiff and linear deformation responses. The 100,000 samples underwent an 80/10/10 training/validation/testing split, and the VAE was trained for 50 epochs using the hyperparameters in Table 4-1.

Table 4- 1. Hyperparameters for Variational Autoencoder Training

Hyperparameter	**Value**
Learning Rate	$5x10^{-4}$
Batch Size	128
Training Epochs	50

4.2.2 Surrogate Model Prediction of Force-Displacement Responses

DRL algorithms are not sample efficient, and therefore, the agent must design and determine the deformation response of a large and diverse array of tessellated unit cells. The deformation response of these tessellated unit cells could be determined using FEA simulations, experimental methods, or, more efficiently, a DNN surrogate model. The surrogate model provides a rapid tool to predict force-displacement values for any unit cell design.

Predictive surrogate models are examples of supervised machine learning techniques, meaning they must be trained on a dataset of labeled data. For the current application, the input of this labeled data is the VAE's 24-dimensional latent space for a unit cell. The output is the 11 resultant force values corresponding to the force-displacement response of the tessellated unit cell.

The training data was collected by designing randomly generated unit cells, tessellating them into 3x3 metamaterials, and conducting an FEA analysis using *Abaqus/CAE*. The same FEA procedure described in Section 4.1.3 was implemented. Separate surrogate models were trained to predict tensile and compressive deformation responses. Five thousand random unit cells were designed and simulated to produce the training data for the tensile model. Ten-thousand-unit cells were designed and simulated for the compression model. The compressive model was trained on more unit cells because compressive deformation results in a broader spectrum of nonlinearities than tensile deformation [83], [123]. The unit cells were designed and simulated in batches of 100 using Clemson University's Palmetto Cluster.

The dataset was filtered to eliminate any unit cell whose FEA result did not converge to the 20% meta-strain in 11 steps. This filtering eliminates failed FEA runs and ensures all output samples have equivalent sizes. After discarding the failed unit cells, the compression and tension surrogate models were left with 6162 and 3321 samples per dataset, respectively.

Typically, datasets of such sizes would not be large enough to teach the surrogate model the underlying relationship between the unit cell and force values. Running more FEA simulations is one approach to producing more data points, but it is also computationally costly. An alternative is to incorporate data augmentation. Data augmentation is a method of generating novel training data for sparse datasets by applying strategic alterations to the current data. This process is typical in image recognition problems, as an existing image can be flipped, cropped, or rotated [124]. A new image is created, but the same labeled output from the original image is still used, resulting in a newly labeled data point. While this method is standard in image recognition problems, it cannot be applied to the unit cells because cropping or rotating the unit cells would alter its corresponding force-displacement response.

Alternatively, new labeled data points were created by sampling the latent space distributions produced by the 6162- and 3321-unit cells from the compression and tension datasets, respectively. Sampling the latent space distributions produces multiple latent space representations corresponding to a single input unit cell. It is well known that similar latent space representations are encoded from similar inputs and decode similar outputs [125]. In turn, these novel latent spaces correspond to nearly identical unit cell designs as the original unit cell, as seen in Figure 4- 7. Therefore, these latent space distributions could serve as new surrogate model input values with corresponding output values of the 11 force values from the original unit cell.

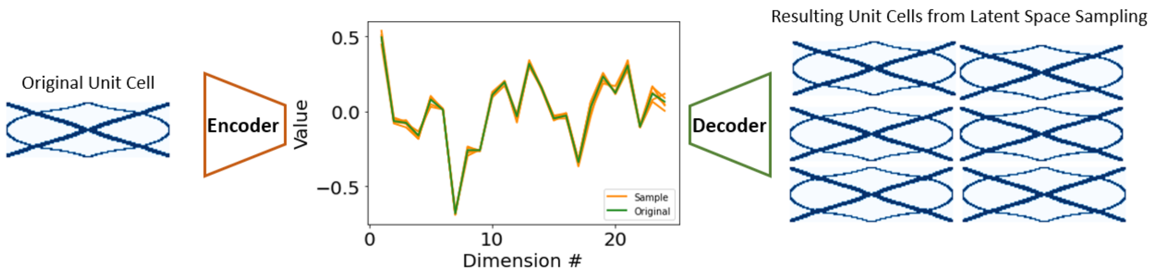


Figure 4- 7. The data augmentation procedure produces new input data to train the surrogate model

This data augmentation procedure was repeated to produce individual training sets of 100,000 samples for compression and tension separately. Producing the 100,000 samples took a few minutes. Alternatively, producing 100,000 samples for compression and tension through FEA simulations would be computationally challenging.

The surrogate model architecture was a multi-layer perceptron, with the 24-dimension latent space of each unit cell as the input and 11 force values as the output. Before training, the force values were normalized. The architecture and training hyperparameters for the tension and compression surrogate models were the same and can be found in Figure 4-8 and Table 4-2, respectively.

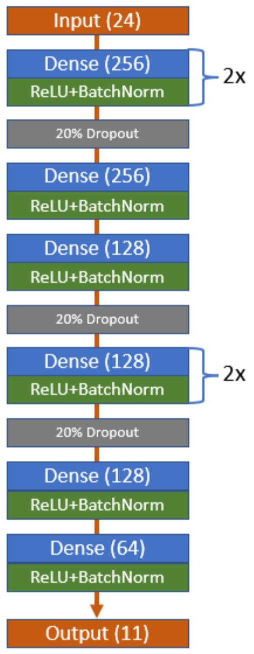


Figure 4-8. Surrogate model architecture where the input is the 24-dimensional latent space of a unit cell, and the output is the 11 normalized force values

Table 4- 2. Hyperparameters of Surrogate Model Training

Hyperparameter	Value
Learning Rate Initial Value	5x10^{-3}
Learning Rate Decay Rate	0.9
Batch Size	128
Training Epochs	150

The models were trained for 150 epochs using an 80/10/10 training/validation/testing split and a learning rate scheduler with an initial learning rate of $5x10^{-3}$ and decay rate of 0.9 at each epoch. The models were trained using Clemon University's Palmetto Cluster. Both models were trained to minimize a mean squared error loss function, as shown in Equation 20:

$$Loss = \frac{1}{11}\sum_{i=1}^{11}\left(Y_i - \widehat{Y_\iota}\left(g_\phi\left(\int^{\Omega} u(\rho(x))dx\right)\right)\right)^2 \tag{20}$$

where, Y_i is the *i-th* actual force value and $\widehat{Y_\iota}$ is the *i-th* predicted force value according to the latent space of the unit cell material distribution, $g_\phi\left(\int^{\Omega} u(\rho(x))dx\right)$. The training and testing results are shown in a later section. The successfully trained surrogate model could now be used as a rapid inline tool for the DRL agent to determine the performance of its proposed unit cell designs.

4.2.3 DRL Environment: State Space

The state space, S, of a DRL environment represents all the combinations of observations an agent can experience while interacting with the environment. For the unit cell design problem, each observation should capture information about the design objective (desired force-displacement curves) and the agent's location in the design domain (unit cell). Therefore, the observations were encoded as a 35x1 vector constructed from a standardized version of the 11 force values corresponding to the desired force-displacement curve and the 24-dimension latent space of the current material distribution (Figure 4-9). The 11 desired force values are standardized according to the compressive or tensile dataset, following Equation 21.

$$\overline{\overline{Y_\iota}} = \frac{Y_i - u_i}{\sigma_i}$$

$$u_{i-C} = \frac{\sum_{i=1}^{6162} Y_i}{6162} \qquad u_{i-T} = \frac{\sum_{i=1}^{3321} Y_i}{3321} \tag{21}$$

$$\sigma_{i-C} = \sqrt{\frac{\sum_{i=1}^{6162}(Y_i - \mu_{i-C})^2}{6162}} \qquad \sigma_{i-T} = \sqrt{\frac{\sum_{i=1}^{3321}(Y_i - \mu_{i-T})^2}{3321}}$$

where $\overline{\overline{Y_i}}$ is the standardized *i-th* force value and Y_i is the *i-th* original force value from the compression (C) or tension (T) dataset. Throughout a single design episode, the 11 desired force values will remain constant, but the 24-dimension latent space values will change as the DRL agent sequentially alters the material distribution.

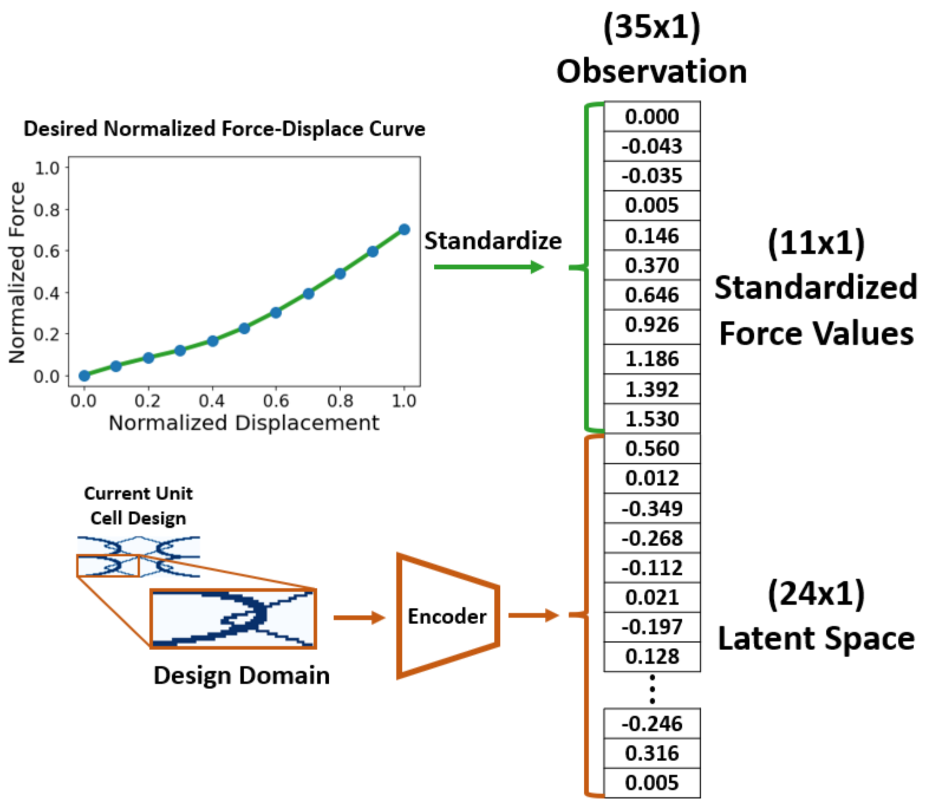


Figure 4- 9. Individual observation within the unit cell designing environment state space

4.2.4 DRL Environment: Action Space

The action space defines how an agent can interact with an environment, taking the agent from its current observation to the next. Within the unit cell designing environment, an action corresponds to defining the cubic Bezier curve that will add material elements to the design domain. A seven-dimension action space was created that allows the agent to select the starting and ending control points, the nodal coordinates for the two intermediate control points, and the thickness of the Bezier curve. This approach results in $1.31\text{x}10^7$ unique Bezier curve combinations. Producing a discrete action space that accounts for all these curves would be infeasible. Therefore, a combination of continuous and discrete action elements was used.

The action space was tailored to assist the agent in creating fully-connected and continuous unit cell designs. Therefore, the starting control point, P_0, was restricted to either the bottom left (0,0) or bottom right (60,0) nodes of the design domain and the final control point, P_3, was restricted to either the top left (0,20) or the top right (60,20) nodes of the design domain. The first entity in the action space is a discrete action element where a value of 0 sets the nodal coordinates of P_0 to (0,0), and a value of 1 sets them to (60,0). The second action entity is also a discrete element where a value of 0 sets the nodal coordinates of P_3 to (0,20), and a value of 1 sets them to (60,20). The third discrete action element corresponds to the additional curve thickness (as described in Section 4.1.1).

The X and Y coordinates for the two intermediate control points, P_1 and P_2, are represented in the action space using continuous values between [0,1]. The action value between [0,1] corresponds to the proportional coordinate value between [0,60] or [0,20] for the X and Y coordinate, respectively. The seven elements in the action space allow the agent to alter the material distribution within the design domain, as seen in Figure 4-10.

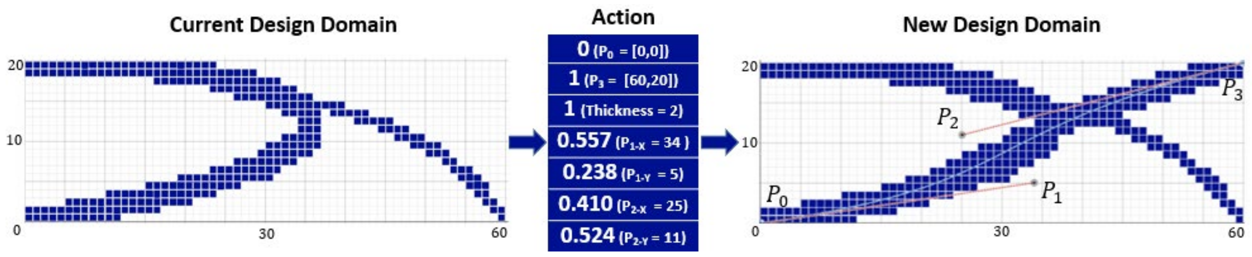


Figure 4- 10. Augmenting Design Domain using action from DRL action space

4.2.5 DRL Environment: Reward Function

In the presented environment, the reward function should enable the agent to design metamaterial unit cells with a resulting force-displacement response like the desired response. A negative reward function was formulated that penalizes the agent according to the maximum percent error (MPE) between the desired and resulting force values. Depending on a user's requirements, the desired force values can be arbitrarily derived. The resulting force values are determined using the trained surrogate model. The agent can continue taking actions until the MPE is less than 10%, at which time the agent will receive a positive reward of 1 minus the MPE. A design episode will also be terminated if the agent has taken more than seven actions in a single episode. To avoid exploding gradients, the maximum penalty the agent can receive for a high percent error is -1. Therefore, if the MPE between the desired and resulting force values is greater than 100%, the agent will only be penalized -1.

A comparison between resulting and desired force values can only be made if the agent has produced a legal unit cell design. A legal design is a 60x20 design domain that is a single material body with material elements in the four corner nodes ([0,0], [0,20], [60,0], [60,20]). Legal designs ensure a continuous body when the design domain is mirrored into a unit cell. If the DRL agent takes an action that results in an illegal design, the agent is penalized -1. This penalty should teach the agent to avoid actions leading to illegal designs. The exception to this rule is during the first action. The agent cannot produce a legal design with a single Bezier curve, as connecting all four corner nodes is impossible. Therefore, on the first action, the agent is always given a reward of 0. The entire reward function, r_t, at a given timestep, t, can be seen in Equation 22.

$$r_t = \begin{cases} 0 & if\ t = 0 \\ -1 & if\ t \neq 0\ and\ Illegal\ Design \\ max\left[-1, -\max_i\left(\frac{\hat{f}_i - f_i}{\hat{f}_i}\right)\ for\ \ i = 1,2\ldots 11\right] & if\ Legal\ Design\ and\ MPE > 0.1 \\ 1 - MPE & if\ Legal\ Design\ and\ MPE \leq 0.1 \end{cases} \tag{22}$$

where f_i and $\hat{f}_i$ are the current and desired *i-th* force values, respectively. While this proposed reward cannot be proven as optimal or the only viable approach, the effectiveness of this reward function is demonstrated by favorable results in the following sections. Alternative methods may lead to comparable results, such as a sparse reward function where the agent is only rewarded at the end of the design episode or measuring the absolute error instead of the percent error.

4.2.6 DRL Environment: Transition Probability Function

I implemented the deep deterministic policy gradient algorithm (DDPG) to account for the continuous state and action spaces. DDPG is an actor-critic method with four deep neural networks that concurrently learn a Q-function and a policy [126]. The policy network, μ, behaves as an actor to select an action from the action space according to the current state. The value network, Q, behaves as a critic and predicts the value of the actor's action, and gives feedback for improvement. Additional target networks, Q' and u' are used to track the original Q and μ networks to mitigate the effect of value overestimations and outdated policies, respectively.

The action of the actor, a_t, is determined by the current policy and the introduction of exploratory noise, given by

$$a_t = \mu(s_t|\theta^{\mu}) + \epsilon \tag{23}$$

where s_t is the current state, θ^{μ} are the weights and biases of μ, and ϵ is Gaussian noise. The value of a given policy is calculated according to the Bellman optimality equation below:

$$Q^*(s_t, a_t) = E\left[r(s_t, a_t) + \gamma \underset{a_t}{\operatorname{argmax}}\big(Q^*(s_{t+1}, a_{t+1})\big)\right] \tag{24}$$

where $E(\cdot)$ denotes the expectation operator, Q^*denotes the optimal value function, r is the reward for taking a_t given s_t and γ is the discount factor. The value network can be iteratively updated using previous experiences stored in a replay buffer by minimizing the following loss function:

$$L_Q(t|\theta^Q) = \left[r(s_t, a_t) + \gamma Q'\big(s_{t+1}a_{t+1}|\theta^{Q'}\big) - Q(s_t, a_t|\theta^Q)\right]^2 \tag{25}$$

$$a_{t+1} = \mu'\big(s_t|\theta^{\mu'}\big) \tag{26}$$

where $Q'\big(s_{t+1}a_{t+1}|\theta^{Q'}\big)$ is the predicted value according to the target value network and $Q(s_t, a_t|\theta^Q)$ is the actual value according to the value network. Given Equation 25, the gradient-descent method can be performed to improve the prediction of a policy's value.

The actor-network attempts to adjust its policy to select actions with high Q-value feedback from the critic network. Therefore, the performance objective of the policy network, ϕ, can be defined as

$$\phi(\theta_\mu) = E\big[-Q\big(s_t, \mu(s_t)\big)\big] \tag{27}$$

The policy network continues updating to minimize Equation 27. Therefore, the updating error can be expressed as the gradient of Equation 27, $\nabla\phi\big(\theta_\mu\big)$. The target networks, Q' and μ', are updated every N steps using the following soft updating strategy.

$$\begin{gathered} \theta^{Q'} = \tau\theta^Q + (1-\tau)\theta^{Q'} \\ \theta^{\mu'} = \tau\theta^\mu + (1-\tau)\theta^{u'} \end{gathered} \tag{28}$$

where τ is the network updating hyperparameter.

I do not claim that DDPG is this application's only or necessarily optimal algorithm. This algorithm was selected due to its prevalence and ability to account for continuous state and action spaces. However, DDPG can suffer from instability during training due to target networks, which can lead to slow convergence, commonly to a non-global optimal. Additionally, DDPG requires considerable training data and is highly sensitive to the choice of hyperparameters [127]. A comparison between various DRL algorithms is beyond the scope of this work but should be investigated as future work to further validate DRL as a powerful tool for mechanical metamaterial design.

4.3 Methods: Training the DRL Agent

The previous sections described the metamaterial design problem as an MDP; therefore, a DRL agent should be able to learn to design mechanical metamaterials with targeted nonlinear deformation responses. Therefore, the agent needs to undergo episodic interactions with the proposed environment to collect experiences and update the weights of its actor and critic networks for better action selection and value approximations. An episode in this environment is defined as the DRL agent sequentially adding material to the design domain according to cubic Bezier curves until the resulting force-displacement response is sufficiently close to the desired force-displacement response or the agent takes too many steps.

Separate agents were trained for compressive and tensile loading responses, but both agents underwent nearly identical training procedures. The agents were trained to design unit cells given a diverse range of desired force-displacement responses. Therefore, at the beginning of each training episode, one of the compression or tension force-displacement curves used to train the surrogate model was randomly selected as the desired force-displacement curve. The 6162 compression and 3321 tension curves underwent a 90/10 training/testing split. During training, the agents are prompted with one of the 5546 compressive or 2989 tensile desired responses and sequentially add material to the design domain to produce unit cells that exhibit similar responses as the desired response.

As previously stated, a fundamental dilemma of DRL training is the tradeoff between exploration and exploitation. The agent explores the environment by adding noise to its continuous action space to compile diverse experiences. However, to ensure efficiency, the agent should not strictly rely on exploration but instead exploit the value mapping it has already learned from its actor and critic networks. Therefore, to balance proper exploration and exploitation, action space noise should be more frequently added when the environmental understanding is poor at the beginning of training. As training continues and the performance of the actor and critic networks improves, the action space noise should be reduced, and heavier reliance should be placed on the actions selected by the actor.

The noise was added to the 7x1 action vector by randomly sampling from a normal distribution with a mean, μ, of 0 and a standard deviation, σ, that decays according to Equation 29.

$$\sigma = max\left[\sigma_{max} - N \cdot \sigma_{decay}, \sigma_{min}\right] \tag{29}$$

where σ_{max} is the starting action noise ($\sigma_{max} = 0.3$), σ_{decay} is the rate at which σ decreases per action ($\sigma_{decay} = 7.5x10^{-6}$) until σ reaches its minimal allowed value, σ_{min} ($\sigma_{min} = 0.005$), and N is the current episode number. Optimizing the tradeoff between exploration and exploitation is an active area of DRL research [128] beyond this paper's scope.

The action and added noise define the cubic Bezier curve that alters the material distribution. The new material distribution is passed to the VAE encoder to produce a new observation. Additionally, the surrogate model predicts the new unit cell's force-displacement response, which is compared to the desired response, and a reward is calculated. After each action, the previous observation, new observation, action, and reward are recorded in a replay buffer. Once an episode is complete and the replay buffer includes at least 3500 experiences, a batch of 256 experiences is randomly selected to train the DDPG actor and critic networks. The architecture of the actor and critic networks and other training hyperparameters can be found in Figure 4-11 and Table 4-3, respectively.

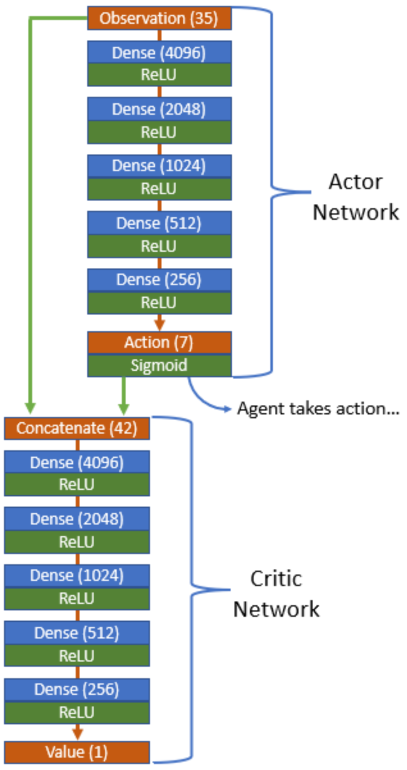


Figure 4- 11. Neural network architectures for Deep Deterministic Policy Gradient Method actor and critic networks

Table 4- 3. Deep reinforcement learning environment training hyperparameters

Hyperparameter	**Value**
Learning Rate	$2.5\text{x}10^{-4}$
Tau	$5\text{x}10^{-3}$
Batch Size	256
Max Replay Buffer Size	20,000
Gamma	0.99
Training Epochs	20,000

The main actor and critic model weights, $\theta^{\mu}$and θ^{Q}, are updated using the ADAM optimizer with a $2.5\text{x}10^{-5}$ learning rate. The target actor and critic model weights, $\theta^{\mu'}$and $\theta^{Q'}$ are updated according to Equation 30:

$$\theta_{t+1}^{\mu'} = \theta_t^{\mu'} \cdot (1-\tau) + \theta_t^{\mu} \cdot \tau$$
$$\theta_{t+1}^{Q'} = \theta_t^{Q'} \cdot (1-\tau) + \theta_t^{Q} \cdot \tau \tag{30}$$

where $\theta_{t+1}^{\mu'}$ and $\theta_{t+1}^{Q'}$ are the new target actor and critic model weights, $\theta_t^{\mu'}$ and $\theta_t^{Q'}$ are the current target actor and critic model weights, θ_t^{μ} and θ_t^{Q} are the current actor and critic model weights, and τ is a learning rate constant $5\text{x}10^{-3}$. As previously stated, DDPG is highly sensitive to hyperparameter selection. Therefore, the aforementioned hyperparameters (learning rates, replay buffer size, noise decay, etc.) underwent a Bayes optimization to determine the hyperparameter combinations resulting in high-performing agents.

