Deep Reinforcement Learning for Continuous Control of Material Thickness

Oliver Dippel[1,2][0000−0002−6252−2248], Alexei Lisitsa[1][0000−0002−3820−643X], and Bei Peng[1][0000−0003−0152−3180]

[1] University of Liverpool, Liverpool, United Kingdom
[2] Centre for Doctoral Training in Distributed Algorithms, Liverpool, United Kingdom
{oliver.dippel, lisitsa, bei.peng}@liverpool.ac.uk

Abstract. To achieve the desired quality standards of certain manufactured materials, the involved parameters are still adjusted by knowledge-based procedures according to human expertise, which can be costly and time-consuming. To optimize operational efficiency and provide decision support for human experts, we develop a general continuous control framework that utilizes deep reinforcement learning (DRL) to automatically determine the main control parameters, in situations where simulation environments are unavailable and traditional PID controllers are not viable options. In our work, we aim to automatically learn the key control parameters to achieve the desired outlet thickness of the manufactured material. We first construct a surrogate environment based on real-world expert trajectories obtained from the true underlining manufacturing process to achieve this. Subsequently, we train a DRL agent within the surrogate environment. Our results suggest a Proximal Policy Optimization (PPO) algorithm combined with a Multi-Layer Perceptron (MLP) surrogate environment to successfully learn a policy that continuously changes parameter configurations optimally, achieving the desired target material thickness within an acceptable range.

Keywords: Reinforcement Leaning · Deep Learning · Real World Manufacturing. · Intelligent Decision Support

1 Introduction

In recent years reinforcement learning (RL) has achieved groundbreaking success in sequential decision-making problems by utilizing function approximation in deep learning [10]. The resulting deep reinforcement learning (DRL) methods have achieved superhuman performance in domains ranging from Atari [19] to Go [23] to chip floorplanning [17]. DRL has also been successful in industrial applications such as robotics [11, 12] and the sanitary area [20]. These works demonstrate the potential of DRL to solve complex control tasks with high-dimensional state and/or action spaces and provide valuable contributions to modern engineering.

In current manufacturing processes, the traditional Proportional Integral Derivative (PID) controller [3] is commonly used in combination with the expertise of human operators to optimize the process. The PID controller's simplicity in implementation and tuning [24] makes it well-suited for most control problems without much mathematical modelling and analysis [2]. However, PID controllers face the challenge of planning ahead to avoid driving either the control effort or the process variable outside their acceptable ranges, which constrains control and does not handle dynamic environments well. Furthermore, in environments with multi-variable control issues and conflicting requirements, PID controllers suffer a multi-objective problem [15]. In this work, we investigate dynamic environments in which PID controllers are not used due to their aforementioned shortcomings. We deal with complex industrial manufacturing processes where the main challenge is finding a (semi-) automatic continuous control policy, primarily due to non-existing simulation environments, non-existing PID controllers, and major barriers to online testing such as high implementation costs. However, expert empirical data is usually collected in these industrial manufacturing processes. Hence, we aim to develop a general control framework that enables us to train a DRL agent to deal with dynamic, uncertain, and (soft-) constrained environments by utilizing expert empirical data.

Inspired by [16], our control framework can be summarized as follows. First, to overcome the difficulty of building a precise simulator, we train a surrogate model using authentic real-world data to forecast the thickness of the manufactured material given the input parameter configuration. Second, we use this trained surrogate model as an environment for training a DRL agent, with the ultimate goal of learning a policy that continuously "finds" the optimal key control parameter configuration to achieve the desired outlet thickness of the manufactured material.

While numerous parameters may be pertinent to an industrial manufacturing process, our focus is on managing the thermal profile at distinct stages of the procedure, which is crucial for determining certain properties of the material and for eliminating significant defects. Managing the thermal profile demands continuous control due to fluctuations caused by unobserved or not controllable exogenous factors, which can cause the material thickness to vary even when all controllable parameters remain constant. The need for continuous control arises from the constant changes in the material target thickness, requiring continuous parameter adjustments.