4.4 Results

4.4.1 Variational Autoencoder Results

The trained VAE was tested using the 10,000 samples from the testing dataset. The VAE was not trained on these samples, and therefore, the testing dataset can evaluate the VAE's ability to generalize and reconstruct material distributions it has not previously seen. The output of the VAE is not a direct reconstruction of the material distribution. Instead, the VAE outputs the probability that each element is a material element. Therefore, probability values greater or equal to 0.5 are rounded to 1 (solid), and values less than 0.5 are rounded to 0 (void) to produce an elementally discrete material distribution.

The VAE had a mean absolute reconstruction error of 4.4%, meaning an average of 52 of the 1200 design elements are misrepresented. A series of sample reconstructions in Figure 4-12 show that the original and reconstructed unit cells are visually similar. The reconstruction process takes less than a second, with the decoding traditionally taking longer than the encoding. Therefore, the 24-dimension latent space finds a proper balance between reconstruction accuracy, efficiency, and low dimensionality and can be implemented as the input for the surrogate model and part of the observation of the DRL environment.

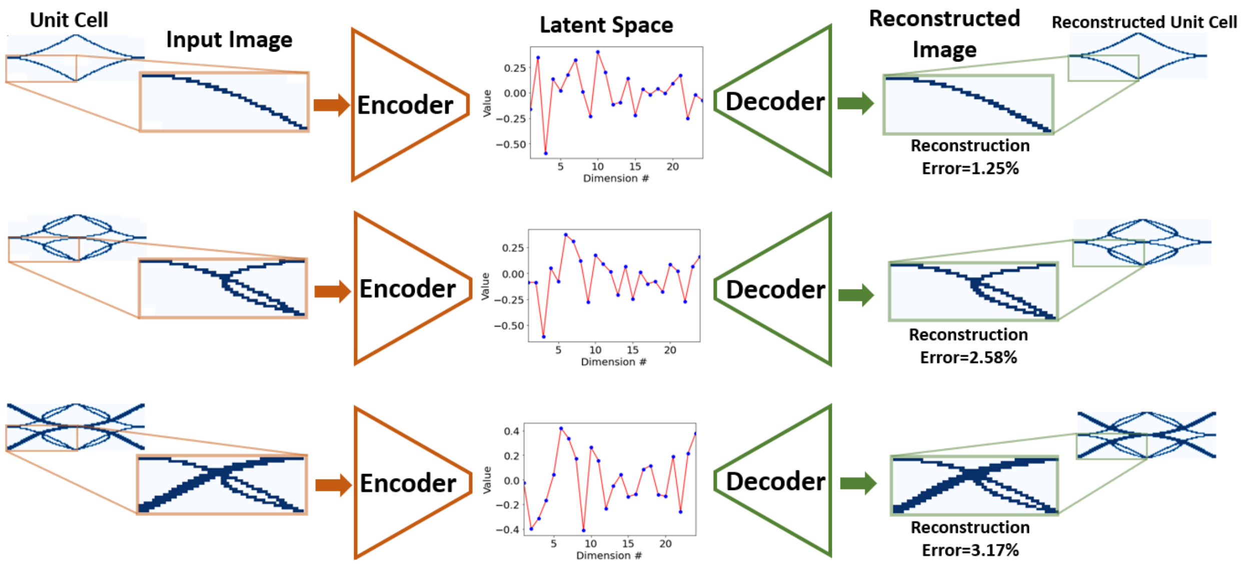


Figure 4- 12. Sample VAE reconstructions from the testing dataset

4.4.2 Results: Surrogate Model Results

The trained compressive and tensile surrogate models were tested using the 10,000 samples from the testing dataset. The surrogate models' performances were measured by the mean absolute percent error (MAPE) between the true and predicted force-displacement curves. The testing set for the compressive surrogate model included a diverse range of force-displacement responses, including curves that exhibit hardening and softening nonlinearity and other curves with high linearity. The tensile testing set had a limited range of nonlinearity. Figure 4-13 shows a scatter plot comparing the true and predicted force values for all 10,000 testing samples within the compressive and tensile datasets. These predictions were completed 60x faster than if they were run in *Abaqus*.

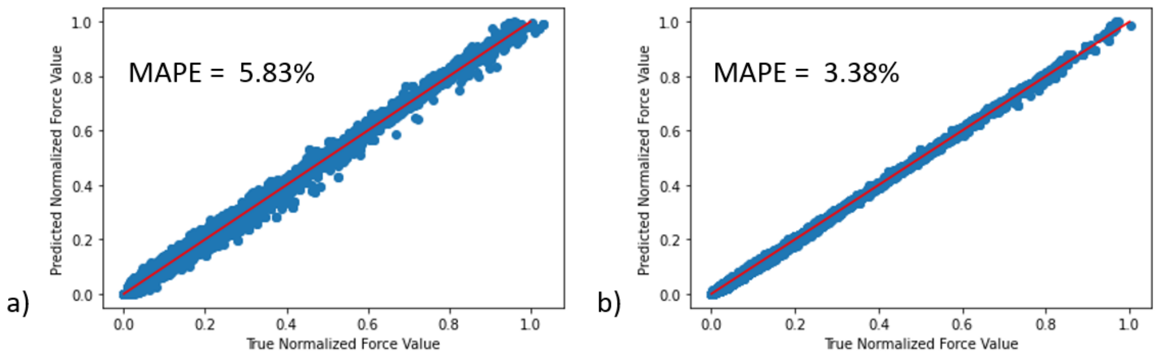


Figure 4- 13. Scatter plot comparison of the true and predicted training set force values a) Compression and b) Tension models

Figures 4-14a,b, and c plot a small sample of the 10,000 responses to show that regardless of hardening or softening nonlinearity or a highly linear response, the compressive surrogate model can accurately predict the force-displacement response of a tessellated unit cell. Figure 4-14d shows the tensile surrogate model equivalent. These figures include the corresponding unit cell that exhibits the true force-displacement response. These results validate that the surrogate model can act as a rapidly accurate force-displacement prediction tool for the DRL agent.

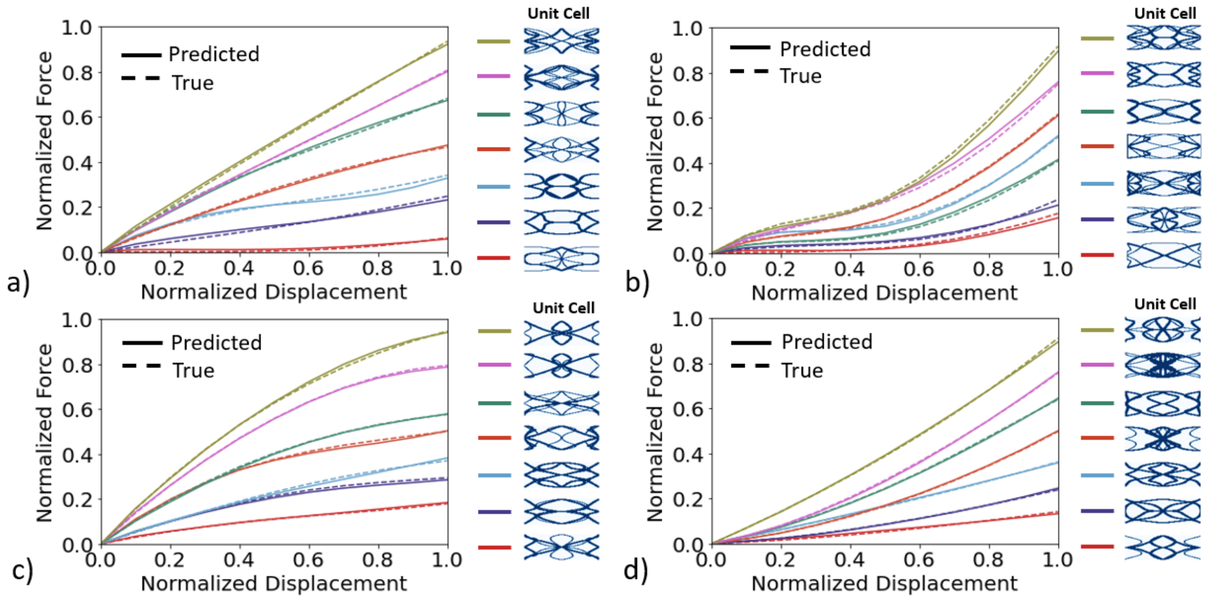


Figure 4- 14. Comparison of the True vs. Predicted force-displacement curves for a) Compressive Highly Linear, b) Compressive Hardening Nonlinear, c) Compressive Softening Nonlinear, d) Tensile responses

4.4.3 DRL Training Results

The tension and compressive DRL agents were trained separately for 20,000 design episodes using Clemson University's Palmetto Cluster. Figure 4-15 shows a moving average of the training reward for both the compressive and tensile models. The average reward increased until convergence around 0.3 and -0.1 for the tensile and compressive models, respectively. This increase in reward coincides with a decrease in the action space noise that decays to its minimum value after ~6300 episodes. The increase in average reward indicates that the agents are designing unit cells that exhibit force-displacement responses similar to the desired responses. Additionally, the average rewards converging around 0 suggests the additional positive reward for achieving less than 10% MPE is being implemented. As previously stated, DDPG tends to converge to locally optimal solutions. Therefore, the authors cannot claim that the policies of these agents are necessarily globally optimal. However, the results in the following section show that the agents have learned a sufficiently accurate and generalizable design strategy.

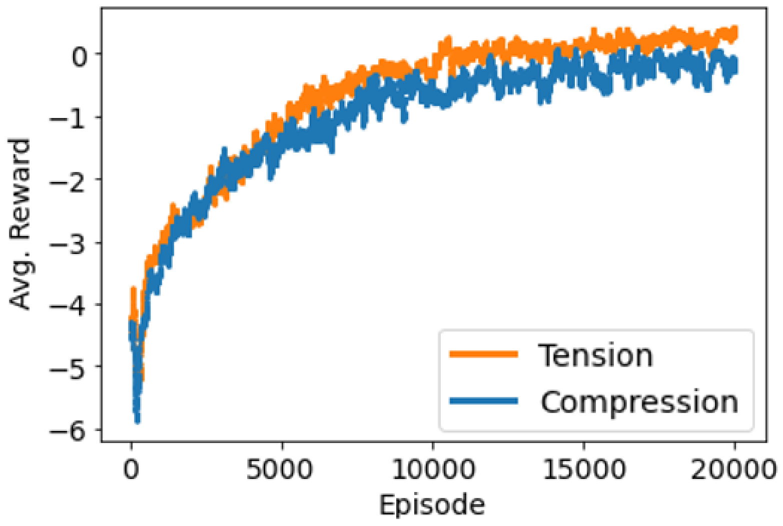


Figure 4-15. The average reward for the tensile and compressive models during training

4.4.4 DRL Testing Results

The trained DRL agents were tested using the 616 and 333 testing force-displacement curves for the compressive and tensile agents, respectively. Each test case introduces a unique desired force-displacement curve that the agents had not seen during training. Testing the DRL agents on previously unseen experiences examines the agents' abilities to learn a generalized design strategy for designing unit cells and ensures the agents have not overfitted to previously seen experiences. In addition, proof of generalization ensures the agents can serve as practical design tools capable of accurately designing for a wide array of user-specified deformation responses.

Figure 4-16 shows an example of design sequences for the compressive and tensile DRL agents. The first actions by the agents do not yield a feasible unit cell design, as it is impossible to connect all four corners of the design domain using a single cubic Bezier curve. By the second action, the agents have successfully designed legal unit cells, but the corresponding responses are far from the desired response. Therefore, the agent takes additional steps to satisfy the maximum percent error threshold.

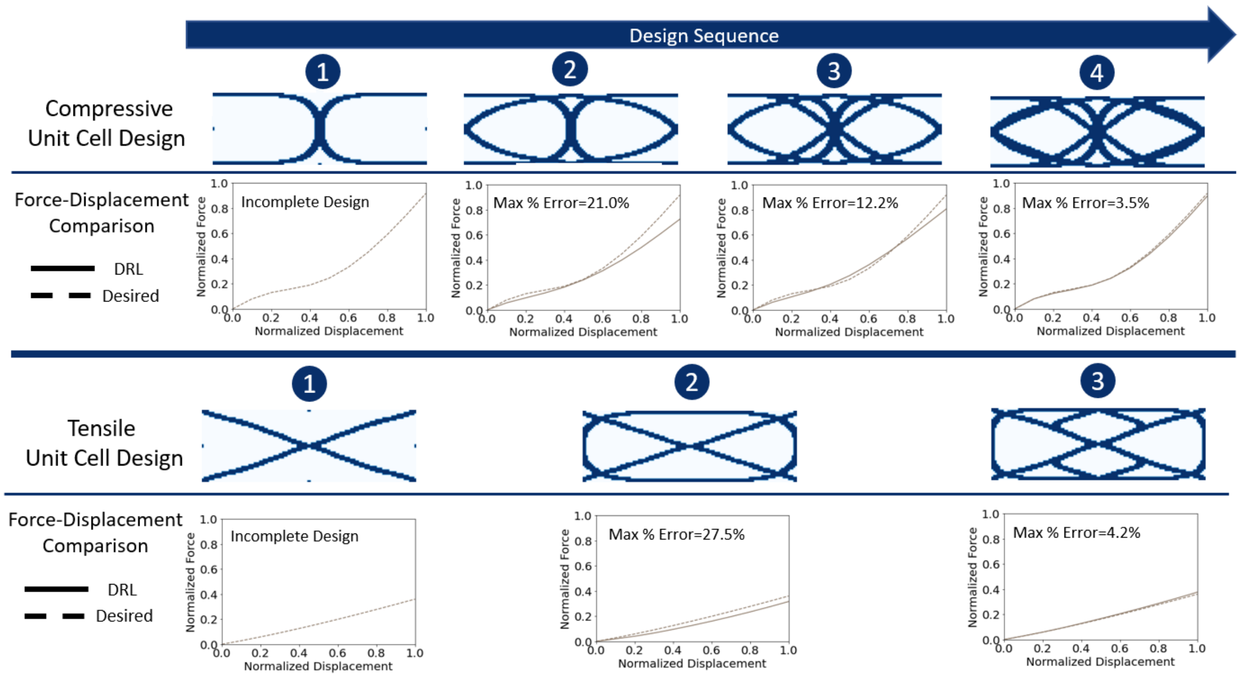


Figure 4-16. Design sequence for the compressive and tensile DRL models

The generalized design strategy shown in Figure 4-16 is expected to be applied to design unit cells for a wide range of force-displacement responses. Therefore, the compressive and tensile agents were tested on all samples from their respective test sets. Similar to testing the surrogate model, the test cases for the compressive model included hardening and softening nonlinear and highly linear force-displacement curves. The tensile testing set had a more limited range of nonlinearity.

Each test design sequence took between 1-3 seconds. Table 4-4 shows the design accuracy of both agents. The table includes both max and mean absolute percent error over the samples. While the agents were trained to minimize the maximum percent error, this measure can be artificially inflated given a single outlier force value. While achieving high similarity between all 11 force values is important, mean absolute percent error can better show how closely the unit cell behaves compared to the entire desired deformation response. Figure 4-17 shows a sample comparison between the DRL resulting and desired force-displacement curves. These comparisons are accompanied by the corresponding unit cell that the DRL agent has designed.

Table 4- 4. Testing accuracy for compressive and tensile DRL models

	Compressive DRL Model	Tensile DRL Model
Average Max Percent Error	13.1%	8.3%
Mean Absolute Percent Error	8.7%	5.3%

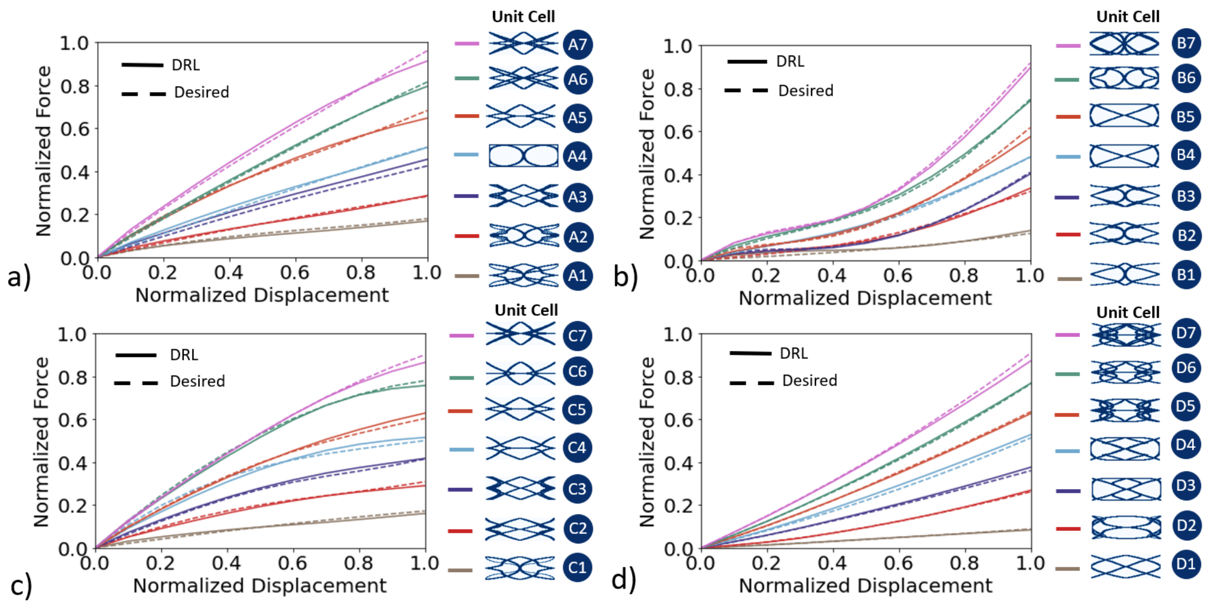


Figure 4-17. Comparison of the resulting DRL to Desired force-displacement curves for a) Compressive Highly Linear, b) Compressive Hardening Nonlinear, c) Compressive Softening Nonlinear, d) Tensile responses

Table 4-4 and Figure 4-17 show that the compressive and tensile DRL agents have learned a generalized strategy for designing mechanical metamaterial unit cells that exhibit force-displacement responses similar to a diverse range of desired responses.

4.5 Discussion

4.5.1 Measuring Acceptable Levels of Nonlinearity

To further understand the design capabilities of the agents, I measured the levels of desired force-displacement dimensional nonlinearity that the agents could accurately achieve. Dimensional nonlinearity is the root mean squared deviation of the response curve from a straight line [129]. The straight line defining the force-displacement curve would be a linear line connecting the curve's first and last force values. The absolute dimensional nonlinearity was calculated for all desired responses in the compressive and tensile testing sets and compared to the MAPE between the desired and resulting force-displacement responses, as shown in Figure 4-18.

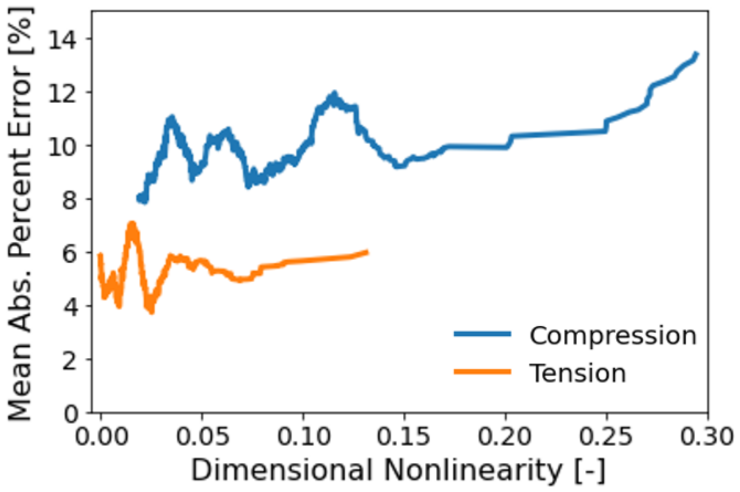


Figure 4-18. Comparison of the mean absolute percent error and the dimensional nonlinearity in the domain

Figure 4-18 further validates that the nonlinearity of the compressive testing set is more significant than the tensile set. Additionally, this figure shows that the tensile testing set has a small enough degree of nonlinearity to avoid a significant increase in MAPE. The small error peak in the tension results around an absolute dimensional nonlinearity of 0.02 is most likely the result of a handful of outlier force-displacement responses that yielded an unexpectedly high MAPE, artificially increasing the rolling average. Therefore, the tensile DRL model should be considered sufficiently accurate for force-displacement curves with an absolute dimensional nonlinearity less than 0.125.

The compressive results show a consistent increase in MAPE once the absolute dimensional nonlinearity is greater than 0.25. The three small peaks with absolute dimensional nonlinearities of less than 0.15 can be attributed to outliers. Additionally, there are more samples from the compressive testing set with dimensional nonlinearities between [0.025,0.15], so poor-performing outlier examples are expected. Therefore, the compressive DRL model should be considered sufficiently accurate for desired force-displacement curves with an absolute dimensional nonlinearity less than 0.25.

4.5.2 Negative Stiffness Designs

The previous section showed the DRL agents' ability to successfully design unit cells for diverse nonlinearities. Next, the compressive agent was tested on a more challenging set of desired force-displacement responses, precisely responses that exhibit negative stiffness. Negative stiffness is indicative of mechanical buckling within and between the unit cells. Buckling occurs as an instability when a structure can no longer support the current compressive load level. This instability can be challenging to capture using a linear elastic, quasi-static FEA solver like the one used in this analysis. Therefore, the authors hypothesized that very few simulated metamaterial designs would exhibit this behavior in compression.

Figure 4-19a shows a sample of unit cell designs exhibiting negative stiffness in compression. The surrogate model can provide a sufficiently accurate prediction of this negative stiffness. However, Figure 4-19b shows that the DRL agent has difficulty producing unit cells that exhibit the desired negative stiffness. While the DRL agent cannot capture the negative stiffness, it does appear to produce designs that exhibit responses similar to a first-order nonlinear variation of the desired curve. The DRL agent's inability to accurately capture the negative stiffness is accredited to the small number of training force-displacement curves that exhibited such behavior, roughly 8%. Therefore, the agent's policy is not geared toward these responses. As a result, the agent still attempts to minimize the maximum percent error by producing simple unit cell designs whose responses seem to run through the inflection point of the desired curves.

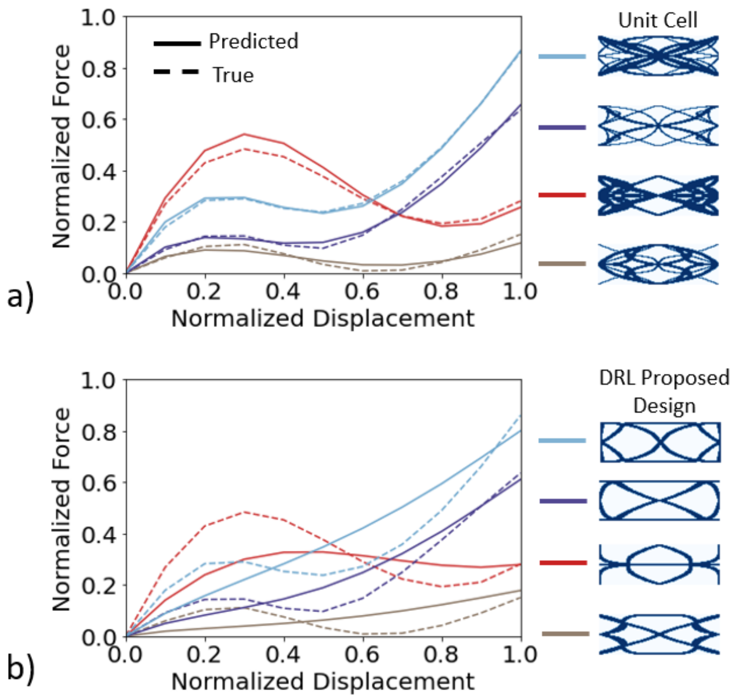


Figure 4-19. a) Surrogate Model prediction of unit cells with negative stiffness force-displacement curves b) DRL Compression model proposed designs

The simplicity of the resulting DRL designs is accredited to the agent attempting to minimize the penalty it accumulates for taking multiple design steps. The original unit cells have a complex material distribution that could only be achieved if the DRL agent took several steps, continually accumulating negative rewards. This issue is an inherent limitation of the proposed negative reward function of the DRL. A positive reward function would not yield the same limitation and is worth investigating in future work. Due to these limitations of the agent, the authors do not recommend testing the proposed DRL method on desired responses that exhibit negative stiffness.

4.5.3 Advantages of DRL Design Method

The work in this chapter takes a step towards demonstrating that DRL can be incorporated as a high-level co-designer working with a human designer to achieve high-performing design solutions in a wide range of engineering applications. This work validates this claim specifically for designing mechanical metamaterials with targeted nonlinear deformation responses. I wanted to compare the performance of the DRL method to the more traditional design methods of unit cell synthesis and TO. Due to variability between the design domains, a direct comparison between the three methods under the same deformation response was deemed infeasible. However, the DRL method was compared against the tensile response of the TO method proposed by Behrou *et al.* [86] and the compressive response of Satterfield *et al.* [83] unit cell synthesis method.

Figure 4-20 compares the DRL proposed unit cell design to the *approximate* unit cell designs from the two publications and the MAPE between the resulting and desired responses. The reader is directed to the respective publications for a more thorough analysis of the resulting designs and tessellated patterns. Due to variability between the three methods regarding boundary conditions, design domains, aspect ratios, and tessellation sizes, the desired curves for the DRL method were approximated to capture the degree of nonlinearity presented in [86] and [83] but with an arbitrary magnitude.

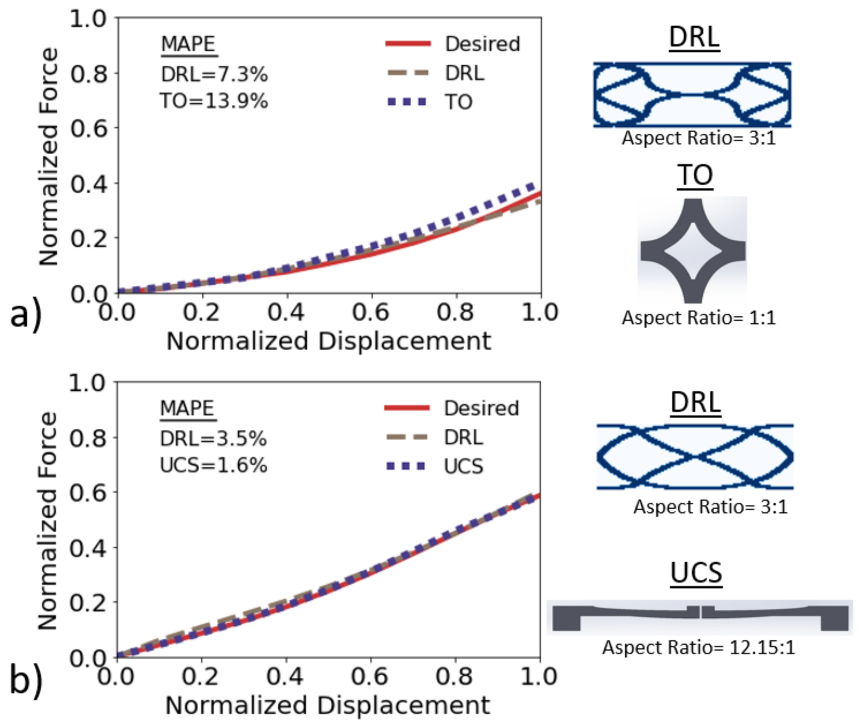


Figure 4-20. Design result comparison between DRL method to a) topology optimization (TO) method proposed for tensile responses by Behrou *et al.* and b) the unit cell synthesis method for compressive responses by Satterfield *et al.*

Figure 4-20a shows that the DRL proposed design achieves better agreement with the desired response than the TO method. Regarding manufacturability, the TO method produced a simpler design with larger material features that would be less susceptible to manufacturing defects. Additionally, the TO method uses a 100x100 discrete domain, allowing for improved refinement compared to the DRL method's 60x20 discrete domain.

A direct computational cost comparison cannot be made. However, due to the DRL agent being pretrained, the DRL method is expected to be orders of magnitude faster than the TO method. It should be noted that there is significant time and computational investment to train the DRL agents and their accompanying VAE and surrogate models. However, once properly trained, the agent provides a near-instantaneous unit cell design recommendation for a wide array of desired responses. Additionally, the DRL method has improved generalizability. The TO method must rerun the time-consuming optimization process every time a new nonlinear response is required. The optimization process must find a new optimal relationship between the design variables and new objective. If the desired response were to change, the DRL method could rapidly design a unit cell without retraining its policy.

While not directly exhibited in Figure 4-20, the main advantage of the DRL approach over TO is the lack of reliance on differentiable objective and constraint function(s). This limitation of TO makes introducing specific rules to the design process challenging. These design rules are often geared towards manufacturing constraints, such as maximum overhang, minimal thickness, single-continuous body, or minimum fillet radii. These design rules can be difficult or impossible to describe using a differentiable constraint function needed to calculate the gradient that updates the design variables.

The DRL method does not need differentiable constraint functions to introduce design rules. Instead, these design rules can be introduced in the reward function. The agent should receive a hefty penalty if it proposes a design that violates any design rules. Over time, the agent will learn to produce objective-based designs that satisfy the given design rules [37]. The DRL approach in this work implements three design rules: 1. The unit cell must be a single continuous body 2. The material must connect the four corners of the design domain 3. The material elements cannot be connected by a single hinge node. While these design rules are relatively simple, the DRL agents could easily be retrained to design unit cells according to more complicated design rules, such as relaxing/changing the four-corner rule, specifying a minimum material thickness value, and reducing the allowable material overhang. These design rules can improve the manufacturability of the proposed designs. This ability to customize specific design rules according to design application is considerably more challenging using gradient-based optimization tools.

Figure 4-20b shows that the unit cell synthesis method achieved a design with slightly better agreement between the desired and resulting response. Both designs appear to have simple shapes that should not pose manufacturing challenges. The most significant advantage of the DRL method is considerable improvements in efficiency and generalizability. Unit cell synthesis requires size and shape optimization for each new desired force-displacement response. Additionally, the desired response may not be achievable given the current geometric parameterization. Deriving a new initial starting unit cell to be parameterized is time-intensive and is often completed through trial and error [83]. Therefore, unit cell synthesis can be feasible when designing for a small number of desired responses but is deemed impractical as a rapid-inline design tool.

The brute-force approach to designing unit cells with targeted nonlinear deformation responses would involve designing many random unit cells, tessellating them, and determining their force-displacement response using simulation or experimental methods. Then, given a large enough sample of unit cells, a user could pick the unit cell with a force-displacement response most similar to the desired response. Regarding the proposed work, the claim could be made that since a large sample of unit cells was already designed and simulated to train the surrogate models, the computational investment has already been made to validate using the brute-force method. While this claim is worth consideration, the DRL method offers an advantage over the brute-force method through its sequential design of the unit cells.

Using the DRL method, a human designer can see the step-by-step design of a unit cell and track how additions and alterations to crucial physical features can result in changes to the resulting force-displacement response. Alternatively, the brute force method will output a single-unit cell design with limited information about how different features contribute to the deformation response. Understanding the structural design of a unit cell can allow a human designer to manually introduce tweaks to the design and act as a co-designer along with the DRL agent.

Figure 4-21 shows how the DRL agent offers insight into its design strategy. The DRL agent takes three steps to produce a unit cell with the desired force-displacement response. However, the third action offers insight into how the force-displacement curve can be tailored. With the third action, the agent adds a small material support connecting the oval and x-shaped regions of the unit cell. A small angle exists between this support material and the bottom horizontal surface. Tracking the force-displacement response throughout the design process shows that this small support stiffens the response of the unit cell. Therefore, a human design could take this information and alter the angle between the support and bottom horizontal surface to provide additional stiffness if the desired response needed to be changed, as seen by the top two unit cell designs. This insight would not be possible with the brute-force approach as there is no sequential tracking of how the force-displacement curve changes.

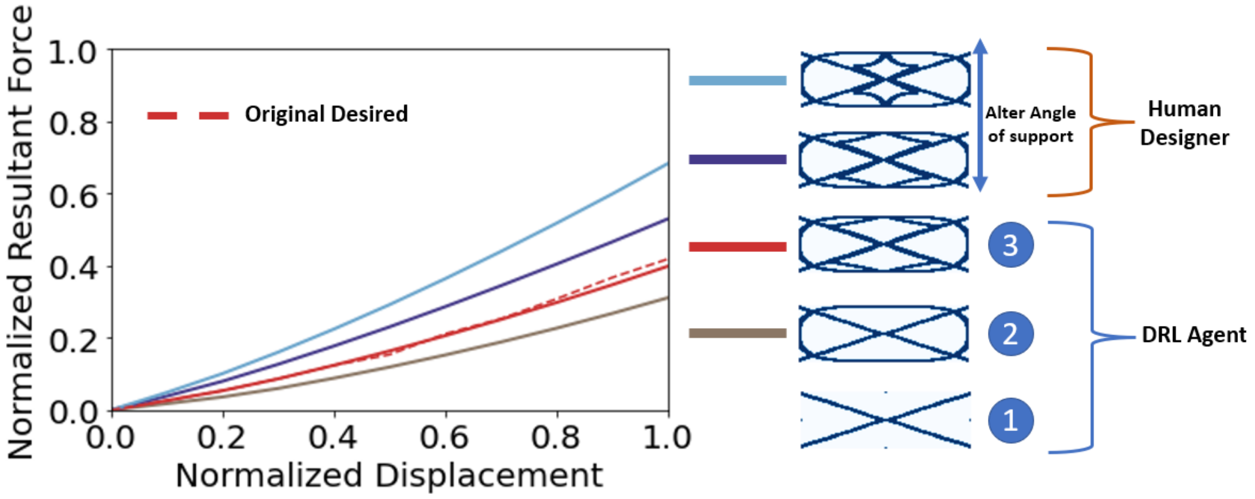


Figure 4-21. DRL agent and human designer collaboration

4.5.4 Domain and DRL Improvements

While the results shown in this chapter are promising, several design domain improvements could be made to increase the validity of using the DRL method as a physical design tool.

The first improvement would be to run the FEA simulations with nonlinear elastic materials. In the presented domain, the unit cells are constructed from linear elastic materials with reference properties. Therefore, the nonlinear deformation responses are entirely controlled by the nonlinear geometry of the unit cell. Incorporating nonlinear material models could task the DRL agents with achieving a desired force-displacement response by combining the nonlinear elasticity of the constitutive material and the nonlinear geometries of the unit cell. Incorporating nonlinear material models will validate that the DRL method can be incorporated to tackle a wide array of real-world design problems. This change is a fundamental component of RQ3.

The second improvement would ensure the DRL agents account for stress concentrations and material yielding when designing the unit cells. The force-displacement response is the only output of interest in the current environment since the metamaterials are evaluated for a single loading sequence. The DRL agents do not check if material yielding has occurred, even though, given the 20% metastrain constraint, it is fair to assume that yielding had occurred for some unit cell designs. Material yielding would drastically alter the mechanical performance if the metamaterials were tested under cyclical loading. Therefore, a design rule should be implemented in future work that ensures that proposed unit cell designs do not exhibit material yielding. This design rule should prompt the agents to propose designs that reduce stress concentrations.

A third improvement would include designing for various loading types. This work only tested metamaterials in uniaxial tension or compression. Few real-world applications call on uniaxial loading. Therefore, the DRL agents should be tested on various loading types, including shear, unevenly distributed loads, and combined loading. These unique loading combinations would teach the DRL agents to develop a generalized design strategy for any force-displacement curve for multiple loads. While implementing these changes may be time intensive, there could be incredible value in designing mechanical metamaterials with targeted deformation responses in multiple axes. These improvements are a small sample of the advances that could be applied to designing mechanical metamaterials via DRL. The proposed work, published in the following article [130], can be the foundation for mechanical metamaterial design tasks such as energy-absorbing ceramics, hysteresis-minimizing foams, and shape-morphing metals.

CHAPTER FIVE

RESEARCH QUESTION 3: FRAMEWORK AND IMPLEMENTATION

The limitations and envisioned future of DRL for metamaterial design discussed in RQ2 gave rise to the work completed to solve RQ3. RQ2 provides an initial validation that DRL can be used to design mechanical metamaterial with tailorable deformation responses. However, the arbitrary material properties and lack of experimental validation limit the real-world applicability of the proposed design method. Therefore, RQ3 expands on RQ2 by introducing a material model validated against experimental results. Additionally, the DRL is given a more challenging design task by designing mechanical metamaterials with tailorable deformation and energy manipulation responses. These improvements are essential in further validating DRL as a powerful engineering design tool and show the first example of mechanical metamaterials being designed and experimentally validated with customizable deformation and energy responses. These unique performing materials have potential applications in such domains as footwear, wearable technologies, and medical equipment. There is considerable overlap between some methods used in RQ2 and 3. In fact, the design domain in RQ3 follows the same format as RQ2 with the alteration that the thickness variable of the added material elements according to the cubic Bezier curves, T, was increased to 3 or 4. This increase is to ensure that the metamaterials can be physically manufactured. Smaller material features can be challenging to manufacture. Therefore, the unit cells in RQ3 (example shown in Figure 5-1) have a higher material volume fraction than RQ2. All other aspects of the design domain remain the same as RQ2, and the reader is directed to the previous chapter for any clarification on the methods.

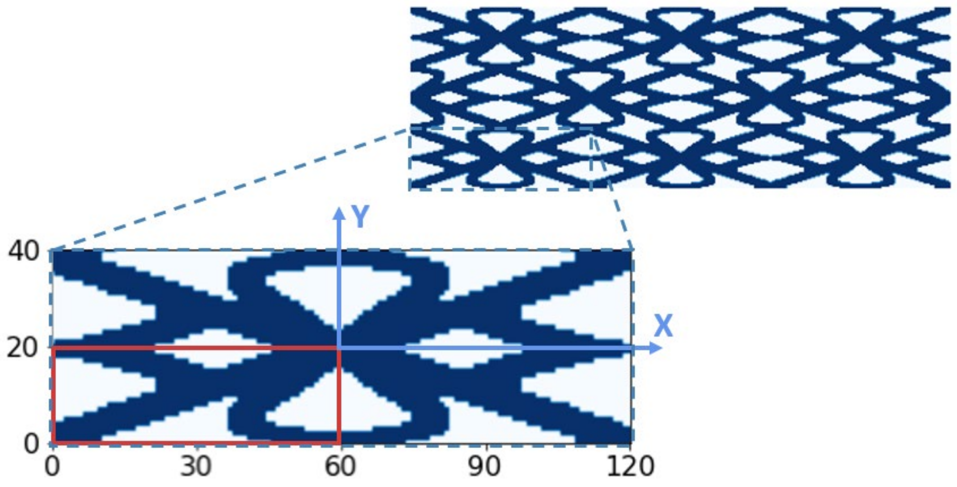


Figure 5- 1. A randomly generated unit cell of discrete elements (material in navy, void in white)

5.1 Methods: Defining Nonlinear Elastic Material Model

5.1.1 Capturing Deformation Response

The metamaterials' mechanical properties rely on the complex interactions between the individual unit cell design and constitutive material. RQ3 aims to design metamaterials that exhibit tailorable deformation and energy manipulation properties. Therefore, finite element analysis (FEA) was performed using *Abaqus/CAE* (Dassault Systèmes) to determine the loading and unloading stress-strain responses of these metamaterials. Unlike the work in Chapter 4, the metamaterial constitutive material model is no longer arbitrary but calibrated to match the experimental results of thermoplastic polyurethane (TPU).

TPU is a thermoplastic elastomer that offers the mechanical performance characteristics of rubber (high resilience, compliance, abrasion resistance, and flexibility) but can be processed like thermoplastics through additive manufacturing [64]. TPUs are built as randomly segmented copolymers consisting of hard and soft segments. The hard segments act as the physical crosslink immersed in the soft segment matrix. This combination of segments imparts the unique strain rate-dependent deformation and hysteretic responses found in TPU [131]. The unique characteristics of

TPUs result in a wide array of applications ranging from wearables and footwear to medical devices and automotive parts. TPU was selected as the constitutive material precisely due to its low cost, ease of manufacturing, and compliance and hysteretic properties relatively similar to those of common running shoe midsole materials such as elastomeric polymers or expanded-thermoplastic polyurethane (E-TPU).

5.1.2 Additively Manufacturing Samples

The FEA material model was calibrated using experimentally determined compressive loading and unloading responses of additively manufactured samples using *Filaflex 70A* (*Recreus,* Elda, Spain) 2.85mm filament. This 'ultra-soft' TPU filament is highly elastic with a Shore hardness of 70A and can be used in most common fused-deposition modeling (FDM) 3D printers. Three 70x30x30mm metamaterials (Figure 5-2) were additively manufactured from *Filaflex 70A* using a LulzBot Mini FDM printer (Fargo Additive Manufacturing Equipment 3D) with 235°C head temperature, 50°C print bed temperature, and 14mm/s print speed. These designs were produced following the cubic Bezier curve design method with a random number of curves with randomly selected control points. To ensure a continuous unit cell, the starting and ending control points were limited to one of the domain's four corners. The solid regions of the metamaterial were printed with 100% infill. Three samples were printed and tested for each design.

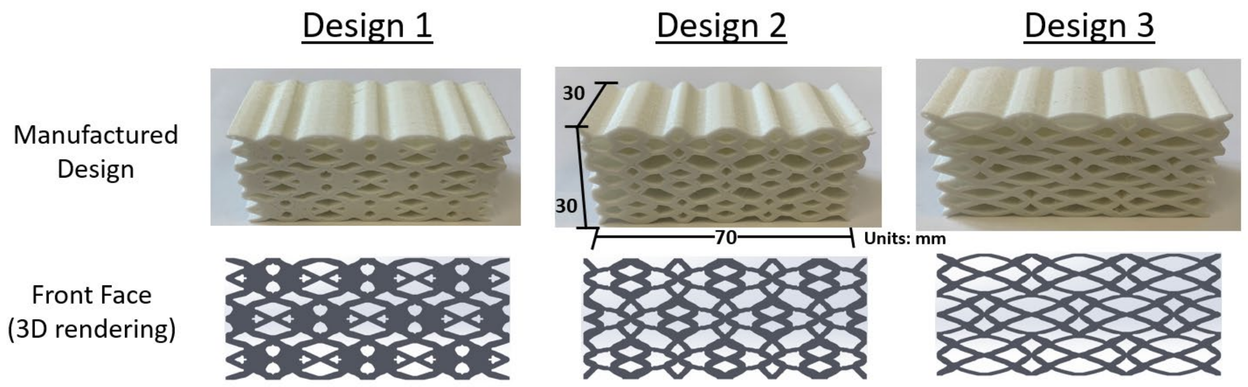


Figure 5- 2. Three randomly designed and manufactured metamaterial samples

5.1.3 Experimentally Testing

The nine metamaterial samples were experimentally tested under cyclic compression loading. The tests were performed at room temperature (25 °C) using a universal testing machine (Instron 68TM-10, Norwood, MA) with a 10 kN load cell. The tests were performed using a 100mm crosshead with a speed of 248 mm/min and tested up to 20% strain (6mm), resulting in a 0.138 s^{-1} strain rate. The compression-decompression cycle was repeated five times per run per specimen to capture any specimen softening that may occur from repeated loading. The loading and unloading stress-strain curves were captured for each specimen. Each sample was subjected to three runs of the cyclic compression testing schedule defined in the Instron Bluehill Universal software.

To serve as a performance benchmark, three 70x30x30mm samples of Infinergy® 230 MP (BASF, Ludwigshafen, Germany) were bandsaw cut from a 400x400x30mm slab of the material and were tested using the same experimental methods. Infinergy® was the world's first expanded thermoplastic polyurethane (E-TPU). This material is produced by molding TPU bead granules under high temperatures and pressures. The pressure and heat cause the granules to expand and exhibit a foam-like structure. Infinergy® provides exceptional energy return, strength, and weight properties for footwear, flooring, and sports equipment applications [132].