In order to find the optimal setting for our framework, we test various established DRL algorithms, namely PPO [22], DDPG [13] and DQN [18], in diverse surrogates, including Random Forest (RF) and Multilayer Perceptron (MLP) environments. We compare their respective performances and determine the most effective combination. Our experimental results show that the combination of PPO with the MLP surrogate is the most effective one. It can rapidly and accurately identify the optimal material thickness-inducing parameters to meet the desired thickness needs and reduce the loss incurred during the process, even under uncertain environmental conditions. Our approach is capable of in-

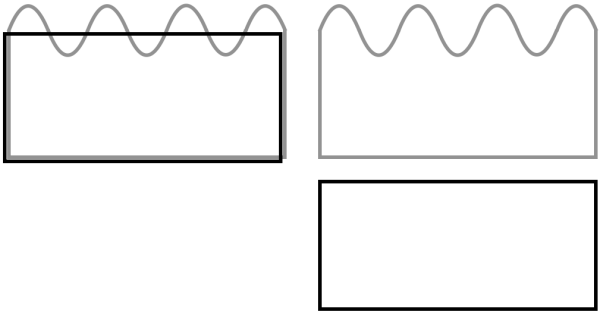

Fig. 1: Material Profile: The black rectangle denotes the optimal product, while the grey rectangle represents the actual fabrication. Any excess material above the black line signifies potential material savings, whereas falling below the black threshold indicates an unsellable product.

corporating these uncertainties, encoded in the state, into the decision-making process. Furthermore, our experiments demonstrate that, interestingly, although both RF and MLP surrogate models achieve equally good accuracy in predicting the material thickness, the MLP surrogate model proves to be a superior simulation environment for training the DRL agent to develop an effective policy for addressing the underlying control problem. In addition, to validate if our trained surrogate model can provide a good approximation to an existing simulation environment for training a DRL agent, we utilize the widely recognized Mountain Car simulation environment [4]. We show that the PPO agent trained in the MLP surrogate environment achieves similar final performance to the PPO agent trained in the Mountain Car environment, demonstrating the ability of our MLP surrogate model to serve as a reliable approximation of a simulation environment.

In this work, we successfully use a DRL agent trained solely on empirical data to control the material thickness in unknown environmental conditions. Our control framework offers a practical solution to comparable industrial control problems where simulation environments are unavailable and traditional PID controllers are not viable options.

2 Background

2.1 Material Thickness Control

The parameters determining the material thickness of certain industrial processes are often still adjusted by human experts. However, unconsidered factors can cause material thickness to vary, even when all controllable parameters are held constant. Moreover, in many industrial processes, a time delay between the change of a process parameter and the resulting material thickness adaptation is common and impedes the control process even more. This delay can be caused

by a variety of factors, such as the time required for a machine to respond to new changes. As a result, accurate prediction of the material thickness based on process parameters can be challenging. The unknown process-specific delay can be learnt using an internal dynamics model or accounted for by shifting the match of process parameter changes and the resulting material thickness adaptation by a certain process-specific factor. In this work, we focus on the latter approach, measuring the time between parameter changes and target thickness convergence. We then adjust our empirical training data by the observed lag time. By incorporating this lag into our training data for our predictive surrogate models, we can more accurately predict product quality and optimize manufacturing processes for improved efficiency and quality control.

In accordance with the requirements of the customer, various material thicknesses are manufactured, while the surface exhibits inherent roughness that we aim to enhance. Figure 1 shows the material profile, where the black rectangle signifies the ideal material thickness, and the grey rectangle represents the actual underlying profile. The objective is to ensure that the material thickness is at least as thick as the optimal black rectangle at every point, while being as close as possible to the optimum.

Various parameters in different zones of the manufacturing process are considered to determine the material thickness and can affect the material properties. In this work, we focus on controlling the heating-related parameters only, which are key parameters that contribute to most of the material thickness variability and can be influenced by external factors.

2.2 Data

For our experiments, we use empirical data collected by the material manufacturer. The data originates from different thickness runs over the course of different years, and is selected based on the quality of the data, e.g., no sensor failures. Measurements are acquired in a short time interval during the entire manufacturing process and cover a wide range of material thicknesses, resulting in a set of multiple distinct target thickness values.