5.1.4 Calibrating Material Model

A constitutive model of TPU has been proposed by Qi and Boyce [64] but was deemed unnecessarily complex to implement in the given study. The primary focus of this work is to validate DRL as a powerful engineering design tool; therefore, a material model calibrated against experimental data was deemed sufficient. The *Abaqus* hyperelastic model combined with hysteretic properties has previously been used to model TPU [131] and was calibrated using the randomly designed metamaterial's experimental results. The polynomial strain energy potential with order two was used, following Equation 31:

$$U = \sum_{i+j=1}^{2} C_{ij}(\bar{I}_1 - 3)^i(\bar{I}_2 - 3)^j \tag{31}$$

where C_{ij} are material properties determined by cyclic compression testing and $\bar{I}_1$ and $\bar{I}_2$ are the first and second deviatoric strain invariants, respectively. The materials are assumed incompressible as they are tested in their rubbery state at room temperature.

The hysteresis model in *Abaqus* is intended for modeling the large-strain, rate-dependent behavior of elastomers, specifically to capture the pronounced hysteresis in stress-strain curves over cyclic loading. The model does not capture the "Mullins effect," commonly seen in TPUs, which refers to the material softening experienced by elastomers during the initial loading cycles. While this limitation may reduce the accuracy of the proposed model, the experimental testing results show limited softening over the five-cycle compressive loading; therefore, the Mullins effect should have a limited effect in the current design domain.

The hysteresis model [133] decomposes the mechanical response into an equilibrium network that works in collaboration with the hyperelastic model and is characterized by the following four parameters: S, a stress-scaling factor that defines the ratio of the stress carried by a time-dependent vs. equilibrium network; m, a parameter characterizing the effective stress dependence of the effective creep strain rate of the time-dependent network; A, a scaling constant for the effective creep strain rate to maintain dimensional consistency; and E, the creep strain rate regularizing variable. The hysteresis parameters and hyperelastic model were calibrated to ensure the FEA simulations accurately capture the loading and unloading responses of the experimentally tested metamaterials. The comparison results are shown in a later section.

5.1.5 Finite Element Modeling

The FEAs were run as *Abaqus* dynamic-implicit simulations using the material model described above. The model boundary conditions (Figure 5-3) were built to mimic the experimental compressive testing conditions. Therefore, each 70x30mm metamaterial was modeled with a 100mm rigid bar in contact with its top nodal surface. The rigid bar is nondeformable and mimics the flat plate used in experimental testing. The rigid bar is prescribed a vertical displacement boundary condition that compresses the metamaterial to 20% meta-strain over 1.45 seconds and then returns to its original position after an additional 1.45 seconds. This boundary condition maintains the 0.138 s^{-1} strain rate used in the experimental tests. The mid-point of the rigid bar is assigned an additional boundary condition that restricts horizontal motion to ensure uniaxial compression.

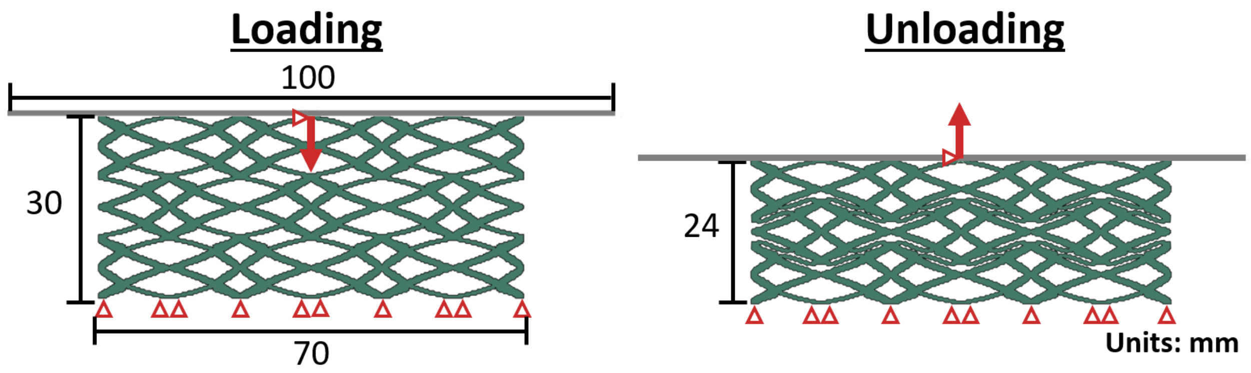


Figure 5- 3. Boundary conditions for cyclic compression simulation

The contact between the rigid bar's bottom surface and the metamaterial's top surface was defined as a rigid, frictionless surface-to-surface contact. Separation after contact was allowed between the two surfaces. The metamaterial's bottom nodal surface was limited to horizontal displacement.

The loading and unloading stress-strain curves were determined by recording the vertical-resultant force and displacement of the mid-point of the rigid bar. The resultant vertical force was divided by the top surface unit area of the metamaterial (70mm^2) to determine the stress, and the displacement was divided by the height of the metamaterial (30mm) to determine the strain.

5.2 Methods: Formation of Deep Reinforcement Learning Problem

The cubic Bezier curve-based method for designing discrete metamaterials needs to be reformulated as a sequential DRL task, precisely, an MDP [41]. Similar to RQ1 and 2, if the Bezier curve design approach can be expressed in terms of state space (S), action space (A), transition probability function (P), and reward function (R), the proposed metamaterial design method can be an MDP.

The details of the agent's interactions with the environment are in the following subsections, but a brief overview helps guide the description. First, the agent starts a design sequence with a wholly voided domain and receives information about the desired loading stress-strain curve and whether hysteresis loss should be minimized or maximized. Given the material distribution and objectives, the agent adds material to the design domain using a cubic Bezier curve. The stress-strain response of the resulting metamaterial is determined, and the agent is rewarded or penalized depending on the difference between the desired and resulting stress-strain curves and the amount of energy loss due to hysteresis. The agent uses this feedback to make intelligent decisions about designing unit cells according to desired deformation and energy characteristics.

5.2.1 Surrogate Model Prediction of Deformation Response

To account for the DRL agent's reliance on a copious amount of data, I trained another surrogate model to predict the deformation response of the metamaterial designs. The deformation responses of the metamaterials are defined as the combined loading and unloading stress-strain curves. The FEA-produced results were reshaped into 42x2 arrays where one column represents the equidistant strain values from 0 to 0.2 and back to 0 over 42 intervals, and the other represents the resulting stress values. Training the surrogate model to capture the relationship between metamaterial design and deformation response by predicting the 42 stress values proved challenging. Therefore, I applied principal component analysis (PCA) to the stress-strain values to produce a low-dimensional data representation. PCA works by finding the directions of maximum variance (principal directions) in a high-dimensional dataset and projecting this data onto these directions. These projections (principal components) now represent a low-dimensional, uncorrelated linear representation of the original data [134].

Ten-thousand metamaterials were designed and simulated in *Abaqus* using the previously described boundary conditions and material model. The combined loading and unloading stress-strain curves were determined for each design, and an 80/20 training/testing set was used to fit the PCA. The results of this PCA fitting are included in a later section, but for clarity, three principal components were used to describe the low-dimensional representation of the stress-strain curves. The first three components were deemed sufficient as they captured 99.9% (83.57%, 15.18%, and

1.15%, respectively) of the variance in the original data. Therefore, these three principal components were determined for each of the 10,000 combined loading and unloading stress-strain curves and served as the output of the surrogate model. Each output was then linked to its corresponding metamaterial design input.

The surrogate model architecture was built with a combination of 2D convolutional and fully connected layers, with a 360x120 discrete array defining the 3x3 tessellated metamaterial as input and the three principal component values as output. The 3x3 tessellated representation was used instead of a single unit cell to visualize the unique features that arise between interacting cells. Making these unique interactions visible proved critical for successful surrogate model training.

Before training, the principal components underwent component-wise normalization. Bayes optimization was used on the architecture (Figure 5-4) and training hyperparameters (Table 5-1) to improve the performance of the surrogate model. The number of convolutional filters, size of the fully connected layers, and training hyperparameters underwent Bayes optimization to improve the model's performance. The parameters of the Bayes optimization were the initial number of convolutional filters, the percent increase of the number of filters every two layers, the initial number of fully connected nodes, and the learning rate. The model was trained to minimize the mean squared error between the true and predicted PCA values. The model was trained for 100 epochs on Clemson University's Palmetto Cluster using an 80/20 training/testing split. The successfully trained surrogate model could now be used as a rapid inline tool for the DRL agent to determine the deformation response of a proposed metamaterial design.

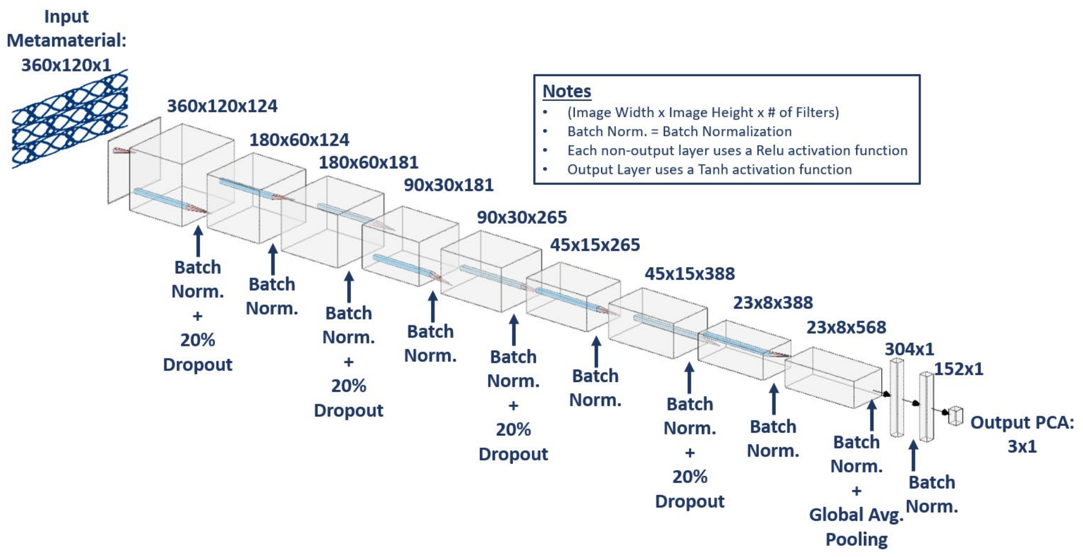

Input Metamaterial: 360x120x1
360x120x124
180x60x124
180x60x181
90x30x181
90x30x265
45x15x265
45x15x388
23x8x388
23x8x568
304x1
152x1
Output PCA: 3x1
Batch Norm. + 20% Dropout
Batch Norm.
Batch Norm. + 20% Dropout
Batch Norm.
Batch Norm. + 20% Dropout
Batch Norm.
Batch Norm. + 20% Dropout
Batch Norm.
Batch Norm. + Global Avg. Pooling
Batch Norm.
Notes
(Image Width x Image Height x # of Filters)
Batch Norm. = Batch Normalization
Each non-output layer uses a Relu activation function
Output Layer uses a Tanh activation function

Figure 5- 4. Deep neural network architecture for PCA prediction surrogate model

Table 5- 1. Hyperparameter for surrogate model training

Hyperparameter	Value
Learning Rate	$1\text{x}10^{-3}$
Batch Size	128
Training Epochs	100
Initial # of Filters	124
Filter Percent Increase	146%
Initial # Fully Connected Nodes	304

The trained surrogate, PCA, and VAE models (the same model trained from Chapter 4) ensure that the cubic Bezier curve design method will be highly efficient and avoid the curse of dimensionality. Therefore, the four components of the DRL environment can be established and should incorporate these powerful tools.

5.2.2 DRL Environment: State Space

For the metamaterial design problem, each observation within the state space should capture information about the design objective (desired deformation and energy responses) and the agent's location in the design domain (current unit cell design). Therefore, the observations are encoded as a 28x1 vector. The first value is a discrete integer that specifies whether the agent should maximize (0) or minimize (1) hysteresis loss. The following three values represent the principal components of the desired deformation response. The remaining values are the 24-dimension latent space of the current unit cell. A sample observation can be seen in Figure 5- 5. The first four values will remain constant throughout a single design episode, but the 24-dimensional latent space values will change as the DRL agent sequentially alters the material distribution.

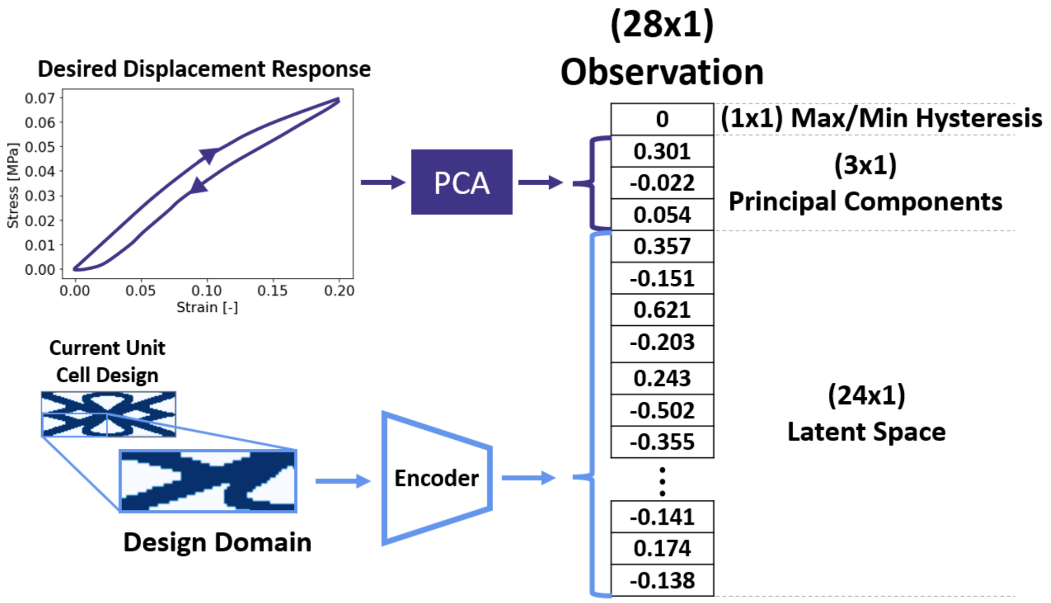


Figure 5- 5. An individual observation within the deep reinforcement learning environment state space

5.2.3 DRL Environment: Action Space

Mimicking the action space defined in Chapter 4, a 7x1 action space allows the agent to select the starting and ending control points, the nodal coordinates for the two intermediate control points, and the thickness of the Bezier curve. The action space was designed to help the agent create fully-connected and continuous unit cell designs. Certain restrictions were placed on the starting and final control points to achieve this. The starting control point (P_0) was limited to either the bottom left (0,0) or bottom right (60,0) nodes of the design domain, while the final control point (P_3) was restricted to either the top left (0,20) or the top right (60,20) nodes of the design domain.

The first element in the action space is a discrete action that determines the position of P_0. A value of 0 sets the nodal coordinates to (0,0), while a value of 1 sets them to (60,0). The second element in the action space is also a discrete element that determines the position of P_3 . A value of 0 sets the nodal coordinates to (0,20), and a value of 1 sets them to (60,20).

The third discrete action element relates to the additional material curve thickness, where a value of 0 results in T=3 and a value of 1 results in T=4. Larger thickness values were used compared to the domain in RQ2 to ensure manufacturability. The X and Y coordinates for the two intermediate control points (P_1) and (P_2) are represented in the action space using continuous values between 0 and 1. The action value between 0 and 1 corresponds to a proportional coordinate value between 0 and 60 for the X coordinate and between 0 and 20 for the Y coordinate. This action space allows the agent to iteratively modify the material distribution within the domain to satisfy the design objectives, as depicted in Figure 5- 6.

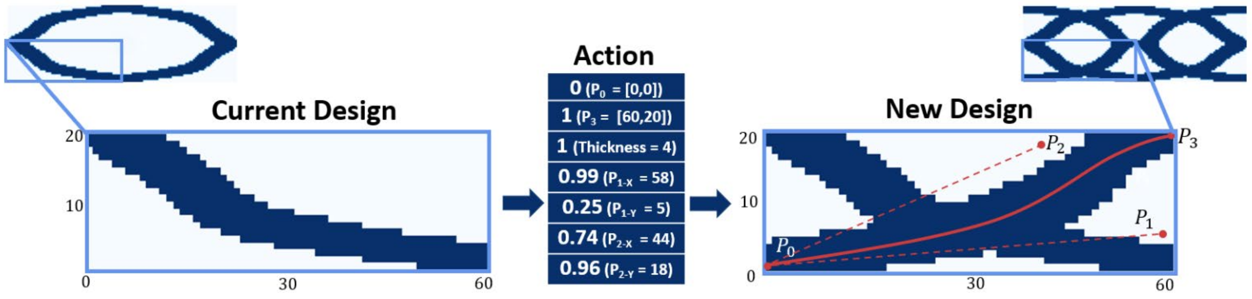


Figure 5- 6. A representation of how the DRL agent adds material to the design domain

5.2.4 DRL Environment: Reward Function

In the presented environment, the reward function was modeled to enable the agent to design metamaterials that exhibit targeted deformation and energy return characteristics. A two-part negative reward function was formulated. The first part penalizes the agent according to the mean absolute percent error (MAPE) between the desired and resulting surrogate-model predicted loading stress-strain curves. To avoid exploding gradients, the maximum curve error penalty is limited to -2. Therefore, if the MAPE between the desired and resulting loading stress-strain curves exceeds 200%, the agent will only be penalized -2. The second component of the reward penalizes the agent according to the difference between the percent energy return and an energy return threshold value.

The energy return phase of the reward function was built to capture the phenomenon that a metamaterial's achievable energy return depends on its compliance. More compliant metamaterials traditionally exhibit less constitutive material strain and internal friction, which can result in higher energy return. Alternatively, less compliant metamaterials commonly have higher material volume fractions and energy return properties that closely align with those of the constitutive material. Therefore, the distribution of achievable energy returns should increase relative to compliance.

The energy return reward function was built to correlate to the maximum stress of the desired loading curve. The max stress and energy return of each of the 10,000 FEA simulations were determined and are shown in Figure 5-7. The mean and standard deviation were calculated for the energy return relative to the max stress in intervals of 0.01 MPa. These values were used to plot curves that correspond to the average (dash-dot), average ±1.28 standard deviations (dashed), and average ±3 standard deviations (solid) energy return relative to the max stress. These curves were produced by curve-fitting fourth-order polynomial equations that result in the following equations, given max stress (MS):

$$Upper\ Bound = 6.042 * MS^4 - 11.827 * MS^3 + 8.096 * MS^2 - 2.228 * MS + 1.023 \qquad (32)$$

$$Upper\ Bonus = 4.693 * MS^4 - 9.113 * MS^3 + 6.184 * MS^2 - 1.678 * MS + 0.957 \qquad (33)$$

$$Avg.Return = 3.689 * MS^4 - 7.093 * MS^3 + 4.761 * MS^2 - 1.269 * MS + 0.907 \qquad (34)$$

$$Lower\ Bonus = 2.684 * MS^4 - 5.074 * MS^3 + 3.338 * MS^2 - 0.860 * MS + 0.859 \qquad (35)$$

$$Lower\ Bound = 1.335 * MS^4 - 2.359 * MS^3 + 1.427 * MS^2 - 0.311 * MS + 0.793 \qquad (36)$$

The average ±3 standard deviations curves, referred to as the Bounding Curves, are used as the energy return threshold values. The agent receives a smaller negative reward for achieving an energy return closer to the upper or lower bounding curve depending on a minimize or maximize hysteresis objective, respectively. If the energy return is outside the ±1.28 standard deviations curves, referred to as the Bonus Curves, and the MAPE between curves is less than 10%, the agent is given a positive reward (1- MAPE), and the design episode is terminated. Achieving an energy return outside ±1.28 standard deviations represents a design greater than the 90th percentile. Therefore, the agent has produced a high-performing design solution and should not continue receiving negative penalties. The only other way for an agent to terminate a design episode is to take more than six actions.

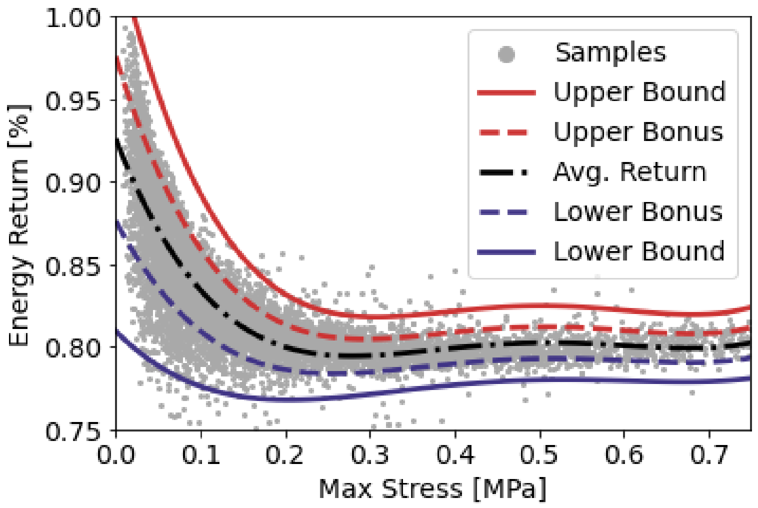


Figure 5-7. A comparison of the energy return and loading curve max stress used to determine energy return threshold and bonus

The trained surrogate model can only predict the resulting stress-strain response of a legally designed metamaterial. Legal designs are represented as 60x20 design domains with a single, continuous material body with material elements in the four corner nodes ([0,0], [0,20], [60,0], [60,20]. Legal designs ensure that the unit cells can be tessellated into continuous metamaterials. If the DRL agent takes an action that results in an illegal design, the agent is penalized -2. This penalty should teach the agent to avoid actions leading to illegal designs. The exception to this rule is during the first action, as it is impossible to connect all four corner nodes with a single cubic Bezier curve. Therefore, on the first action, the agent always receives a reward of 0.

The reward function, r_t, at a given episode timestep, t, where t:[0,5] can be summarized for hysteresis maximizing and minimizing objectives according to Equations 37 and 38, respectively:

$$r_t = \begin{cases} 0 & if\ t = 0 \\ -2 & if\ t \neq 0\ \&\ Illegal \\ 1 - MAPE & if\ Legal\ \&\ MAPE \leq 0.1\ \&\ ER < LBS(MS) \\ max[-2, -MAPE] - \big(ER - LBD(MS)\big) * S & Else \end{cases} \tag{37}$$

where MAPE is the mean absolute percent error between the desired and resulting curves, ER is the energy return of the resulting stress-strain curve, LBS(MS) is the Lower Bonus Value given the max stress of the desired loading curve, MS, LBD(MS) is the Lower Bound Value given MS, and S is a scaler value that equals the inverse of the normalized energy return standard deviation at the max stress of the desired stress-strain curve. The standard deviation is normalized about the energy return standard deviation at all other max stress values. This scaler value normalizes the contribution of the energy return penalty regardless of the max stress.

$$r_t = \begin{cases} 0 & if\ t = 0 \\ -2 & if\ t \neq 0\ \&\ Illegal \\ 1 - MAPE & if\ Legal\ \&\ MAPE \leq 0.1\ \&\ ER > UBS(MS) \\ max[-2, -MAPE] - (UBD(MS) - ER) * S & Else \end{cases} \tag{38}$$

where UBS(MS) is the Upper Bonus Value given the max stress of the desired curve, MS, and UBD(MS) is the Upper Bound Value given MS. While this objective-based reward function cannot be proven as optimal or the only viable approach, the effectiveness of this reward function is demonstrated by the favorable results in the following sections.