When adjusting the heat parameters, a natural lag occurs which causes the material thickness to change only after a certain time. As a result, observed heat and measured material thickness do not align at the same time. To account for this difference, we train a machine learning model to predict the material thickness given the heating parameter at several time shifts. In this case, the heat measurements are equated with thickness measurements, which are measured $8 * k$, with $k = \{0, 1, ..., 20\}$ minutes later. To evaluate the predictive power, we use 10-fold cross-validation for each time shift between 0 and 160 minutes. We do not test for possible transitions longer than $160min$ due to data preservation. Our results show that a lag of $72min$ results in the lowest mean absolute error (MAE) and is therefore considered consequently. It is important to note that this is an empirical finding based on our data sample and may not generalize to other datasets.

2.3 Reinforcement Learning

Markov Decision Process. Reinforcement learning considers the problem of a goal-directed agent interacting with an uncertain environment and trying to maximize its long-term expected cumulative reward. The underlying decision-making problem can be modelled as a Markov Decision Process (MDP), which can be represented by a tuple $< S, A, P, R, \gamma >$. At each discrete timestep t, the agent is in an environment state $s_t \in S$, which is a momentary representation of the world the agent finds itself in. The agent then selects an action $a_t \in A$ to take to influence the environment. After executing action a_t, the agent moves to the next state s_{t+1} with some probability defined by the state transition function $P(s_{t+1}|s_t, a_t)$, and receives a numeric reward r_t specified by the reward function $R(s_t, a_t)$. $\gamma \in [0, 1]$ is a discount factor specifying how much immediate rewards are preferred to future rewards.

Through the training process, the agent can learn a stochastic policy $\pi(a_t|s_t) : S \times A \to [0, 1]$, which is a per-state action probability distribution that fully defines an agent's behavior. The goal of the agent is to find the optimal policy π^*, which maximizes the expected cumulative discounted reward, denoted as follows:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}, \tag{1}$$

where r_t is the reward received at timestep t. When the agent follows some policy π, a so-called trajectory τ is generated, forming a sequence of states, actions, and rewards. In DRL, the policy π is represented by a neural network, which can be seen as a universal function approximator [9]. With the use of neural networks, reinforcement learning can become more unstable, as high correlations between actions and states may cause network weights to oscillate. To overcome this problem, we can use the experience replay mechanism to store the agent's experiences at each timestep and randomly sample a small batch of experiences to facilitate learning [14].

State-Value Function and Action-Value Function. The state-value function and action-value function are two important concepts in MDPs, which can be used to predict future rewards. The state-value function $V^\pi(s)$ of an MDP can be defined as: $V^\pi(s) = \mathbb{E}_\pi[G_t|s_t = s]$, which estimates the expected total discounted reward the agent will receive starting from state s and following some policy π thereafter. The action-value function $Q^\pi(s, a)$ of an MDP can be defined as $Q^\pi(s, a) = \mathbb{E}_\pi[G_t|s_t = s, a_t = a]$, which estimates the expected total discounted reward the agent will receive after taking action a in state s and following some policy π thereafter. These equations can be recursively defined using the Bellman equation [25]:

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) \Big[R(s, a) + \gamma V^\pi(s') \Big]. \tag{2}$$

$$Q^\pi(s, a) = \sum_{s'} P(s'|s, a) \Big[R(s, a) + \gamma \sum_{a'} \pi(a'|s') Q^\pi(s', a') \Big]. \tag{3}$$

3 Related Work

While DRL has shown impressive performance in a large variety of complex sequential decision-making problems, in the literature, there have only been a few attempts at applying DRL to real-world manufacturing to control material thickness.

Process control has been addressed in several works. [6, 8, 26, 29] use a DQN, [28] use PPO and [27, 7] use a DDPG approach. Of existing work, the approach proposed by [16] and [5] is closest to the control framework presented in this paper. To control the flatness of steel [5] combines PPO with ensemble learning to reduce the risk of falling into local optima. In [16], the aim is to control heating and speed process parameters, called “recipe", in an industrial electric furnace with the goal of reaching a specific desired outlet temperature. Typically, the recipe is discovered by a trial-and-error procedure of human experts, leaving room for improvement in terms of time and cost invested to discover a recipe. To solve the problem of parameter identification, [16] train a DQN [18] agent, whose policy decisions yield such a recipe. A classic environment in the context of RL is provided by a “self-prediction" model. This model is a RF which predicts the outlet temperature of a material given the heat and speed parameters. The data used to train the “self-prediction" model is acquired by simulation.

In contrast to the work of [16], we focus on tackling the problem of controlling material thickness. Furthermore, we compare the performance of three different DRL algorithms instead of only using DQN. DQN aims to learn a good estimate of the optimal action-value function in order to find the greedy deterministic policy. Standard DQN only works for discrete action tasks since maximizing a complex nonlinear action-value function at each update becomes difficult in continuous action tasks. However, using a discrete action space for temperature adjustments is questionable since it can lead to discretization errors or sparse rewards. By the introduction of an unnecessary discretization hyperparameter, we are running the risk of overshooting if the step size is too large. At the same time, a step size too small can result in sparse rewards, making it more difficult for the agent to learn. To overcome this problem, we use a continuous action space. We choose DDPG [13] and PPO [22] due to their competitive performance in continuous control tasks and compare them against the performance of DQN in a discrete action setting. In addition, [16] rely on simulated data to train their self-prediction model, whereas we utilize an authentic real-world dataset to acquire knowledge of the intrinsic dynamics of a manufacturing process. Their self-prediction model is based on a RF approach, but we demonstrate that its capacity for generalization to unobserved instances can be inadequate compared to an MLP approach. This inadequacy can hinder the training of the RL agent and lead to unnecessary complex policies. Our experimental results demonstrate that, interestingly, although both RF and MLP surrogate models achieve equally good accuracy in predicting the material thickness, the MLP surrogate model acts as a better simulation environment for training the DRL agent to learn a more effective policy for solving our problem.

4 Methodology

In this section, we present our control framework that utilizes DRL to automatically determine the main control parameters to achieve the desired outlet thickness of the material. We start by describing the self-prediction model and refer to it as a surrogate model. To train the surrogate model, we use real-world data to predict the thickness value of the material. We then discuss how this learned surrogate model is used as a simulation environment for training the DRL agent and how the material thickness control problem is formulated as an RL problem in our framework.

4.1 Surrogate Model

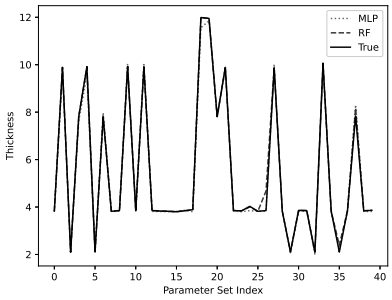

Fig. 2: Predicted thicknesses from either the RF or MLP surrogate model versus the true thickness values. Test data is randomly drawn and consists of 40 different parameter constellations from the empirical dataset.

An RL agent learns how to solve a sequential decision-making problem through real-time interaction with an uncertain environment. The agent learns what actions to take based on a scalar reward signal provided by the environment. In most RL works, (simulation) environments are pre-given with well-defined reward functions. However, we entirely lack a simulation environment for the real-world material process due to its complexity and variability. Hence, real-world data collected from an industrial process is extremely valuable, providing a good representation of the internal dynamics. In our work, we utilize available real-world process data from a production site to build our own simulation environment by learning a good representation of the data itself.

We explore two surrogate models: a Random Forest (RF) with 128 decision trees and a MLP with one hidden layer containing 128 units. These surrogates are trained end-to-end in a supervised manner on the full dataset, using L2 loss to predict material thickness from input parameters. We assess surrogate model

performance through 10-fold cross-validation (80/20 split) using L1 loss. To ensure comparability, both RF and MLP machting an L1 loss of approximately 0.02. These trained surrogate models serve as the simulation environment for training the subsequent DRL agent. They provide state updates and scalar rewards after agent actions. Figure 2 showcases the surrogate models' predictive accuracy for material thickness based on input parameters.

In our experiments, we find that, even though both RF and MLP surrogate models achieve equally good accuracy in predicting the material thickness, the MLP surrogate model demonstrates superior generalization performance over the RF surrogate model, acting as a better simulation environment for training the DRL agent. Specifically, the RF surrogate model, when presented with previously unseen parameter configurations, has a tendency to predict the most frequently observed material thickness value. This results in a significant degree of volatility in the predicted material thickness, even when input parameters exhibit only minor changes.