5.2.5 DRL Environment: Transition Probability Function

The deep deterministic policy gradient (DDPG) was implemented in the DRL environment to account for the continuous state and action spaces. This algorithm has also been used for RQ2. While the DDPG has been successfully implemented for RQ2, I would like to restate that I do not claim that DDPG is the only or optimal DRL algorithm for this application. DDPG can suffer from instability during training, inefficiency, high sensitivity to hyperparameters, and sub-optimal policy convergence. However, DDPG was selected due to its ease of implementation, previous success, and ability to account for continuous state and action spaces.

5.3 Methods: Training the DRL Agent

The previous sections defined the metamaterial design problem as an MPD, and therefore, a DRL agent should be able to learn to design mechanical metamaterials with customizable deformation and energy return responses. In the given design problem, an episode is defined as the DRL agent sequentially adding material to the unit cell design domain until the resulting loading stress-strain curve is sufficiently close to the desired curve and the energy return satisfies the Bonus threshold or the agent takes too many steps.

The agent was trained to design unit cells given a diverse range of desired stress-strain curves and randomly selected objectives of maximizing or minimizing hysteresis loss. The 10,000 stress-strain curves used to train the surrogate model were used as the possible desired curves. These curves underwent an 80/20 training/testing split. Therefore, at the beginning of each training episode, one of the 8,000 training stress-strain curves was randomly selected as the desired loading curve. Additionally, the energy return objective was randomly selected to maximize or minimize hysteresis.

The desired loading curve, in the form of its principal components, and the Boolean energy return objective define the first four values of the observation, while the remaining 24 values are 0 due to the wholly voided starting design domain. This observation is passed to the actor-network, and an action is selected. To promote design domain exploration, the same decaying noise vector used in Chapter 4 is added to the selected action by sampling from a normal distribution with a mean, μ, of 0 and standard deviation, σ, that decays according to the following equation:

$$\sigma = max\left[\sigma_{max} - N \cdot \sigma_{decay}, \sigma_{min}\right] \tag{39}$$

where σ_{max} is the starting action noise ($\sigma_{max} = 0.25$), σ_{decay} is the rate at which σ decreases per action ($\sigma_{decay} = 4.5x10^{-6}$) until σ reaches its minimal allowed value, σ_{min} ($\sigma_{min} = 0.01$), and N is the current number of actions taken during all training episodes.

The action and added noise define the cubic Bezier curve that adds material to the design domain. The resulting design domain is double-mirrored to produce the unit cell of the metamaterial. The surrogate model predicts the metamaterial's stress-strain response (via principal component prediction), which is compared to the desired curve and energy return objective to calculate the reward. Next, the new material distribution is passed to the VAE encoder to produce the new observation. After each action, the previous observation, new observation, action, and reward are recorded in a replay buffer. Once the replay buffer includes at least 300 experiences, a batch of 64 experiences is randomly sampled after each design episode to train the DDPG actor and critic networks. The architecture of the actor and critic networks and other hyperparameters can be found in Figure 5-8 and Table 5-2, respectively. Due to increased training times, the hyperparameters underwent limit tuning but not Bayes optimization and cannot be considered optimal. However, the favorable performance of the DRL agent confirms that the selected hyperparameters are sufficient.

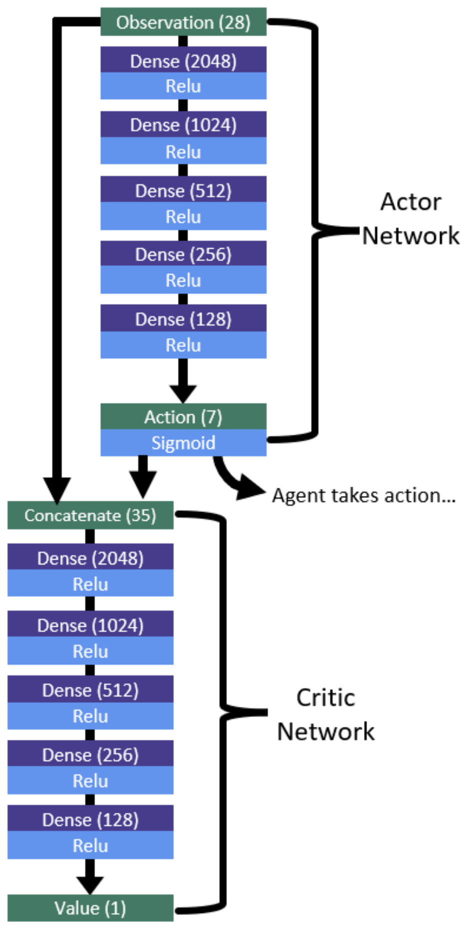


Figure 5- 8. The architecture of the actor and critic networks in the deep deterministic policy gradient algorithm

Table 5-2. The deep deterministic policy gradient architecture hyperparameters

Hyperparameter	**Value**
Learning Rate	$1x10^{-4}$
Tau	$5x10^{-3}$
Batch Size	64
Max Action Noise	0.25
Action Noise Decay	$4.5x10^{-6}$
Min Action Noise	0.01
Min Replay Buffer Size	300
Max Replay Buffer Size	20,000
Gamma	0.99
Training Epochs	20,000

5.4 Results

5.4.1 Material Model Results

The *Abaqus* hyperelastic material model and hysteresis parameters were validated against the experimental cyclic loading. Figure 5-9 shows the average of the three experimental loading and unloading stress-strain curves for each metamaterial design compared to the FEA response of the metamaterials captured by the *Abaqus* material model. The three experimental deformation responses used to determine the average response for each metamaterial design can be found in Appendix A. The figure shows that the FEA model can accurately capture the general compressive trends from the experimental tests.

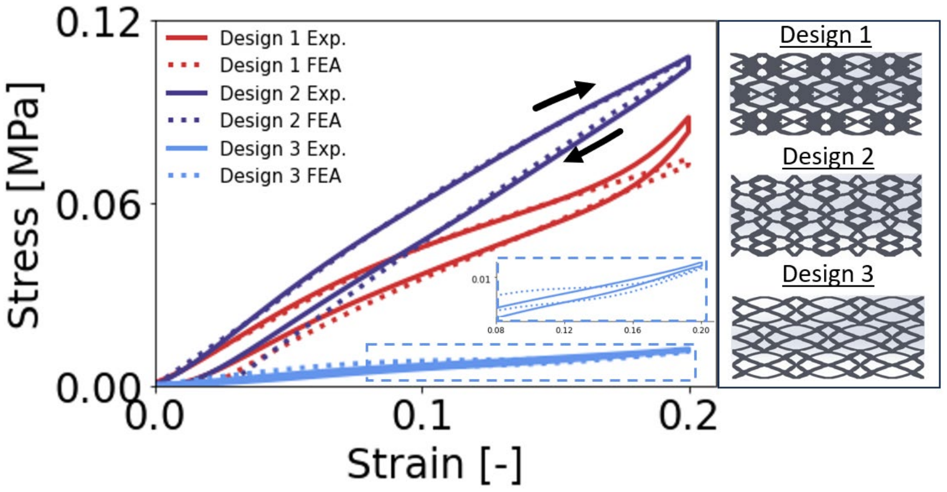


Figure 5-9. The average experimental cyclic compressive response compared to the FEA simulation for the three random metamaterials

As previously stated, each sample underwent five compressive loading and unloading cycles. The experimental results showed slight material softening between the first and second cycles, as seen in Figure 5-10; therefore, the loading and unloading curves of the third cycle have been used to define the deformation response of the metamaterials. The model accurately captures compliance magnitude and nonlinearity exhibited in the experimental results. However, limitations such as disregard for internal contacts and larger residual strain values can hinder the model's accuracy. These limitations are discussed in more detail in a later section. While these two limitations should be considered, I believe the model is sufficiently accurate to validate its use in the DRL design problem.

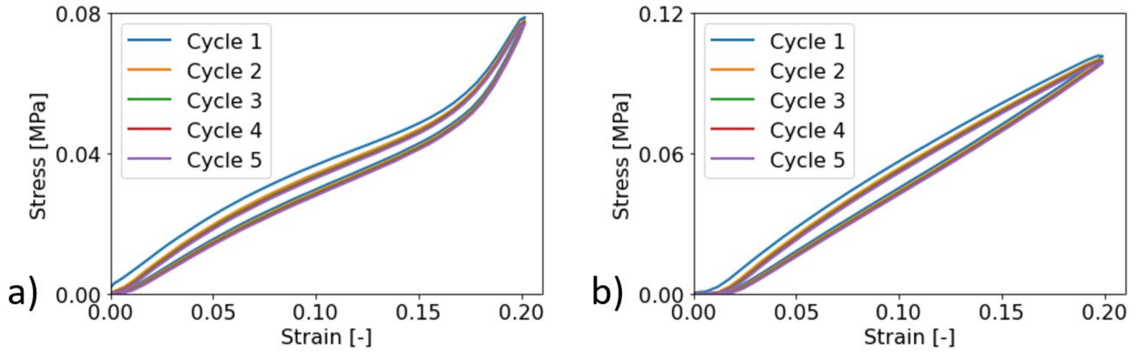


Figure 5- 10. The five loading and unloading stress-strain curves from a single experimental trial of a) Metamaterial Design 1 and b) Metamaterial Design 2

5.4.2 Stress-Strain Prediction Results

The validated material model was used in the 10,000 FEA metamaterial simulations that trained the PCA and surrogate models. The accuracy of the trained PCA model can be determined by calculating the MAPE between the original 2,000 testing stress-strain curves and recreated curves produced by inversely transforming the principal components. To determine the minimal number of required principal components, the authors calculated the MAPE for PCA models trained using an increasing number of principal components. Table 5-3 shows that the MAPE decreases as the number of principal components increases. The error appears to converge at three principal components, where an additional principal component would result in a minuscule increase in reconstruction accuracy.

As previously stated, three principal components account for 99.9% of the variance in the stress-strain curves. This surprisingly high variance attribution may indicate that the *Abaqus* material model is overly simplified. While the simplification of the proposed material model should be addressed in future work, the reconstruction accuracy of the three-component PCA proved to be sufficient, such that the trained PCA can be used to define a low-dimensional representation of the loading and unloading stress-strain curves of the metamaterials. Therefore, the principal components can be used as the output of the surrogate model and within the DRL observation.

Table 5- 3. Mean absolute percent error and variance attribution for an increasing number of principal components to describe the stress-strain curves

No. of Principal Components	MAPE [%]	Variance Attribution [%]
1	9.49	83.57
2	4.62	15.18
3	2.87	1.15
4	2.57	1.34e-3
5	2.25	2.07e-4

The trained surrogate model was tested using the 2,000 samples from the testing dataset. The surrogate model was evaluated by calculating the mean absolute error (MAE) between the true and predicted principal components of these 2,000 samples. The principal components of the 2,000 samples underwent the inverse PCA process to produce the resulting predicted stress-strain curves. Additionally, the MAPE was calculated between the predicted and true stress-strain curves. The MAE was calculated for the principal components instead of MAPE due to the large concentration of values near 0 which could artificially inflate a percent-based error metric.

The results in Figure 5-10 and Table 5-4 show that the surrogate model can predict the principal components corresponding to the stress-strain curves of random metamaterials with sufficiently high accuracy. These results show that the surrogate model's prediction capabilities progressively worsen from principal component #1 to #3. This worsening performance can be seen in Table 5-4, with a weaker linear correlation between the true and predicted values and a higher reconstruction error for principal component #3 compared to principal components #1 and #2.

Table 5- 4. The error and R^2 values for the surrogate model performance for predicting each principal component

Dataset	MAE [-]	R^2
All Principal Components	0.079	0.922
Principal Component #1	0.046	0.984
Principal Component #2	0.064	0.950
Principal Component #3	0.121	0.760
	MAPE [%]	
Stress-Strain Values	8.8	0.992

It should be noted that principal component #3 has the largest concentration of true and predicted values approaching 0. Very small numbers can result in excessively high percent errors, which aren't indicative of the actual performance of the surrogate model. Fortunately, principal component #3 has the smallest variance attribution on the stress-strain curves, and poor prediction performance on this one component does not severely hinder the surrogate model's ability to accurately predict the stress-strain curves. The 2,000 surrogate model predictions were completed ~170x faster than running the FEAs. The combination of accuracy and efficiency validates that the surrogate model can act as a rapidly accurate deformation response prediction tool for the DRL agent.

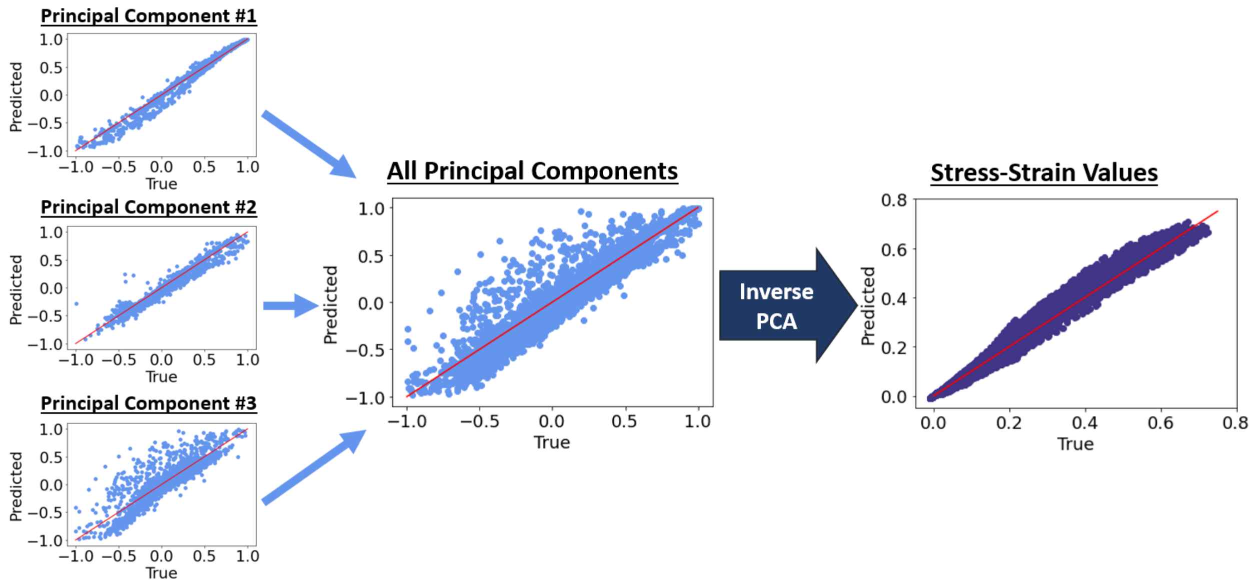


Figure 5- 11. Comparing the true vs. predicted values from the surrogate model

5.4.3 DRL Training Results

The DRL agent was trained for 20,000 design episodes using Clemson University's Palmetto Cluster. Figure 5-11 shows the moving average training reward increases until convergence around 0.5. The increase in reward coincides with the decrease in the action space noise, which decayed to its minimum value after approximately 11,5000 episodes. The increase indicates that the agent is taking better steps, resulting in high-performing metamaterials that more closely align with the desired deformation and energy return characteristics. Also, the reward converging at a value greater than 0 means that the agent regularly receives the positive bonus reward for satisfying the curve error and energy return thresholds.

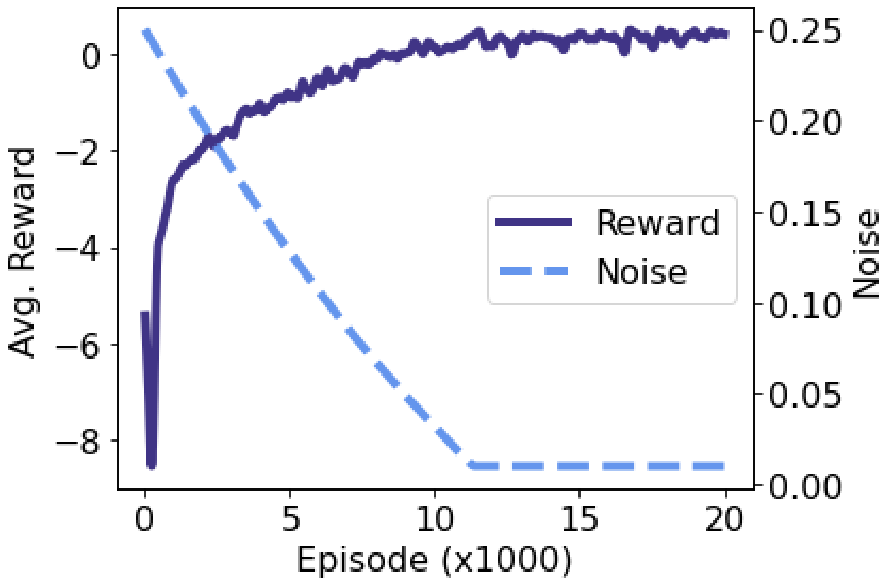


Figure 5-12. The running average of the reward (solid) steadily increases as the action noise (dashed) decreases.

5.4.4 DRL Testing Results

I wanted to provide an initial testing benchmark for the DRL agent by prompting it to design hysteresis-maximizing and -minimizing metamaterials with a compressive deformation response like the experimentally tested E-TPU. The DRL agent was not prompted to design for the E-TPU deformation response during training and could not be overfitting to previously seen experiences. Figure 5- 12 shows that the agent can satisfy the objectives in a two-step design process. By the second step, the proposed unit cells resulted in metamaterials with compressive deformation with a MAPE of less than 10% compared to the desired response. Additionally, the designs exhibit considerably different energy return characteristics that satisfy their respective bonus thresholds. Regarding the hysteresis minimizing metamaterial, the energy return of 88.9% was slightly lower than the E-TPU's return of 90.0% but proved to be in the 92nd percentile of possible solutions at that maximum desired stress. The hysteresis maximizing metamaterial was in the 99th percentile.

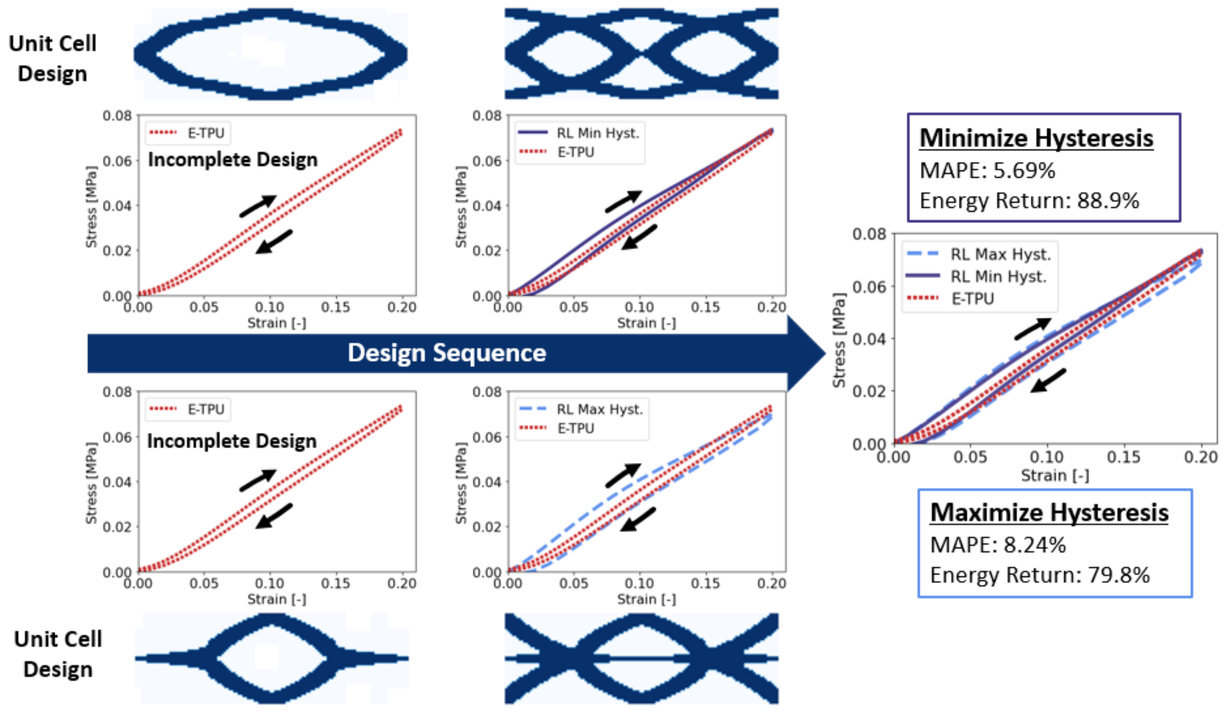


Figure 5- 13. The agent produced hysteresis minimizing and maximizing designs with loading responses similar to the E-TPU material

I wanted to validate that the optimized metamaterials can exhibit variable deformation and energy return characteristics in both simulation and physical testing. Therefore, three samples of the two optimized metamaterials were additively manufactured and experimentally tested using the same methods as the randomly designed metamaterials. Figure 5- 14 compares the DRL predicted (via the trained surrogate model), FEA simulated, and experimental loading and unloading stress-strain curves of the optimized designs. Due to printing variability, the average of the three experimental tests was plotted.

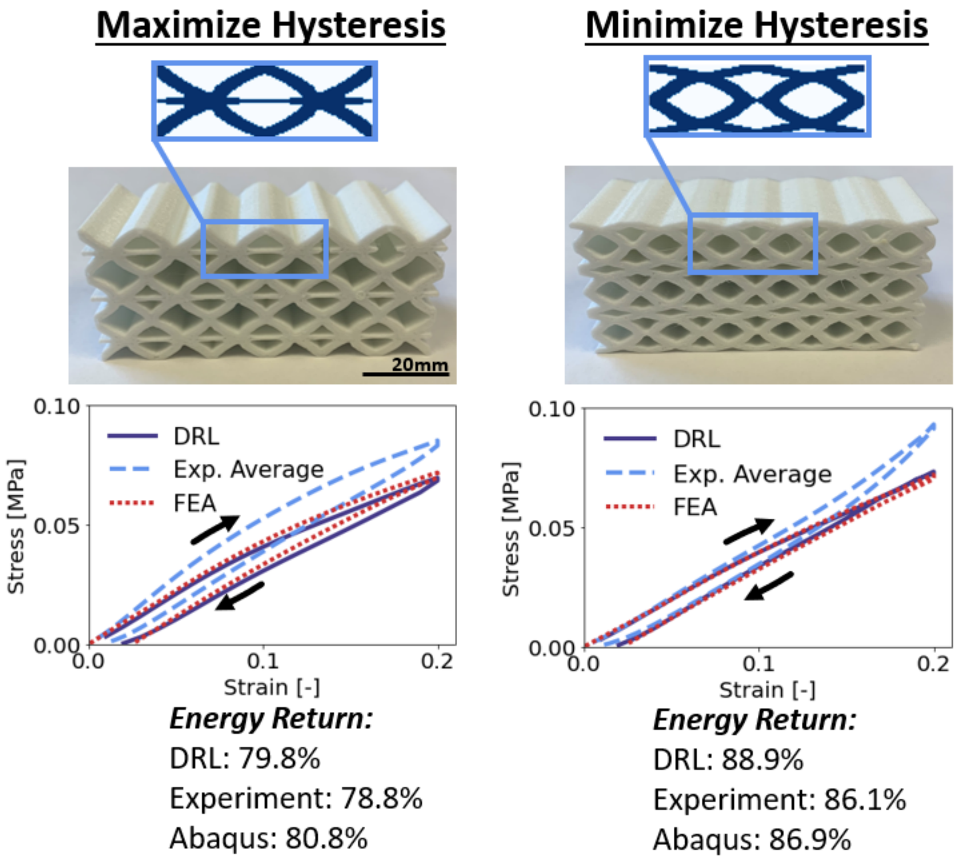


Figure 5- 14. Comparison of the DRL agent (via surrogate model prediction), Abaqus simulation, and experimental cyclic compression testing for the optimized designs.