4.2 Reward Structure

We now explain how the trained surrogate model is used to determine the numeric reward signal sent to the RL agent. Typically, when designing the reward function for an RL problem, we aim to align the rewards with the long-term goal as best as possible. Without further reward engineering, we follow the approach used in [16]. Specifically, we define the reward function by taking the difference between the previous and the current material thickness compared to the target thickness and normalizing the resulting value by the target thickness:

$$r_t = \frac{|MT_{t-1} - MT^{target}| - |MT_t - MT^{target}|}{MT^{target}}. \tag{4}$$

With MT_{t-1} being the previous and MT_t the currently observed material thickness. If the agent's action choice results in a material thickness within a certain acceptance interval, an additional reward of $+1$ is provided to the agent and the current episode is terminated early. Noticeably, the acceptance interval might differ between different target material thickness runs. Mostly, the acceptance interval is set to be $[MT^{target} - 0.2, MT^{target} - 0.1]$, with the lower bound determined by the customer's minimum tolerance. A target thickness of $2mm$ constitutes an exception, as empirically we never observe thicknesses below $2mm$ and the exact reason is unknown to us. Consequently, the maximum cumulative reward the agent can earn within one episode, assuming an initial random state with a material thickness of $2mm$, is approximately 1.75 for a target thickness of $8mm$. Further experiment results refer to a target thickness of $8mm$.

4.3 State and Action Spaces

The state space consists of the current measured heat of several different zones and the actual material thickness observed, denoted as follows:

$$s_t = (FH_t^1, FH_t^2, ..., FH_t^J, MT_t) \in S, \tag{5}$$

where FH_t^j is the heat the jth zone and MT_t is the material thickness currently observed based on the trained surrogate model. At each timestep t of an episode, the RL agent takes an action a_t in environment state s_t, moves to the next state s_{t+1} and receives a scalar reward r_t. Such a sequence describes a trajectory τ_t by which the agent uses to learn to maximize the expected cumulative reward.

We consider both continuous and discrete action spaces by modelling the temperature of each zone as a continuous action or as a set of discrete actions. For the continuous action setting, the action space is defined as:

$$a_t = \{\alpha_t^1, \alpha_t^2, ..., \alpha_t^J\} \in A, \tag{6}$$

where α_t^j implies the increase or decrease in temperature of the jth zone. Hence, the agent is able to change each zone temperature at every timestep t. The natural limitation in the actions is the maximum or minimum observed temperature of the respective zone with an action range of $[-10, 10]$.

In the discrete action setting, the state representation stays the same. However, the action space doubles and we insert an action-step size ζ, which defines a hyperparameter in our setting. The action space in the discrete setting is then defined as:

$$a_t = \{\alpha_t^1, \alpha_t^2, ..., \alpha_t^J, \alpha_t^{J+1}, ..., \alpha_t^{J+J}\} \in A, \tag{7}$$

where $\alpha_t^1, ..., \alpha_t^J$ implies an increase in temperature of the jth zone by $\zeta = 10°\text{C}$ and $\alpha_t^{J+1}, ..., \alpha_t^{J+J}$ implies a decrease in temperature of the jth zone by $\zeta = -10°\text{C}$.

4.4 Agent Description

This work compares the performance of three different DRL algorithms in various surrogate model environments. We now give a brief summary of the DRL methods used.

The first is PPO [22], a policy gradient method that aims to optimize a stochastic policy in a stable and sample-efficient manner similar to Trust Region Policy Optimization (TRPO) [21]. Different to TRPO, PPO uses a clipped surrogate objective function that constrains the update to a small region around the current policy, preventing large policy updates while ensuring policy sustainability. Secondly, we use DDPG [13], an actor-critic algorithm that learns a deterministic policy, meaning it directly outputs the optimal action for a given state. It combines deep neural networks with the traditional actor-critic algorithm to handle high-dimensional continuous action spaces. And thirdly DQN [18], a Q-learning algorithm that uses a deep neural network to estimate the action-value function. It employs an experience replay buffer and a separate target network to stabilize the learning process and prevent overfitting.