The individual sample responses are found in Figure 5-14. These results show a sufficient similarity between the three methods for both designs. The average experimental responses are stiffer than the FEA or DRL predicted responses, which is believed to arise from overprinting by the relatively inexpensive FDM printer used in this study. Overprinting can reduce the compliance of a metamaterial by introducing excess material in the unit cell. These overprinted metamaterials will be less compliant and exhibit higher energy loss than the nominal metamaterial designs used in the FEA and DRL simulations. Additively manufacturing TPU via an FDM printer can be challenging [135]. Therefore, future work should investigate alternative additive manufacturing methods for TPU, such as selective laser sintering, to improve the accuracy and reliability of the manufactured parts.

Figure 5-14 shows that the samples suffered from performance variability between prints. This variability was not seen in the original additively manufactured samples and may indicate that the optimized designs are particularly susceptible to overprinting or poor material bonding. These results also indicate that further printing parameter calibration is needed to ensure reliable design performance.

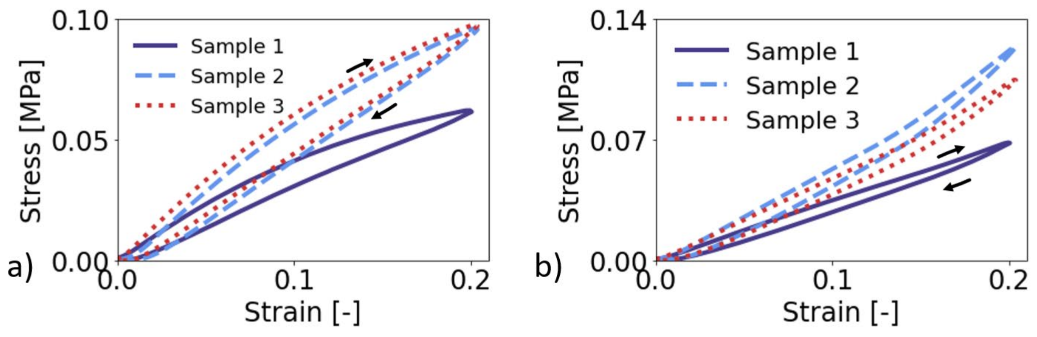


Figure 5- 15. The individual experimental results for the hysteresis a) -maximizing and b) -minimizing metamaterials

This example shows that the DRL agent can design mechanical metamaterials with a targeted deformation response while altering the energy return characteristics. However, it does not prove that the agent has produced a generalized design strategy that can design for various desired deformation and energy return characteristics. Therefore, the trained DRL agent was further tested using the 2,000 testing loading stress-strain curves. The agent was tested on each unique curve twice with both hysteresis minimizing and maximizing objectives. The performance of the agent can be determined by (1) the MAPE between the desired loading stress-strain curve and the resulting stress-strain curve of the DRL-produced metamaterials and (2) whether or not the agent has achieved an energy return outside of the bonus thresholds.

The 2,000 unit cells were designed following a similar process seen in Figure 5- 12. A small sample of resulting hysteresis minimizing and maximizing unit cells are shown in Figure 5-15, based on various desired stress-strain curves. Table 5-5 shows that the DRL agent could design metamaterials that exhibit compressive deformation responses similar to those not seen during training. Additionally, the metamaterials exhibit hysteretic properties that regularly satisfy the upper or lower energy return threshold, depending on the objective. Each of the design episodes took between 1-3 seconds. Therefore, the DRL has proven itself as a highly-accurate and efficient engineering design tool to produce mechanical metamaterials with customizable deformation and energy return characteristics.

Table 5- 5. Evaluating the design performance of the DRL agent

MAPE [%]	Lower Bonus Satisfied [%]	Upper Bonus Satisfied [%]
9.23	95.4	81.8

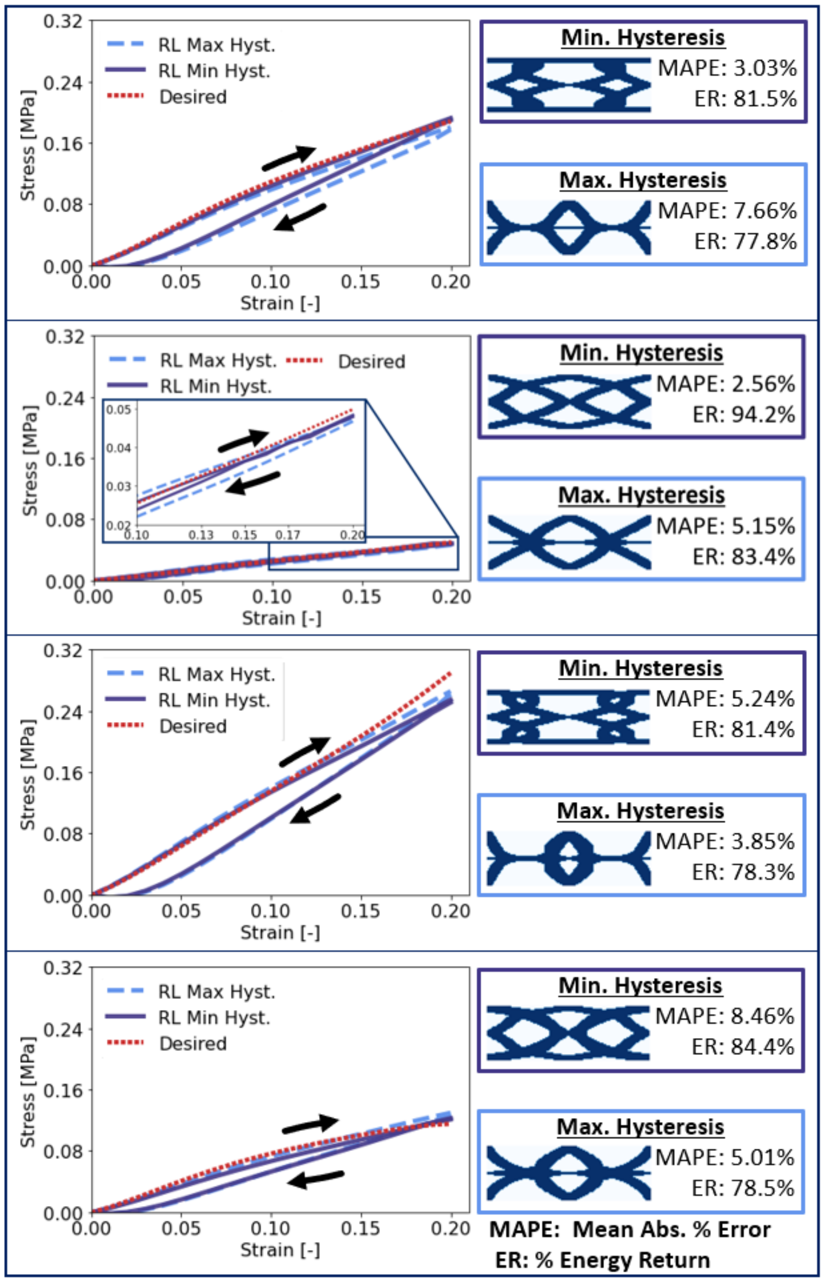


Figure 5- 16. The DRL agent can produce hysteresis-minimizing and -maximizing metamaterials that exhibit an array of desired loading stress-strain curves

5.5 Discussion

5.5.1 Comparing TPU and E-TPU

The Infinergy® E-TPU material was used as a benchmark due to its application as the midsole material for the Adidas® Ultraboost™ running shoe. Running shoe midsoles must be compliant, lightweight, and resilient (high energy return). The experimental results from this work and the literature [136], [137] show that Infingergy® exhibits all these qualities. While the performance capabilities of E-TPUs are impressive, their manufacturing challenges and costs must be considered.

E-TPUs are manufactured by injection molding TPU pellets into the desired shape at high temperatures and pressure. After cooling, the mold cavity is vulcanized with a blowing agent, such as N_2 or CO_2, under high temperatures and pressures to achieve a foam-like appearance and performance. This manufacturing process can require substantial investment in the appropriate equipment and calibration parameters, such as blowing agent type and concentration, mold temperature and shape, cooling time, and injection speed and volume [63]. This investment may be practical for large corporations with mass-manufactured products but may not be feasible under smaller budgets, shorter timelines, and increased customization, justifying an investigation into alternative compliant, lightweight, and resilient materials.

Optimized mechanical metamaterials paired with additive manufacturing may provide an alternative to costly E-TPU. *Adidas*® has already begun implementing this combination with their FUTURECRAFT 4D midsole. This midsole is built as an array of additively manufactured patterned lattices using Digit Light Synthesis (DLS) [138]. The DRL metamaterial design method proposed in this work can also provide an introductory investigation into this domain. The method showed that mechanical metamaterials with a TPU constitutive material could achieve similar deformation and energy return characteristics as an E-TPU sample. It should be noted that the energy return of the hysteresis-minimizing metamaterial was approximately 3.7% lower than the E-TPU, which can be substantial when viewed in the domain of running shoe performance.

Additionally, the mass of the hysteresis-minimizing metamaterial was ~2x heavier than the E-TPU, which would drastically inhibit running performance. With additional investigation, the optimized metamaterial solutions could be made lighter by introducing an additional design rule to the DRL environment that assigns more rewards for designs with less mass. Regardless, the metamaterial samples achieved impressive results while being produced using a relatively inexpensive FDM printer and filament. The authors acknowledge that the testing conditions and manufactured shapes are not representative of a midsole running shoe, and therefore, the applicability of this comparison is limited to the specific experimental methods.

Further research should investigate alternative additive manufacturing methods and materials for developing compliant, resilient, and lightweight mechanical metamaterials for applications beyond midsoles, including other sports equipment, wearable technologies, medical devices, and flooring. The key advantage of implementing metamaterials and additive manufacturing is improved customization. Metamaterials do not need to be built with homogenous unit cells so long as the nonhomogeneous unit cells have features that form a continuous body when patterned together. Therefore, an engineering designer can tailor a structure's deformation and energy response using variable metamaterial designs to account for nonuniform dynamic loading, providing a superior performance to monolithic materials.

5.5.2 Material and DRL Model Limitations

The results from Figure 5-15 show two potential limitations of the proposed DRL design method. The first is that the DRL agent can only design for a relatively limited degree of desired deformation nonlinearity. The second is that the DRL proposed designs show limited topological diversity. The metamaterials' deformation nonlinearities are controlled by the geometric nonlinearities of their base unit cells and the hyperelastic nonlinearity of the TPU constitutive material. The contribution of each is dependent on the material volume fraction of the unit cell. The deformation response of a metamaterial with high material volume fraction is more controlled by the hyperelastic nonlinearity of the constitutive material. Low-volume fraction metamaterials can be highly compliant, and the interaction between neighboring unit cells with geometric nonlinearities helps manipulate the deformation nonlinearity.

The metamaterials investigated in the present work had an average volume fraction of 59.5%, which is considerably higher than the unit cells designed for RQ2, which had an average volume fraction of 35.7%. The present work had higher volume fraction metamaterials because the thickness of the material features was made larger to ensure manufacturability. The larger volume fractions result in a database of metamaterials whose deformation nonlinearity will primarily be controlled by the material model's hyperelasticity. Therefore, the degree of achievable deformation nonlinearity will be limited.

To validate this claim, the authors designed 100 metamaterials using the same design method but limited the material thickness variable, T, to 1 or 2. These 100 samples were simulated using the same material model and FEA methods of RQ3. The dimensional nonlinearity of the resulting 100 loading curves was calculated as the root mean squared deviation of the curve from the straight line connecting the first and last stress and strain values. These 100-dimensional nonlinearities were compared to 100 randomly selected metamaterials designed using the original T=3 or 4 method. Figure 5- 16 shows that the low-volume fraction metamaterials achieve a wider range of dimensional nonlinearity, thus proving the contribution of geometric nonlinearity to the entire deformation nonlinearity. Therefore, the limited range of deformation nonlinearity in the present work can be attributed to the metamaterials being designed with larger concentrations of constitutive material.

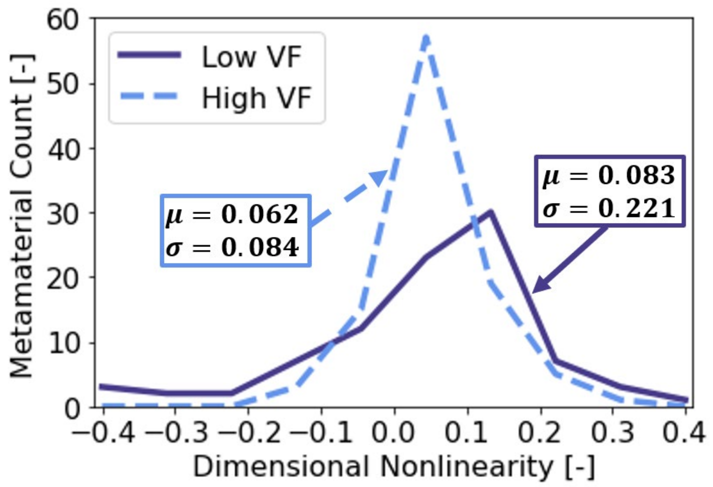


Figure 5- 17 The diversity of dimensional nonlinearity is limited given samples with larger material volume fractions

The assumptions and limitations of the constitutive material model and FEA simulations play a critical role in the limited nonlinear diversity seen in this work. The hysteretic parameters introduced a residual strain in the unloading curve. This residual strain represents the separation of the loading crosshead and the specimen's top surface. This phenomenon was not seen in the experimental testing due to the small strain rate. Therefore, the material model's residual strain artificially increased the hysteresis for the FEA simulations.

The simulations did not consider internal contacts and plastic deformation due to computational costs and convergence concerns. These assumptions ensure that the simulations remain computationally efficient but can also limit accuracy. I recognize that under 20% meta-strain boundary conditions, specific metamaterial designs will experience internal contact that will not be captured in the FEA model. Additionally, the permanent distortion of plastic deformation should be considered in future work if the objective is to achieve targeted nonlinear deformation responses in multi-cycle loading.

Beyond material model improvements, I believe the DRL method should be improved in future work to increase the topological diversity of the proposed metamaterial designs. Figure 5-15 shows that the DRL-designed metamaterials have similar topologies, especially the hysteresis maximizing designs. This similarity can be accredited to the agent taking similar actions given similar observations. The limited nonlinear diversity of the desired deformation responses results in the agent being regularly prompted with similar observations. These similar observations will result in the agent taking similar actions and producing metamaterials with limited topological diversity.

A potential solution for future work is the introduction of new DRL algorithms or reward functions that help the agent achieve design diversity. One such algorithm is Soft Actor-Critic (SAC) [139]. SAC is an actor-critic DRL algorithm based on the maximum entropy framework, which prompts the agent to take more random actions. In the metamaterial design task, the agent would be prompted to produce metamaterials with greater topological diversity. Additionally, including action diversity in the DRL reward function is an active area of research [140], [141]. Incorporating a diversity-based reward function would give the agent additional rewards for producing unique, high-performing metamaterials. This reward would track the previous design suggestions of the agent, calculate the variance of a current design, and assign additional rewards based on the variance.

CHAPTER SIX

GENERAL FRAMEWORK FOR DEEP REINFORCEMENT LEARNING FOR ENGINEERING DESIGN

This manuscript shows that deep reinforcement learning environments can be built to allow agents to iteratively design 2D structural topologies to achieve high-performing design solutions for a diverse range of objectives and constraints. While this work has been limited to 2D grid design domains, I believe that deep reinforcement learning can be used to achieve optimized design solutions in various engineering design domains. To implement deep reinforcement learning, a human user must establish an environment that encodes the core aspects of the design problem that will allow the agent to discover hidden relationships and generate non-intuitive designs that may not be apparent to the human designer.

I wanted to recommend a general framework that could be followed to represent an engineering design problem as a DRL problem. DRL is a generative design tool that can iteratively alter a design to generate an output that meets specified constraints and objectives. The iterative design process is controlled by feedback that controls design domain exploration. Therefore, the most critical components of any generative design problem are design domain representation, feedback, and "tools" to alter the current state in the design domain. All of these features have been addressed in each of the previously discussed DRL design problems.

The first step of establishing an engineering design problem as a DRL problem is ensuring the design domain can be analytically derived. The design domain must be represented mathematically such that its defining features can be utilized for the state space, action space, and reward function of the DRL environment. These forms include but are not limited to:

1. **Discrete elements:** Each design variable is represented as a single element that takes the form of solid material (1) or void (0).

2. **Continuous elements**: The densities of the material elements can range from [0,1]. This representation is common in topology optimization approaches and can drastically increase the design domain but can lead to potentially unmanufactured solutions as there is no physical representation of partial material densities.
3. **Discrete beams**: The design variables take the form of connected beams with variable lengths, orientations, and thicknesses.
4. **Geometric Parameterization**: An initially proposed design is altered to optimize the geometric parameters of critical features in the design space to best satisfy an objective and constraint(s). This parameterization can drastically increase computational efficiencies but limit the explorable design domain.
5. **Point Clouds**: The shape of a structure is defined by the coordinates and connections between a finite number of nodes.
6. **Physical Model**: Obtaining a physical structure model will yield the most helpful information but can be infeasible due to time and cost constraints.

These design domain representations must provide sufficient information to define a relationship between the current design, objective (s), and constraint(s). This information could include topology, shape, segment connectivity, structural performance, and manufacturability. Properly selecting a design domain representation depends on the human designer's thorough understanding of the design task and what information would be critical for satisfying the objective(s) and constraint(s).

The design domain should be rich in information that can be fed into the DRL state space. However, as discussed in the unit cell design problems, large-dimensional design domains can be susceptible to the curse of dimensionality. Therefore, the domain should have a relatively small number of design variables or be able to be represented in a low-dimensional manner through the use of parameter-reduction tools like the principal component analysis or variational autoencoder discussed in the presented work.

DRL agents succeed in producing high-performing design solutions when the state spaces include structural information about the current design and additional information regarding at least one of the following: 1) Objectives, 2) Constraints, and 3) Mechanical performance. The agent can be given this information from partial differential equation solvers, finite element analysis (used in RQ1), analytical equations, encoded features (used in all RQs), and surrogate models (used in RQ2 & 3). This information could also prioritize objectives not explored in the presented work, such as design aesthetics, joinability, or manufacturability.

The action space is highly dependent on the representation of the design domain. The agent can add, delete, or geometrically alter the material in the structure. Selecting an action space representation depends on how the design variables are viewed in the design domain. In discrete grid domains, the agent should be allowed to add and/or remove elements to satisfy the objective. This addition or removal can be completed on individual elements or following constrained forms such as Bezier curves or predefined geometric shapes. The presented work limited the agent to *either* material addition or removal, but the combination would allow the agent to mimic the design methodology of humans. However, initial investigations into this method show that the agent can quickly become stuck in a suboptimal, conservative policy where the agent takes consecutive actions that add and remove the same elements for an entire design episode without producing a meaningful solution. While untested in the presented work, a reward function that penalizes the agent for immediately undoing a step could eliminate this phenomenon.

If the combined action space does not seem practical in a presented design domain, a user can select *either* material addition or subtraction depending on two factors. The first is whether feasible and practical designs can quickly be achieved using material addition. Adding material you need compared to removing material you don't need will almost always be more efficient. For example, the agent took considerably more steps to achieve optimized designs in the TO environment compared to the unit cell design. In the TO environment, the agent could iteratively remove a single material element from a starting topology because individually adding elements would take several steps until a legal design could be achieved, resulting in a sparse reward problem. Alternatively, the unit cell design problem allowed the agent to iteratively add material to a starting blank domain. Therefore, if the design domain can produce continuous, single-body material concentrations that satisfy all constraints using material addition, then the action space should be configured accordingly. Therefore, material addition should be prioritized over material subtraction. This method is particularly beneficial if multiple material elements can be added in a single action, like the Bezier curve method.

This recommendation is limited to discrete grid domains. In geometrically parameterized domains, the action space should allow the agent to iteratively increase, decrease, or maintain the parameters to satisfy the objectives and constraints. These action spaces can either adjust the parameters individually or simultaneously.

Defining the reward function depends highly on a human user's understanding of the design problem. The reward function should incorporate the design objectives and constraints such that actions that achieve solutions closer to optimizing the objective should achieve higher rewards. If actions lead to solutions that do not satisfy the constraints, then a sizeable negative penalty should be introduced to teach the agent to avoid these actions.

The reward function can be comprised of positive or negative rewards. In the unit cell design problem, prompting the agent to minimize the magnitude of a negative reward was used in the material addition problems to ensure the agent was highly efficient and did not attempt to keep adding material to continue accumulating reward. Alternatively, the TO problem prompted the agent to maximize a positive reward as the agent should take the most actions to continually reduce the volume fraction. Based on these findings, a negative objective-based reward function should be used in material addition problems, and a positive objective-based reward function should be used in material subtraction problems. With combined addition and subtraction action spaces, a negative reward function should also be used to ensure the agent does not continue accumulating rewards for taking as many actions as possible without achieving a meaningful design. In geometrically parameterized domains, a negative reward function will also ensure the agent can efficiently achieve a high-performing solutions without being stuck in a loop of taking and undoing small parameter changes.

Regardless of positive or negative reward formulation, the reward function should undergo some form of normalization to ensure no significant disparities between the rewards for different state-action pairs. These disparities can cause exploding gradients in the training of the DRL agent, which can greatly inhibit the agent's performance. To avoid this exploding gradient threat, the reward function should be normalized, standardized, or paired with reward saltation (as in RQ1).

Termination thresholds should be established for all domains that allow the agent to terminate a design episode. This threshold should relate to the design objective. If the agent proposes a design that sufficiently satisfies the objective, the design episode should be terminated, and the agent should receive a significant reward bonus. This bonus will teach the agent that it has achieved a sufficiently high-performing design solution. The human user must specify this termination threshold depending on the design domain and required level of performance.