5 Experiments

We train the DRL agent by interacting with the surrogate model. All experiments use the same hyperparameters, run for 1000 episodes with 10 random seeds and limit episodes to 2000 timesteps. If the agent reaches the target thickness within an acceptable range, it receives +1 reward, ending the episode early. Initial states are randomly sampled from empirical data, excluding those already within the acceptable range. Action limits are based on observed data. We use $\gamma = 0.999$ for all experiments. Surrogate models include Random Forest (RF) and Multi-Layer Perceptron (MLP), while DRL agents encompass DQN, DDPG, and PPO. We evaluate their performance combinations for solving the continuous control problem. Experimental results are reported for a target thickness of $8mm$.

As we do not have a simulation environment or access to online testing for our control problem, it is intriguing to determine whether the surrogate model can accurately depict the dynamics of the actual underlying environment. We validate our approach by comparing the performance of a PPO agent either trained in the Mountain Car Gym environment (continuous) [4] or trained in the MLP surrogate environment. The training of the MLP surrogate is based on expert trajectories generated from an optimal policy for the Mountain Car Gym environment. In this environment, the agent receives a negative reward of $-0.1 * action^2$ for each step in which the car fails to reach the final position on the hill and a positive reward of +100 upon reaching the final position.

PPO. We use 8 parallel environments and approximate policy and value functions using neural networks with a 2-layer, 64-unit fully-connected MLP architecture. The policy network outputs adjustments for each of the six zones as mean and standard deviation from a normal distribution. We train these networks using stochastic gradient ascent and descent, both using Adam optimizer at a learning rate of 0.004. Advantages are computed with Generalized Advantage Estimation (GAE) as suggested [1] and minibatch normalization with a size of 68 and a lambda of 0.95 during training

DDPG. We maintain two networks: a policy network mapping states to actions and a critic network assessing expected cumulative rewards. Both networks have MLP architectures with 2 hidden layers of 64 units each and ReLU non-linearities. We train the actor and critic networks using Adam optimizer with a learning rate of 0.004. We employ soft updates for the target networks and an exploration strategy based on the Ornstein-Uhlenbeck process.

DQN. We use a Q-network and a target network as twin initialization. Both networks consist of 2 hidden layers with 64 units and ReLU non-linearities. The parameters of the Q-network are updated via Adam optimizer with a learning rate of 0.004. The target network parameters are soft updates of the Q-network parameters. Exploration is performed during training using ϵ greedy. The output of the network is of size $number\ of\ zones\ * 2$ and the index of the maximum value indicates which zone temperature is to be increased or decreased.
The left pane of figure 3 shows the mean cumulative reward attained by the agent during training for a fixed target thickness, using various surrogate models and DRL algorithm combinations in our framework. The maximum cumulative

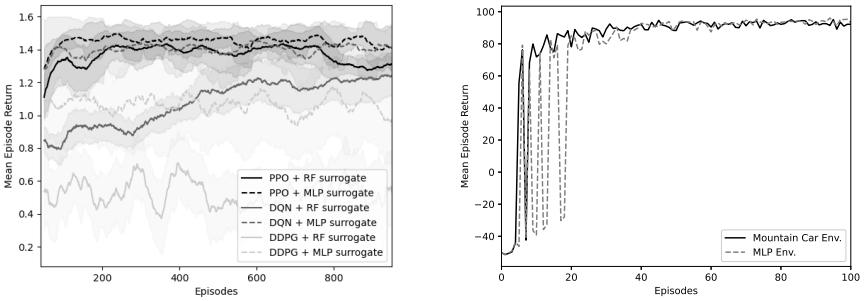

Fig. 3: Left: Mean episode return achieved during training when using various surrogate model and DRL algorithm combinations in our framework. The mean across 10 random seeds is plotted and the ± standard deviation is shown shaded. Right: Mean episode return achieved during training for a PPO agent either trained in the Mountain Car simulation environment or trained in our MLP surrogate environment.

reward achievable ranges between approximately $[1.1, 1.75]$, depending on the difference between the initial material thickness and the target thickness. We can see that the combination of PPO and the MLP surrogate model performs the best among the tested algorithms and surrogate models. It quickly and successfully converges to the maximum attainable reward and learns policies to accurately identify the optimal parameter configuration for the target material thickness within an acceptance interval.