A critical exception to the negative reward paired with termination criteria should be if the environment is established to ensure the agent has to take a predetermined number of actions, and the final performance of a design is only evaluated after these steps are taken. A positive reward function should allow the agent to accumulate the maximum positive reward in this situation.

The final consideration for reformulating an engineering design problem as a DRL problem is the selection of the DRL algorithm. This can be the most challenging recommendation due to the constant influx of publications with new and improved DRL algorithms. Additionally, there was limited algorithm exploration in this dissertation. That said, selecting a DRL algorithm depends on the state and action space representations. With discrete state and action spaces, value-based algorithms will provide robust and easy-to-implement options. Alternatively, continuous action spaces will generally require actor-critic policy-based approaches. I highly recommend that the dissertation reader examine the literature for the current state of the art for DRL algorithms that can appropriately integrate your DRL environment's state and action spaces.

Additionally, ensure your computational resources will be capable of training the DNN that serves as the cornerstone of these algorithms. The choice of DNN depends on the representation of the state space observations. Observations represented as 2D arrays will require convolution filter layers, while 1D vectors will require fully connected dense layers. Network architecture and hyperparameter optimization are critical steps for successfully training a DRL agent. Therefore, Bayes's optimization of the number and size of the DNN layers and hyperparameters (learning rate, discount factor, action space noise, batch size, etc.) is critical to ensure improved model performance. Baye's optimization cannot guarantee optimal architectures or hyperparameters but will ensure appropriate steps are taken for a high-performing model.

These recommendations are derived from my experiences with incorporating DRL environments to achieve high-performing design solutions in several structural topology design problems. These recommendations may not apply to all design problems but should give human users a strong starting point for environment creation. The user should creatively and iteratively refine the components to produce a DRL environment rich with information that can teach a DRL agent to act as the powerful co-designer.

CHAPTER SEVEN

CONCLUSIONS

This dissertation aims to validate that deep reinforcement learning methods can be a highly valuable engineering design tool for various domains. Deep reinforcement learning offers a gradient-free optimization approach that achieves high-performing design solutions by teaching an agent to complete a task by taking actions and receiving feedback in an interactive environment. This work explicitly targets the domain of structural topologies. In the context of this work, structural topologies are the connections of solid or voided material within a continuous structure or subsystem. Therefore, the agent is tasked with altering the material layout within a topology to satisfy an objective. Three unique design problems were introduced to validate the effectiveness of using deep reinforcement learning as a design tool.

The first design problem was addressed by showing that a deep reinforcement learning agent can successfully optimize discretized 2D topologies similarly to traditional gradient-based topology optimization. The deep reinforcement learning environment was structured to allow an agent to sequentially remove elements from a solid topology and be rewarded for actions that lead to a minimal increase in strain energy while decreasing the volume fraction of the topology. This multi-objective reward function was built to mimic the multi-objective nature of compliance minimization topology optimization. The agent was trained to design optimal topologies at a 6x6 topology size with randomly selected loaded and bounded elements. A multi-step progressive refinement approach was used during testing to improve the detailed representation of the topologies from 6x6 to 12x12 to 24x24 without retraining the agent. The agent was tested on a series of common load cases, including cases the agent had not seen during training. The results showed that after training, the deep reinforcement learning agent adopted a generalized design strategy that could generate high-performing topologies similar to the output of gradient-based topology optimizers. The results imply that a deep reinforcement learning agent possesses generalized performance capabilities for a complex engineering design task.

The second phase of this dissertation work expands into a more complex design domain by showing that a deep reinforcement learning agent could successfully design mechanical metamaterials that exhibit targeted nonlinear deformation responses. The deep reinforcement learning environment was structured to allow an agent to design latent space-encoded metamaterial unit cells by sequentially adding material elements to a starting blank design domain according to Cubic Bezier curve functions. The agent was rewarded for producing unit cells that, when tessellated, have a surrogate-model predicted force-displacement response similar to a desired response.

Separate agents were trained to design for compressive and tensile loading. Both agents were tested on a series of force-displacement curves with varying degrees of nonlinearity they had not seen during training. The results showed that after training, the deep reinforcement learning agents adopted a generalized design strategy that could sequentially design metamaterial unit cells with force-displacement responses with mean absolute percent errors of 5.3% and 8.7% compared to the desired response for the tensile and compressive agent, respectively. The results imply that a deep reinforcement learning agent possesses generalized performance capabilities for designing rapid, on-demand mechanical metamaterials with customizable nonlinearity.

The final phase of this research expanded on the second to increase the metamaterial capabilities by designing metamaterials with customizable deformation and energy return capabilities and improve the world-world applicabilities of the design domain by introducing a constitutive material model calibrated against thermoplastic polyurethane (TPU) experimental data. The agent designed the metamaterials using a similar environment as the second phase of this manuscript but was rewarded for producing metamaterials with a surrogate-model predicted compressive loading stress-strain curve similar to a desired response and minimizing or maximizing the energy return upon unloading.

The agent was tested by designing mechanical metamaterials that mimic the compressive response of the expanded-thermoplastic polyurethane (E-TPU) material Infingergy® while maximizing or minimizing hysteric energy loss in compressive cyclic loading. These optimized designs were additively manufactured using a TPU filament and a fused deposition modeling (FDM) printer and experimentally tested. The experimental results show that the optimized designs achieve a highly similar loading response as the E-TPU material while being able to customize the energy return. These results show that the strategic design of mechanical metamaterials can result in materials that exhibit customizable deformation and energy manipulation responses.

The deep reinforcement learning agent was also tested on an array of desired loading deformation responses it had not seen during training while randomly being prompted to minimize or maximize hysteresis. The agent designed metamaterials that exhibited deformation responses with an average mean absolute percent error of 9.23% compared to the desired response. Additionally, 95.4% and 81.8% of the hysteresis-maximizing and -minimizing designs had energy return percentages in at least the 90th percentile of feasible designs. The results imply that a deep reinforcement learning agent possesses generalized performance capabilities for designing rapid, on-demand mechanical metamaterials with customizable deformation and energy return characteristics.

This work aims to show that formulating complex engineering design problems as a deep reinforcement learning optimization problem can lead to high-performing solutions. By interacting with an environment that encodes the core aspects of the problem, deep reinforcement learning agents can discover hidden relationships and generate non-intuitive optimal designs that may not be apparent to a human designer for a range of design problems. This work aims to inspire further research into applying deep reinforcement learning as a high-level co-designer for solving arbitrarily complex design problems across many domains.

REFERENCES

[1] A. Rodrigues Da Silva, “Model-driven engineering: A survey supported by the unified conceptual model,” *Comput. Lang. Syst. Struct.*, vol. 43, pp. 139–155, 2015, doi: 10.1016/j.cl.2015.06.001.

[2] L. T. Biegler, O. Ghattas, M. Heinkenschloss, and B. van Bloemen Waanders, “Large-Scale PDE-Constrained Optimization: An Introduction,” in *Large-Scale PDE-Constrained Optimization*, 2003, pp. 3–13.

[3] P. Ascencio, A. Astolfi, and T. Parisini, “Backstepping PDE Design: A convex optimization approach,” *IEEE Trans. Automat. Contr.*, vol. 63, no. 7, pp. 1943–1958, 2018, doi: 10.1109/TAC.2017.2757088.

[4] A. Kawamoto, T. Matsumori, S. Yamasaki, T. Nomura, T. Kondoh, and S. Nishiwaki, “Heaviside projection based topology optimization by a PDE-filtered scalar function,” *Struct. Multidiscip. Optim.*, vol. 44, no. 1, pp. 19–24, 2011, doi: 10.1007/s00158-010-0562-2.

[5] C. Dunbar and G. Qu, “Designing trusted embedded systems from finite state machines,” *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 5, 2014, doi: 10.1145/2638555.

[6] E. Börger, “The Abstract State Machines Method for High-Level System Design and Analysis,” in *Formal Methods: State of the Art and New Directions*, P. Boca, J. P. Bowen, and J. Siddiqi, Eds. London: Springer London, 2010, pp. 79–116.

[7] H. Elmqvist, F. Gaucher, S. E. Mattson, and F. Dupont, “State Machines in Modelica,” *Proc. 9th Int. Model. Conf. Sept. 3-5, 2012, Munich, Ger.*, vol. 76, pp. 37–46, 2012, doi: 10.3384/ecp1207637.

[8] M. Lu, A. Mohammadi, Z. Meng, X. Meng, G. Li, and Z. Li, “Deep neural operator for learning transient response of interpenetrating phase composites subject to dynamic loading,” *Comput. Mech.*, pp. 1–23, 2023, doi: 10.1007/s00466-023-02343-6.

[9] G. Sun and S. Wang, “A review of the artificial neural network surrogate modeling in aerodynamic design,” *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.*, vol. 233, no. 16, pp. 5863–5872, 2019, doi: 10.1177/0954410019864485.

[10] Q. Xu, E. Wehrle, and H. Baier, “Knowledge-Based Surrogate Modeling in Engineering Design Optimization,” in *Surrogate-Based Modeling and Optimization: Applications in Engineering*, S. Koziel and L. Leifsson, Eds. New York, NY: Springer New York, 2013, pp. 313–336.

[11] L. Hawchar, C. P. El Soueidy, and F. Schoefs, “Global kriging surrogate modeling for general time-variant reliability-based design optimization problems,” *Struct. Multidiscip. Optim.*, vol. 58, no. 3, pp. 955–968, 2018, doi: 10.1007/s00158-018-1938-y.

[12] F. Ding and A. Kareem, “A multi-fidelity shape optimization via surrogate modeling for

civil structures," *J. Wind Eng. Ind. Aerodyn.*, vol. 178, no. December 2017, pp. 49–56, 2018, doi: 10.1016/j.jweia.2018.04.022.

[13] V. Quintana, L. Rivest, R. Pellerin, F. Venne, and F. Kheddouci, "Will Model-based Definition replace engineering drawings throughout the product lifecycle? A global perspective from aerospace industry," *Comput. Ind.*, vol. 61, no. 5, pp. 497–508, 2010, doi: 10.1016/j.compind.2010.01.005.

[14] J. A. Busby and P. A. Lloyd, "Influences on solution search processes in design organizations," *Res. Eng. Des. - Theory, Appl. Concurr. Eng.*, vol. 11, no. 3, pp. 158–171, 1999, doi: 10.1007/s001630050012.

[15] K. A. Saar, F. Giardina, and F. Iida, "Model-Free Design Optimization of a Hopping Robot and Its Comparison With a Human Designer," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 1245–1251, 2018.

[16] R. K. Arora, *Optimization: algorithms and applications*, vol. 53, no. 08. Boca Raton: CRC Press, 2015.

[17] T. Yang, "Advancing non-convex and constrained learning," *AI Matters*, vol. 5, no. 3, pp. 29–39, 2019, doi: 10.1145/3362077.3362085.

[18] H. Qiu and Y. Liu, "Novel heuristic algorithm for large-scale complex optimization," *Procedia Comput. Sci.*, vol. 80, pp. 744–751, 2016, doi: 10.1016/j.procs.2016.05.364.

[19] L. Regenwetter, A. H. Nobari, and F. Ahmed, "Deep Generative Models in Engineering Design: A Review," *ASME J. Mech. Des.*, 2021, [Online]. Available: http://arxiv.org/abs/2110.10863.

[20] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science (80-.).*, vol. 349, no. 6245, 2015.

[21] T. L. Tepla *et al.*, "Alloys selection based on the supervised learning technique for design of biocompatible medical materials," *Arch. Mater. Sci. Eng.*, vol. 1, no. 93, pp. 32–40, 2018, doi: 10.5604/01.3001.0012.6944.

[22] M. Abdullah and S. Koziel, "Supervised-Learning-Based Development of Multibit RCS-Reduced Coding Metasurfaces," *IEEE Trans. Microw. Theory Tech.*, vol. 70, no. 1, pp. 264–274, 2022.

[23] X. Lei, C. Liu, Z. Du, W. Zhang, and X. Guo, "Machine learning-driven real-time topology optimization under moving morphable component-based framework," *J. Appl. Mech. Trans. ASME*, vol. 86, no. 1, pp. 1–9, 2019, doi: 10.1115/1.4041319.

[24] C. Wen *et al.*, "Machine learning assisted design of high entropy alloys with desired property," *Acta Mater.*, vol. 170, pp. 109–117, 2019, doi: 10.1016/j.actamat.2019.03.010.

[25] W. Ma, F. Cheng, and Y. Liu, "Deep-Learning-Enabled On-Demand Design of Chiral Metamaterials," *ACS Nano*, vol. 12, no. 6, pp. 6326–6334, 2018, doi:

10.1021/acsnano.8b03569.

[26] H. T. Kollmann, D. W. Abueidda, S. Koric, E. Guleryuz, and N. A. Sobh, “Deep learning for topology optimization of 2D metamaterials,” *Mater. Des.*, vol. 196, p. 109098, 2020, doi: 10.1016/j.matdes.2020.109098.

[27] H. Huang, W. Xia, J. Xiong, J. Yang, G. Zheng, and X. Zhu, “Unsupervised Learning-Based Fast Beamforming Design for Downlink MIMO,” *IEEE Access*, vol. 7, pp. 7599–7605, 2019, doi: 10.1109/ACCESS.2018.2887308.

[28] L. Di, “Study on a new assembled monolithic RC structure Based on Unsupervised Learning Method,” *Int. Core J. Eng.*, vol. 3, no. 11, pp. 183–188, 2017.

[29] “UNSUPERVISED- LEARNING ASSISTED ARTIFICIAL NEURAL NETWORK FOR OPTIMIZATION by Varun Kote B . E . June 2016 , SIR M Visvesvarya Institute of Technology A Thesis Submitted to the Faculty of Old Dominion University in Partial Fulfillment of the Requirements ,” no. June 2016, 2019.

[30] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 9, no. 5. Cambridge, MA: MIT Press, 1998.

[31] D. Silver *et al.*, “Mastering the game of Go without human knowledge,” *Nature*, vol. 550, no. 7676, pp. 354–359, 2017, doi: 10.1038/nature24270.

[32] J. Schrittwieser *et al.*, “Mastering Atari, Go, chess and shogi by planning with a learned model,” *Nature*, vol. 588, no. 7839, pp. 604–609, 2020, doi: 10.1038/s41586-020-03051-4.

[33] J. Jumper *et al.*, “Highly accurate protein structure prediction with AlphaFold,” *Nature*, vol. 596, no. May, 2021, doi: 10.1038/s41586-021-03819-2.

[34] I. Akkaya *et al.*, “Solving Rubik’s Cube with a Robot Hand,” pp. 1–51, 2019, [Online]. Available: http://arxiv.org/abs/1910.07113.

[35] J. Degrave *et al.*, “Magnetic control of tokamak plasmas through deep reinforcement learning,” *Nature*, vol. 602, no. 7897, pp. 414–419, 2022, doi: 10.1038/s41586-021-04301-9.

[36] F. Dworschak, S. Dietze, M. Wittmann, B. Schleich, and S. Wartzack, “Reinforcement Learning for Engineering Design Automation,” *Adv. Eng. Informatics*, vol. 52, no. April, p. 101612, 2022, doi: 10.1016/j.aei.2022.101612.

[37] N. K. Brown, A. P. Garland, G. M. Fadel, and G. Li, “‘Deep reinforcement learning for engineering design through topology optimization of elementally discretized design domains,’” *Mater. Des.*, vol. 218, p. 110672, 2022, doi: 10.1016/j.matdes.2022.110672.

[38] A. Mirhoseini *et al.*, “A graph placement methodology for fast chip design,” *Nature*, vol. 594, no. 7862, pp. 207–212, 2021, doi: 10.1038/s41586-021-03544-w.

[39] C. Lin, T. Fan, W. Wang, and M. Nießner, “Modeling 3D Shapes by Reinforcement Learning,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12355 LNCS, pp. 545–561, 2020, doi: 10.1007/978-3-030-58607-2_32.

[40] F. Sui, R. Guo, Z. Zhang, G. X. Gu, and L. Lin, “Deep Reinforcement Learning for Digital Materials Design,” *ACS Mater. Lett.*, vol. 3, no. 10, pp. 1433–1439, 2021, doi: 10.1021/acsmaterialslett.1c00390.

[41] R. Bellman, “Markovian decision processes,” *Mathematics in Science and Engineering*, vol. 130, no. C. pp. 172–187, 1977, doi: 10.1016/S0076-5392(08)61190-X.

[42] Z. Hu, K. Wan, X. Gao, and Y. Zhai, “A Dynamic Adjusting Reward Function Method for Deep Reinforcement Learning with Adjustable Parameters,” *Math. Probl. Eng.*, vol. 2019, 2019, doi: 10.1155/2019/7619483.

[43] A. El Sallab, M. Abdou, E. Perot, and S. Yogamani, “Deep reinforcement learning framework for autonomous driving,” *IS T Int. Symp. Electron. Imaging Sci. Technol.*, pp. 70–76, 2017, doi: 10.2352/ISSN.2470-1173.2017.19.AVM-023.

[44] M. Lapan, *Deep Reinforcement Learning*, 2nd ed. Birmingham, UK: Packt Publishing, 2020.

[45] D. Hadfield-Menell, S. Milli, P. Abbeel, S. Russell, and A. D. Dragan, “Inverse reward design,” *Adv. Neural Inf. Process. Syst.*, vol. 2017-Decem, no. Nips, pp. 6766–6775, 2017.

[46] T. D. Kulkarni, K. R. Narasimhan, A. Saeedi, and J. B. Tenenbaum, “Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation,” *Adv. Neural Inf. Process. Syst.*, pp. 3682–3690, 2016.

[47] D. Rengarajan, G. Vaidya, A. Sarvesh, D. Kalathil, and S. Shakkottai, “Reinforcement Learning with Sparse Rewards using Guidance from Offline Demonstration,” pp. 1–21, 2022, [Online]. Available: http://arxiv.org/abs/2202.04628.

[48] D. Silver, S. Singh, D. Precup, and R. S. Sutton, “Reward is enough,” *Artif. Intell.*, vol. 299, p. 103535, 2021, doi: 10.1016/j.artint.2021.103535.

[49] O. Sigmund and K. Maute, “Topology optimization approaches: A comparative review,” *Struct. Multidiscip. Optim.*, vol. 48, no. 6, pp. 1031–1055, 2013, doi: 10.1007/s00158-013-0978-6.

[50] M. Kociecki and H. Adeli, “Two-phase genetic algorithm for topology optimization of free-form steel space-frame roof structures with complex curvatures,” *Eng. Appl. Artif. Intell.*, vol. 32, pp. 218–227, 2014, doi: 10.1016/j.engappai.2014.01.010.

[51] K. Hayashi and M. Ohsaki, “Reinforcement Learning and Graph Embedding for Binary Truss Topology Optimization Under Stress and Displacement Constraints,” *Front. Built Environ.*, vol. 6, no. April, pp. 1–15, 2020, doi: 10.3389/fbuil.2020.00059.

[52] S. Y. Wang and K. Tai, “Structural topology design optimization using Genetic Algorithms with a bit-array representation,” *Comput. Methods Appl. Mech. Eng.*, vol. 194, no. 36–38, pp. 3749–3770, 2005, doi: 10.1016/j.cma.2004.09.003.

[53] X. Yu, J. Zhou, H. Liang, Z. Jiang, and L. Wu, “Mechanical metamaterials associated with stiffness, rigidity and compressibility: A brief review,” *Prog. Mater. Sci.*, vol. 94, pp. 114–173, 2018, doi: 10.1016/j.pmatsci.2017.12.003.

[54] U. D. Larsen, O. Sigmund, and S. Bouwstra, “Design and fabrication of compliant micromechanisms and structures with negative Poisson’s ratio,” *J. Microelectromechanical Syst.*, vol. 6, no. 2, pp. 99–106, 1997, doi: 10.1109/84.585787.

[55] A. Montalbano, G. M. Fadel, and G. Li, “Design for energy absorption using snap-through bistable metamaterials,” *Mech. Based Des. Struct. Mach.*, vol. 51, no. 3, pp. 1368–1386, 2023, doi: 10.1080/15397734.2020.1867167.

[56] O. Sigmund and S. Torquato, “Design of materials with extreme thermal expansion using a three-phase topology optimization method,” *J. Mech. Phys. Solids*, vol. 45, no. 6, pp. 1037–1067, 1997, doi: 10.1016/S0022-5096(96)00114-7.

[57] X. Wang, Y. Qin, and X. Zhang, “Concurrent Analysis and Design of Structure and Its Periodic Materials,” *Acta Mech. Solida Sin.*, vol. 29, no. 6, pp. 663–674, 2016, doi: 10.1016/S0894-9166(16)30336-6.

[58] A. Alla, M. Hinze, P. Kolvenbach, O. Lass, and S. Ulbrich, “A certified model reduction approach for robust parameter optimization with PDE constraints,” *Adv. Comput. Math.*, vol. 45, no. 3, pp. 1221–1250, 2019, doi: 10.1007/s10444-018-9653-1.

[59] A. Rajeev *et al.*, “Parametric optimization of corner radius in hexagonal honeycombs under in-plane compression,” *J. Manuf. Process.*, vol. 79, no. April, pp. 35–46, 2022, doi: 10.1016/j.jmapro.2022.04.041.

[60] I. A. Fotiou, P. Rostalski, B. Sturmfels, and M. Morari, “An algebraic geometry approach to nonlinear parametric optimization in control,” *Proc. Am. Control Conf.*, vol. 2006, no. 4, pp. 3618–3623, 2006, doi: 10.1109/acc.2006.1657280.

[61] W. Zhu, L. Guo, Z. Jia, D. Tian, and Y. Xiong, “A Surrogate-Model-Based Approach for the Optimization of the Thermal Design Parameters of Space Telescopes,” *Appl. Sci.*, vol. 12, no. 3, 2022, doi: 10.3390/app12031633.

[62] W. Zhang, X. Gong, S. Xuan, and W. Jiang, “Temperature-dependent mechanical properties and model of magnetorheological elastomers,” *Ind. Eng. Chem. Res.*, vol. 50, no. 11, pp. 6704–6712, 2011, doi: 10.1021/ie200386x.

[63] S. Xiaofei, K. Hrishikesh, and T. Lih-Sheng, “Fabrication of Highly Expanded thermoplastic Polyurethane Foams Using Microcellular Injection Modling and Gas-Laden Pellets,” *Polym. Eng. Sci.*, pp. 2643–2652, 2015, doi: 10.1002/pen.

[64] H. J. Qi and M. C. Boyce, “Stress-strain behavior of thermoplastic polyurethanes,” *Mech. Mater.*, vol. 37, no. 8, pp. 817–839, 2005, doi: 10.1016/j.mechmat.2004.08.001.

[65] K. Yonekura and H. Hattori, “Framework for design optimization using deep reinforcement learning,” *Struct. Multidiscip. Optim.*, vol. 60, no. 4, pp. 1709–1713, 2019, doi: 10.1007/s00158-019-02276-w.