Moreover, when trained in the MLP surrogate environment, all three DRL algorithms demonstrate superior performance compared to their performance in the RF surrogate environment. Compared to PPO, DQN and DDPG are more sensitive to the choice of the surrogate model. Specifically, when trained in the RF surrogate environment, both DQN and DDPG exhibit significantly worse performance in terms of both absolute performance and learning speed, compared to when they are trained in the MLP surrogate environment. The RF surrogate model tends to revert to the most commonly observed target thickness for unseen parameter configurations, thereby introducing noise in the reward signal. Additionally, the DQN algorithm's discrete and fixed step size of 10°C seems to cause the predicted target thickness of the RF surrogate model to vary significantly, further increasing the noise in the reward signal and prolonging the training process. Whereas the discrete step size does not appear to pose any issues for the combination of DQN and MLP surrogate. In our setting, DPPG fails to converge to a good policy completely in the RF surrogate environment and exhibits highly volatile training performance in the MLP surrogate environment.

Surrogate validation. In this work, we consider industrial manufacturing processes where online evaluation is unfeasible due to significant financial costs and the absence of compatible simulations. Consequently, the question arises as

to whether our trained surrogate model can provide a good approximation to a simulation environment for training the DRL agent to learn (sub-) optimal performance. To assess this, we utilize the well-known continuous Mountain Car simulation environment from Gym. The state space of the Mountain Car consists of the position of the car on the x-axis and its velocity, with the agent's action limited to a continuous scale within the range of $[-1, 1]$, representing the directional force applied on the car.

To demonstrate the ability of an RL agent to learn an effective policy through our surrogate model, we first train a PPO agent end-to-end in the Mountain Car environment, using a neural network with 2 hidden layers of 64 units. Subsequently, we use the resulting policy to produce expert trajectories in the Mountain Car environment for 1000 episodes. These trajectories of states s, actions a and next states s' are preserved as training data for the surrogate model. We use the MLP surrogate model, with the states s and actions a as input parameters to predict the next state s'. We employ the same MLP architecture as mentioned above and train it for 5000 episodes with a batch size of 32, resulting in an L2 loss of < 0.01. Upon successful training of the surrogate model, we train another PPO agent using the surrogate predictions (to provide reward signal and state updates) instead of using the Gym environment.

Our argument posits that a few predictions emulate the true dynamics with a high degree of accuracy, and thus, each 50th state update is undertaken by the dynamics of the true environment to stabilize the training procedure. Subsequently, we compare the policy trained with the MLP surrogate model to the policy trained exclusively in the Mountain Car environment. The right pane of figure 3 shows the training curves of both PPO agents in either the true Mountain Car environment or in the MLP surrogate environment. We can observe that the PPO agent trained in the MLP surrogate environment requires more training iterations to converge compared to the PPO agent trained in the Mountain Car environment, but ultimately achieves similar performance levels. This demonstrates that our MLP surrogate model serves as a reliable approximation of the Mountain Car simulation environment.

6 Conclusion and Future Work

In this paper, we proposed a new framework in which we successfully train a DRL agent in a surrogate environment based on real-world data. Our aim is to continuously control the optimal input parameters to achieve a desired prespecified material thickness in a manufacturing process while reducing the loss incurred during the process. To achieve this, we first established an RL environment by training a surrogate model (i.e., an MLP model) on real-world data to predict the material thickness given the input parameters. We then trained a PPO agent by interacting with the established RL environment, to automatically find the main control parameters that lead to the desired target material thickness. To the best of our knowledge, this is the first time a DRL approach has been successfully used to control material thickness in a manufacturing process

using real-world data. We validate our approach by showing that a DRL agent trained in our MLP surrogate environment can achieve similar final performance to the one trained in the Mountain Car simulation environment.

Our framework is general and can be applied to similar control problems where no simulations and online testing are available but access to empirical data is given. Our results identified the optimal pairing of a DRL algorithm with a surrogate environment to be the PPO algorithm when coupled with the MLP surrogate. This is a general recommendation, although a solution tailored to the problem should achieve equal or better results. We argue that rising environmental complexity should be encountered with a fine-tuned surrogate model to minimize the reality gap. In future work, a human-in-the-loop is considered. While other methods are not applicable such as real-world testing due to enormous costs.