[66] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolic, “AutoCkt: Deep Reinforcement Learning of Analog Circuit Designs,” *Proc. 2020 Des. Autom. Test Eur. Conf. Exhib. DATE 2020*, pp. 490–495, 2020, doi: 10.23919/DATE48585.2020.9116200.

[67] B. Liu, L. Xu, and J. Huang, “Reinforcement learning approach to thermal transparency with particles in periodic lattices,” *J. Appl. Phys.*, vol. 130, no. 4, 2021, doi: 10.1063/5.0054023.

[68] P. Rajak *et al.*, “Autonomous reinforcement learning agent for stretchable kirigami design of 2D materials,” *npj Comput. Mater.*, vol. 7, no. 1, 2021, doi: 10.1038/s41524-021-00572-y.

[69] Martin Philip Bendsoe and Noboru Kikuchi, “Generating optimal topologies in structural design using a homogenization method,” *Comput. Methods Appl. Mech. Eng.*, vol. 71, pp. 197–224, 1988.

[70] M. P. Bendsøe, “Optimal shape design as a material distribution problem,” *Struct. Optim.*, vol. 1, no. 4, pp. 193–202, 1989, doi: 10.1007/BF01650949.

[71] A. Chandrasekhar and K. Suresh, “TOuNN: Topology Optimization using Neural Networks,” *Struct. Multidiscip. Optim.*, vol. 63, pp. 1135–1149, 2021, doi: 10.1016/j.cad.2021.103017.

[72] D. Wang, C. Xiang, Y. Pan, A. Chen, X. Zhou, and Y. Zhang, “A deep convolutional neural network for topology optimization with perceptible generalization ability,” *Eng. Optim.*, 2021, doi: 10.1080/0305215X.2021.1902998.

[73] E. Ulu, R. Zhang, and L. B. Kara, “A data-driven investigation and estimation of optimal topologies under variable loading configurations,” *Comput. Methods Biomech. Biomed. Eng. Imaging Vis.*, vol. 4, no. 2, pp. 61–72, 2016, doi: 10.1080/21681163.2015.1030775.

[74] I. Sosnovik and I. Oseledets, “Neural networks for topology optimization,” *Russ. J. Numer. Anal. Math. Model.*, vol. 34, no. 4, pp. 215–223, 2019, doi: 10.1515/rnam-2019-0018.

[75] H. Chi *et al.*, “Universal machine learning for topology optimization,” *Comput. Methods Appl. Mech. Eng.*, vol. 375, p. 112739, 2021, doi: 10.1016/j.cma.2019.112739.

[76] S. Nanda, P. K. Sahu, and R. K. Mishra, “Inverse artificial neural network modeling for metamaterial unit cell synthesis,” *J. Comput. Electron.*, vol. 18, no. 4, pp. 1388–1399, 2019, doi: 10.1007/s10825-019-01371-x.

[77] L. Wang, T. Wang, Y. Nie, and R. Gong, “Synthesis design of metamaterial absorbers using a genetic algorithm,” *Conf. Proc. Int. Symp. Signals, Syst. Electron.*, vol. 1, no. 1, pp. 39–42, 2010, doi: 10.1109/ISSSE.2010.5607053.

[78] L. Wang, Y. C. Chan, Z. Liu, P. Zhu, and W. Chen, “Data-driven metamaterial design with Laplace-Beltrami spectrum as ‘shape-DNA,’” *Struct. Multidiscip. Optim.*, vol. 61, no. 6, pp. 2613–2628, 2020, doi: 10.1007/s00158-020-02523-5.

[79] C. Ma *et al.*, “Accelerated design and characterization of non-uniform cellular materials via a machine-learning based framework,” *npj Comput. Mater.*, vol. 6, no. 1, 2020, doi: 10.1038/s41524-020-0309-6.

[80] P. Jiao and A. H. Alavi, “Artificial intelligence-enabled smart mechanical metamaterials: advent and future trends,” *Int. Mater. Rev.*, vol. 66, no. 6, pp. 365–393, 2021, doi: 10.1080/09506608.2020.1815394.

[81] C. Sharpe, C. C. Seepersad, S. Watts, and D. Tortorelli, “Design of mechanical metamaterials via constrained Bayesian optimization,” *Proc. ASME Des. Eng. Tech. Conf.*, vol. 2A-2018, pp. 1–11, 2018, doi: 10.1115/DETC2018-85270.

[82] F. Wang, O. Sigmund, and J. S. Jensen, “Design of materials with prescribed nonlinear properties,” *J. Mech. Phys. Solids*, vol. 69, no. 1, pp. 156–174, 2014, doi: 10.1016/j.jmps.2014.05.003.

[83] Z. Satterfield, N. Kulkarni, G. Fadel, G. Li, N. Coutris, and M. P. Castanier, “Unit cell synthesis for design of materials with targeted nonlinear deformation response,” *J. Mech. Des. Trans. ASME*, vol. 139, no. 12, pp. 14–16, 2017, doi: 10.1115/1.4037894.

[84] N. Kulkarni, S. J. Franklin, G. Fadel, G. Li, N. Coutris, and M. P. Castanier, “Multiobjective design of meta-materials exhibiting a targeted non-linear deformation response,” *Int. J. Interact. Des. Manuf.*, vol. 14, no. 4, pp. 1357–1377, 2020, doi: 10.1007/s12008-020-00707-3.

[85] Q. Zeng, S. Duan, Z. Zhao, P. Wang, and H. Lei, “Inverse Design of Energy-Absorbing Metamaterials by Topology Optimization,” *Adv. Sci.*, vol. 10, no. 4, pp. 1–10, 2023, doi: 10.1002/advs.202204977.

[86] R. Behrou *et al.*, “Topology optimization of nonlinear periodically microstructured materials for tailored homogenized constitutive properties,” *Compos. Struct.*, vol. 266, no. June 2020, p. 113729, 2021, doi: 10.1016/j.compstruct.2021.113729.

[87] X. Tan *et al.*, “Real-time tunable negative stiffness mechanical metamaterial,” *Extrem. Mech. Lett.*, vol. 41, p. 100990, 2020, doi: 10.1016/j.eml.2020.100990.

[88] X. Tan *et al.*, “Novel multidirectional negative stiffness mechanical metamaterials,” *Smart Mater. Struct.*, vol. 29, no. 1, 2020, doi: 10.1088/1361-665X/ab47d9.

[89] H. Yang and L. Ma, “Multi-stable mechanical metamaterials by elastic buckling

instability," *J. Mater. Sci.*, vol. 54, no. 4, pp. 3509–3526, 2019, doi: 10.1007/s10853-018-3065-y.

[90] R. Gao, S. Guo, X. Tian, and S. Liu, "A negative-stiffness based 1D metamaterial for bidirectional buffering and energy absorption with state recoverable characteristic," *Thin-Walled Struct.*, vol. 169, no. February, p. 108319, 2021, doi: 10.1016/j.tws.2021.108319.

[91] H. Al-Rifaie and W. Sumelka, "The development of a new shock absorbing uniaxial graded auxetic damper (UGAD)," *Materials (Basel).*, vol. 12, no. 16, 2019, doi: 10.3390/ma12162573.

[92] X. W. Zhang and D. Q. Yang, "Numerical and Experimental Studies of a Light-Weight Auxetic Cellular Vibration Isolation Base," *Shock Vib.*, vol. 2016, 2016, doi: 10.1155/2016/4017534.

[93] L. Jin *et al.*, "Guided transition waves in multistable mechanical metamaterials," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 117, no. 5, pp. 2319–2325, 2020, doi: 10.1073/pnas.1913228117.

[94] E. Kim, Y. Han, N. Kim, and J. Yang, "Wave Propagation in Woodpile Mechanical Metamaterials," *2014 Int. Symp. Optomechatronic Technol. ISOT 2014*, pp. 12–14, 2014, doi: 10.1109/ISOT.2014.12.

[95] B. Deng, J. R. Raney, K. Bertoldi, and V. Tournat, "Nonlinear waves in flexible mechanical metamaterials," *J. Appl. Phys.*, vol. 130, no. 4, 2021, doi: 10.1063/5.0050271.

[96] A. P. Garland *et al.*, "Coulombic friction in metamaterials to dissipate mechanical energy," *Extrem. Mech. Lett.*, vol. 40, p. 100847, 2020, doi: 10.1016/j.eml.2020.100847.

[97] R. Hamzehei *et al.*, "Parrot Beak-Inspired Metamaterials with Friction and Interlocking Mechanisms 3D/4D Printed in Micro and Macro Scales for Supreme Energy Absorption/Dissipation," *Adv. Eng. Mater.*, vol. 2201842, 2023, doi: 10.1002/adem.202201842.

[98] Q. Chen *et al.*, "Topology Optimization of Metamaterials for Energy Dissipation," *MARSS 2018 - Int. Conf. Manip. Autom. Robot. Small Scales*, pp. 1–6, 2018, doi: 10.1109/MARSS.2018.8481152.

[99] H. Deng, L. Cheng, X. Liang, D. Hayduke, and A. C. To, "Topology optimization for energy dissipation design of lattice structures through snap-through behavior," *Comput. Methods Appl. Mech. Eng.*, vol. 358, p. 112641, 2020, doi: 10.1016/j.cma.2019.112641.

[100] C. Conlan-Smith and K. A. James, "A stress-based topology optimization method for heterogeneous structures," *Struct. Multidiscip. Optim.*, vol. 60, no. 1, pp. 167–183, 2019, doi: 10.1007/s00158-019-02207-9.

[101] G. Hailu Shimels, W. Dereje Engida, and H. Fakhruldin Mohd, "A comparative study on stress and compliance based structural topology optimization," *IOP Conf. Ser. Mater. Sci.*

Eng., vol. 241, no. 1, 2017, doi: 10.1088/1757-899X/241/1/012003.

[102] G. Li, “Introduction to the Finite Element Method and Implementation with MATLAB®,” *Introd. to Finite Elem. Method Implement. with MATLAB*®, 2020, doi: 10.1017/9781108559058.

[103] J. Sola and J. Sevilla, “Importance of input data normalization for the application of neural networks to complex industrial problems,” *IEEE Trans. Nucl. Sci.*, vol. 44, no. 3 PART 3, pp. 1464–1468, 1997, doi: 10.1109/23.589532.

[104] H. Van Hasselt, A. Guez, and D. Silver, “Double DQN.pdf,” *Proc. 30th AAAI Conf. Artif. Intell.*, pp. 2094–2100, 2016.

[105] I. Y. Kim and O. L. De Weck, “Variable chromosome length genetic algorithm for progressive refinement in topology optimization,” *Struct. Multidiscip. Optim.*, vol. 29, no. 6, pp. 445–456, 2005, doi: 10.1007/s00158-004-0498-5.

[106] C. Le, J. Norato, T. Bruns, C. Ha, and D. Tortorelli, “Stress-based topology optimization for continua,” *Struct. Multidiscip. Optim.*, vol. 41, no. 4, pp. 605–620, 2010, doi: 10.1007/s00158-009-0440-y.

[107] O. Sigmund, “A 99 line topology optimization code written in Matlab,” Springer-Verlag, 2001. Accessed: Feb. 14, 2019. [Online]. Available: http://www.topopt.dtu.dk.

[108] J. Mason, “Developing strategic thinking,” *Long Range Plann.*, vol. 19, no. 3, pp. 72–80, 1986, doi: 10.1016/0024-6301(86)90201-3.

[109] R. Ansola Loyola, O. M. Querin, A. Garaigordobil Jiménez, and C. Alonso Gordoa, “A sequential element rejection and admission (SERA) topology optimization code written in Matlab,” *Struct. Multidiscip. Optim.*, vol. 58, no. 3, pp. 1297–1310, 2018, doi: 10.1007/s00158-018-1939-x.

[110] W. Yang, S. L. Ho, and W. N. Fu, “A multiscale topology optimization methodology based on sequential element rejection-admission and boundary element evolvement,” *IEEE Trans. Magn.*, vol. 55, no. 6, pp. 2019–2022, 2019, doi: 10.1109/TMAG.2019.2895458.

[111] H. A. Eschenauer and N. Olhoff, “Topology optimization of continuum structures: A review,” *Appl. Mech. Rev.*, vol. 54, no. 4, pp. 331–390, 2001, doi: 10.1115/1.1388075.

[112] Z. T. Satterfield, “Design of a MetaMaterial with Targeted Nonlinear Deformation Response,” Clemson University, 2015.

[113] W. Zhongke, L. Feng, S. H. Soon, and C. K. Yun, “Evaluation of difference bounds for computing rational Bézier curves and surfaces,” *Comput. Graph.*, vol. 28, no. 4, pp. 551–558, 2004, doi: 10.1016/j.cag.2004.04.010.

[114] J. Fazil and V. Jayakumar, “Investigation of airfoil profile design using reverse engineering Bezier curve,” *J. Eng. Appl. Sci.*, vol. 6, no. 7, pp. 43–52, 2011.

[115] H. N. Fitter, A. B. Pandey, D. D. Patel, and J. M. Mistry, “A review on approaches for handling Bezier curves in CAD for Manufacturing,” *Procedia Eng.*, vol. 97, pp. 1155–1166, 2014, doi: 10.1016/j.proeng.2014.12.394.

[116] A. Álvarez-Trejo, E. Cuan-Urquizo, A. Roman-Flores, L. G. Trapaga-Martinez, and J. M. Alvarado-Orozco, “Bézier-based metamaterials: Synthesis, mechanics and additive manufacturing,” *Mater. Des.*, vol. 199, p. 109412, 2021, doi: 10.1016/j.matdes.2020.109412.

[117] C. Czech, “Design of Meta-Materials Outside the Homogenization Limit Using Multiscale Analysis and Topology Optimization,” *Ph.D. Diss. Clemson Univ.*, pp. 10–13, 112–116, 2012.

[118] O. Guéant and I. Manziuk, “Deep Reinforcement Learning for Market Making in Corporate Bonds: Beating the Curse of Dimensionality,” *Appl. Math. Financ.*, vol. 26, no. 5, pp. 387–452, 2019, doi: 10.1080/1350486X.2020.1714455.

[119] W. Curran, T. Brys, M. Taylor, and W. Smart, “Using PCA to Efficiently Represent State Spaces,” pp. 1–8, 2015, [Online]. Available: http://arxiv.org/abs/1505.00322.

[120] H. Van Hoof, N. Chen, M. Karl, P. Van Der Smagt, and J. Peters, “Stable reinforcement learning with autoencoders for tactile and visual data,” *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2016-Novem, pp. 3928–3934, 2016, doi: 10.1109/IROS.2016.7759578.

[121] B. Prakash, M. Horton, N. R. Waytowich, W. D. Hairston, T. Oates, and T. Mohsenin, “On the use of deep autoencoders for efficient embedded reinforcement learning,” *Proc. ACM Gt. Lakes Symp. VLSI, GLSVLSI*, pp. 507–512, 2019, doi: 10.1145/3299874.3319493.

[122] F. Kuo and I. Sloan, “Lifting the curse of Dimensionality,” *Not. AMS*, vol. 52, no. 11, pp. 1320–1328, 2015, doi: 10.1038/scientificamerican0515-50.

[123] W. Wu, P. Liu, and Z. Kang, “A novel mechanical metamaterial with simultaneous stretching- and compression-expanding property,” *Mater. Des.*, vol. 208, p. 109930, 2021, doi: 10.1016/j.matdes.2021.109930.

[124] L. P. Aggarwal, “Data augmentation in dermatology image recognition using machine learning,” *Ski. Res. Technol.*, vol. 25, no. 6, pp. 815–820, 2019, doi: 10.1111/srt.12726.

[125] L. Wang, Y. C. Chan, F. Ahmed, Z. Liu, P. Zhu, and W. Chen, “Deep generative modeling for mechanistic-based learning and design of metamaterial systems,” *Comput. Methods Appl. Mech. Eng.*, vol. 372, pp. 1–41, 2020, doi: 10.1016/j.cma.2020.113377.

[126] T. P. Lillicrap *et al.*, “Continuous control with deep reinforcement learning,” in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[127] H. E. P. With, U. Fail, U. Environments, and W. Sparse, “The Problem With Ddpg :

Understanding Fail - Ures in Deterministic Environments With," 2020.

[128] C. Hu and M. Xu, "Adaptive Exploration Strategy with Multi-Attribute Decision-Making for Reinforcement Learning," *IEEE Access*, vol. 8, pp. 32353–32364, 2020, doi: 10.1109/ACCESS.2020.2973169.

[129] K. Emancipator and M. H. Kroll, "A quantitatvie measure of nonlinearity," *Clin. Chem.*, vol. 39, no. 5, pp. 766–772, 1993.

[130] N. K. Brown, A. P. Garland, G. M. Fadel, and G. Li, "Deep Reinforcement Learning for the Rapid On-Demand Design of Mechanical Metamaterials with Targeted Nonlinear Deformation Responses," *Eng. Appl. Artif. Intell.*, vol. 126, 2023.

[131] F. Shen *et al.*, "Energy Absorption of Thermoplastic Polyurethane Lattice Structures via 3D Printing: Modeling and Prediction," *Int. J. Appl. Mech.*, vol. 8, no. 7, 2016, doi: 10.1142/S1758825116400068.

[132] I. Fachtagung, "E-TPU Datasheet," 2016.

[133] M. Smith, *Abaqus/Standard User's Manual*, 6.9. Providence, RI: Dassault Systems Simulia Corp.

[134] I. T. Jollife and J. Cadima, "Principal component analysis: A review and recent developments," *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.*, vol. 374, no. 2065, 2016, doi: 10.1098/rsta.2015.0202.

[135] P. Awasthi and S. S. Banerjee, "Fused deposition modeling of thermoplastic elastomeric materials: Challenges and opportunities," *Addit. Manuf.*, vol. 46, no. July, p. 102177, 2021, doi: 10.1016/j.addma.2021.102177.

[136] S. C. Fay, "Optimizing Shoe Midsoles for Running Performance by," Massachusetts Institute of Technology, 2021.

[137] W. Hoogkamer, S. Kipp, J. H. Frank, E. M. Farina, G. Luo, and R. Kram, "A Comparison of the Energetic Cost of Running in Marathon Racing Shoes," *Sport. Med.*, vol. 48, no. 4, pp. 1009–1019, 2018, doi: 10.1007/s40279-017-0811-2.

[138] J. Balli, S. Kumpaty, and V. Anewenter, "Continuous liquid interface production of 3D objects: An unconventional technology and its challenges and opportunities," *ASME Int. Mech. Eng. Congr. Expo. Proc.*, vol. 5, pp. 1–6, 2017, doi: 10.1115/IMECE2017-71802.

[139] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *35th Int. Conf. Mach. Learn. ICML 2018*, vol. 5, pp. 2976–2989, 2018.

[140] B. Eysenbach, J. Ibarz, A. Gupta, and S. Levine, "Diversity is all you need: Learning skills without a reward function," *7th Int. Conf. Learn. Represent. ICLR 2019*, pp. 1–22, 2019.

[141] K. Zhou, Y. Qiao, and T. Xiang, "Deep reinforcement learning for unsupervised video

summarization with diversity-representativeness reward," *32nd AAAI Conf. Artif. Intell. AAAI 2018*, pp. 7582–7589, 2018, doi: 10.1609/aaai.v32i1.12255.

Appendix A

Individual Experimental Results

The loading and unloading stress-strain responses for the three samples of the three randomly produced metamaterial designs have been included. These designs were used to calibrate the *Abaqus* material model.

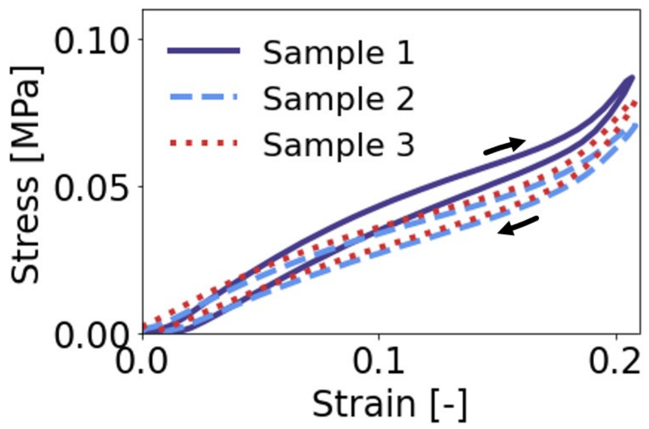


Figure A- 1. Metamaterial Design 1 experimental results from the three additively manufactured samples

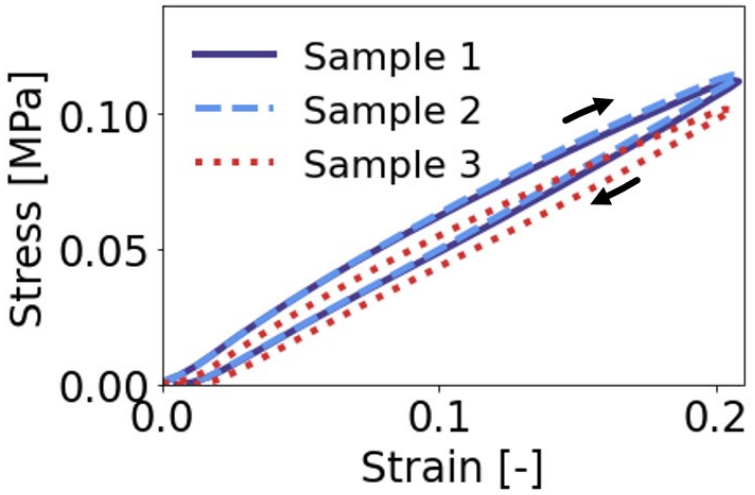


Figure A-2. Metamaterial Design 2 experimental results from the three additively manufactured samples

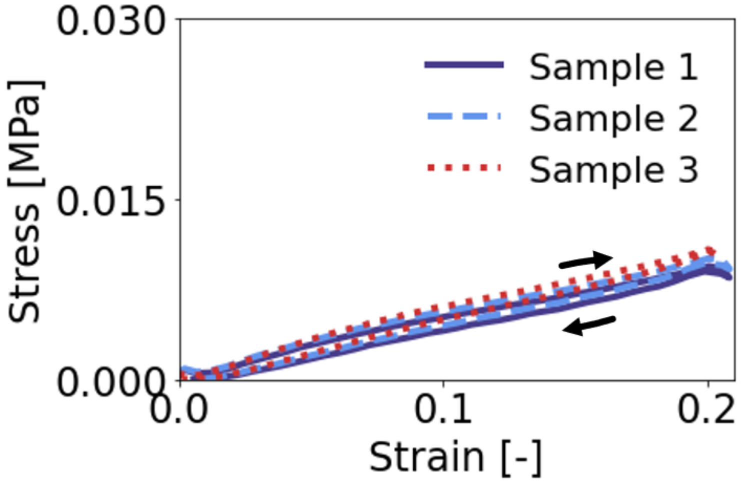


Figure A-3. Metamaterial Design 3 experimental results from the three additively manufactured samples