References

1. Andrychowicz, M., Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., et al.: What matters for on-policy deep actor-critic methods? a large-scale study. In: International conference on learning representations (2021)
2. Araki, M.: Pid control. Control Systems, Robotics and Automation: System Analysis and Control: Classical Approaches II pp. 58–79 (2009)
3. Bennett, S.: Development of the pid controller. IEEE Control Systems Magazine **13**(6), 58–62 (1993)
4. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym (2016)
5. Deng, J., Sierla, S., Sun, J., Vyatkin, V.: Reinforcement learning for industrial process control: A case study in flatness control in steel industry. Computers in Industry **143**, 103748 (2022)
6. Dornheim, J., Link, N., Gumbsch, P.: Model-free adaptive optimal control of episodic fixed-horizon manufacturing processes using reinforcement learning. International Journal of Control, Automation and Systems **18**, 1593–1604 (2020)
7. Gamal, O., Mohamed, M.I.P., Patel, C.G., Roth, H.: Data-driven model-free intelligent roll gap control of bar and wire hot rolling process using reinforcement learning. International Journal of Mechanical Engineering and Robotics Research **10**(7), 349–356 (2021)
8. Guo, F., Zhou, X., Liu, J., Zhang, Y., Li, D., Zhou, H.: A reinforcement learning decision model for online process parameters optimization from offline data in injection molding. Applied Soft Computing **85**, 105828 (2019)
9. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural networks **2**(5), 359–366 (1989)
10. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)
11. Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep visuomotor policies. The Journal of Machine Learning Research **17**(1), 1334–1373 (2016)
12. Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., Quillen, D.: Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. The International journal of robotics research **37**(4-5), 421–436 (2018)

13. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
14. Lin, L.J.: Reinforcement learning for robots using neural networks. Carnegie Mellon University (1992)
15. Martínez, M.A., Sanchis, J., Blasco, X.: Multiobjective controller design handling human preferences. Engineering applications of artificial intelligence **19**(8), 927–938 (2006)
16. Mazgualdi, C.E., Masrour, T., Hassani, I.E., Khdoudi, A.: A deep reinforcement learning (DRL) decision model for heating process parameters identification in automotive glass manufacturing. In: Artificial Intelligence and Industrial Applications. pp. 77–87. Springer International Publishing (2021)
17. Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J.W., Songhori, E., Wang, S., Lee, Y.J., Johnson, E., Pathak, O., Nazi, A., et al.: A graph placement methodology for fast chip design. Nature **594**(7862), 207–212 (2021)
18. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
19. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)
20. Ruelens, F., Claessens, B.J., Quaiyum, S., De Schutter, B., Babuška, R., Belmans, R.: Reinforcement learning applied to an electric water heater: From theory to practice. IEEE Transactions on Smart Grid **9**(4), 3792–3800 (2016)
21. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In: International conference on machine learning. pp. 1889–1897. PMLR (2015)
22. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms (Jul 2017)
23. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. nature **529**(7587), 484–489 (2016)
24. Stewart, G., Samad, T.: Cross-application perspectives: Application and market requirements. The impact of control technology pp. 95–100 (2011)
25. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
26. Wu, T., Zhao, H., Gao, B., Meng, F.: Energy-saving for a velocity control system of a pipe isolation tool based on a reinforcement learning method. International Journal of Precision Engineering and Manufacturing-Green Technology pp. 1–16 (2021)
27. Yu, J., Guo, P.: Run-to-run control of chemical mechanical polishing process based on deep reinforcement learning. IEEE Transactions on Semiconductor Manufacturing **33**(3), 454–465 (2020)
28. Zinn, J., Vogel-Heuser, B., Gruber, M.: Fault-tolerant control of programmable logic controller-based production systems with deep reinforcement learning. Journal of Mechanical Design **143**(7), 072004 (2021)
29. Zirngibl, C., Dworschak, F., Schleich, B., Wartzack, S.: Application of reinforcement learning for the optimization of clinch joint characteristics. Production Engineering **16**(2-3), 315–325 (2022)