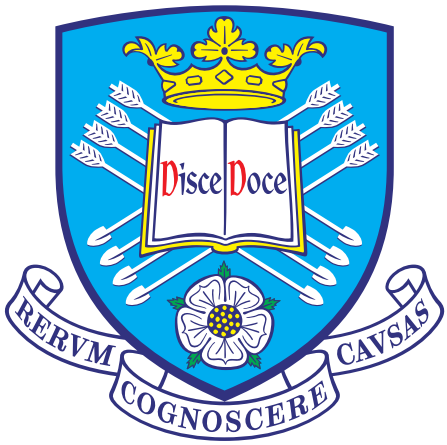
# Sample-Efficient Deep Reinforcement Learning for On-The-Fly Thermal Process Control in Laser Powder Bed Fusion

**Ricardo Dominguez-Olmedo**

May 2019




Supervisor: Dr George Panoutsos
Second reader: Dr Lyudmila Mihaylova

Department of Automatic Control & Systems Engineering
University of Sheffield

A dissertation submitted in partial fulfilment of the requirements for the degree
*BEng Mechatronics and Robotic Engineering*

# EXECUTIVE SUMMARY


## INTRODUCTION AND BACKGROUND

Deep reinforcement learning (DRL) is a framework where control can be learned through direct interaction with a system, without requiring any previous knowledge about the system dynamics. Recently developed model-based DRL algorithms may be sufficiently sample-efficient for learning control of physical systems.

Powder Bed Fusion (PBF) is a manufacturing process where it is difficult to reliably produce high-quality outputs. Despite real-time control being identified as a key technology necessary for enhancing product quality, no previous studies have demonstrated closed-loop thermal control of PBF, since the dynamics of PBF are unknown, high-dimensional and very non-linear.

## AIMS AND OBJECTIVES

The project aimed to use state-of-the-art model-based DRL for thermal process control of a real-world PBF process. The main objectives were to asses, in simulation and on a real-world PBF manufacturing system, the sample-efficiency and control performance achievable by DRL.

## ACHIEVEMENTS

The model-based DRL algorithm provided excellent control performance for the real-world PBF manufacturing system and exhibited exceptional sample-efficiency. The algorithm learned control on-the-fly without any previous knowledge about the system dynamics and was shown to work for different materials object dimensions without requiring any tuning.

## CONCLUSIONS AND RECOMMENDATIONS

Model-based DRL can learn excellent control of physical systems with remarkable sample-efficiency. However, the use of model-based DRL for control of the PBF process did not improve, nor lessen, product quality. Further research must be conducted to determine in what manner PBF thermal process signatures must be controlled in order to enhance product quality.

# ABSTRACT

Deep reinforcement learning (DRL) is a framework where control can be learned through direct interaction with a system. Recent research on model-based DRL has resulted in the development of algorithms which may be sufficiently sample-efficient to control physical systems. This study tests this hypothesis by using model-based DRL to control a complex Powder Bed Fusion (PBF) manufacturing process. The dynamics of the PBF process are unknown, high-dimensional and very non-linear. It is highly desirable to achieve thermal control of the PBF processes, as this could improve the quality of the manufactured products.

The model-based DRL algorithm provided excellent control performance for the real-world PBF manufacturing system and exhibited exceptional sample-efficiency. The algorithm learned control on-the-fly without any previous knowledge about the system dynamics and was shown to work for different materials object dimensions without requiring any tuning. However, the use of model-based DRL for control of the PBF process did not improve, nor lessen, product quality.

This study is the first to demonstrate that DRL can be used for process control of physical systems, indicating that model-based DRL algorithms are sufficiently sample-efficient for physical application domains. Furthermore, this study is the first to demonstrate closed-loop thermal process control of a PBF process, a landmark which likely will facilitate future research efforts aiming to characterise the formation of manufacturing defects in PBF.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **AM** | Additive manufacturing |
| **DoE** | Design of experiment |
| **DRL** | Deep reinforcement learning |
| **LQG** | Linear quadratic Gaussian |
| **MPC** | Model predictive control |
| **PBF** | Powder bed fusion |
| **PETS** | Probability Ensembles with Trajectory Sampling |
| **PI** | Proportional integral |
| **PS** | Policy search |
| **RL** | Reinforcement learning |
| **TS** | Trajectory sampling |
| **VF** | Value function |

# Chapter 1

# Introduction

## 1.1 Background and Motivation

### Deep Reinforcement Learning

Reinforcement learning (RL) is a framework for experience-driven control learning: the controller interacts with a system and, by receiving *rewards* for its actions, autonomously adjusts its behaviour with the aim of maximising future rewards. The combination of deep learning with RL has been shown to enable control to be learned "end-to-end" [1]: control laws which directly map raw high dimensional observations to control inputs can be autonomously learned. This is in contrast with traditional system identification and controller design methods, which require substantial application-specific knowledge or human expertise.

However, deep reinforcement learning (DRL) algorithms require millions of interactions with a system before adequate control can be learned, rendering DRL highly impractical in most physical domains due to cost and time constraints. To address this issue, model-based DRL has become a very active area of research, leading to the recent introduction of substantially more sample-efficient DRL algorithms. However, to date, these algorithms have only been demonstrated in robotic toy-problems such as pick-and-place tasks [1, 2].

Sample-efficient DRL could have a substantial impact on industrial control applications, since DRL algorithms could be used to autonomously learn control of non-linear and high-dimensional systems without requiring the often expensive and time-consuming processes of system identification and controller design.

### Additive Manufacturing and Powder Bed Fusion

Additive manufacturing (AM) encompasses a range of technologies capable of manufacturing 3D objects through a process of stacking layers of material. Compared to other manufacturing techniques, AM offers exceptional geometric freedom while maintaining simple design processes [3]. Thus, bespoke objects with intricate shapes can be designed and produced within low lead times. Despite the advantages of AM, the difficulty of reliably producing high-quality outputs has prevented AM from becoming a widely adopted technology [4].

Powder Bed Fusion (PBF) is a category of AM where a laser or an electron beam is used to fuse a fine powder in order to form the different layers of material comprising the 3D object being manufactured. Despite real-time control being identified as a key technology necessary for enhancing the quality and consistency of products manufactured using PBF [5], no previous studies have demonstrated closed-loop thermal control of PBF.

The lack of closed-loop control can be largely attributed to the very challenging nature of PBF process control: the dynamics of PBF are unknown, high-dimensional and very non-linear. The development of PBF control is further difficulted by the high cost of operating PBF manufacturing systems, since a single build may cost upwards of thousands of pounds.

## 1.2 Aim and Contribution

This project aims to use state-of-the-art model-based DRL for thermal process control of a real-world PBF process. The project objectives were to:

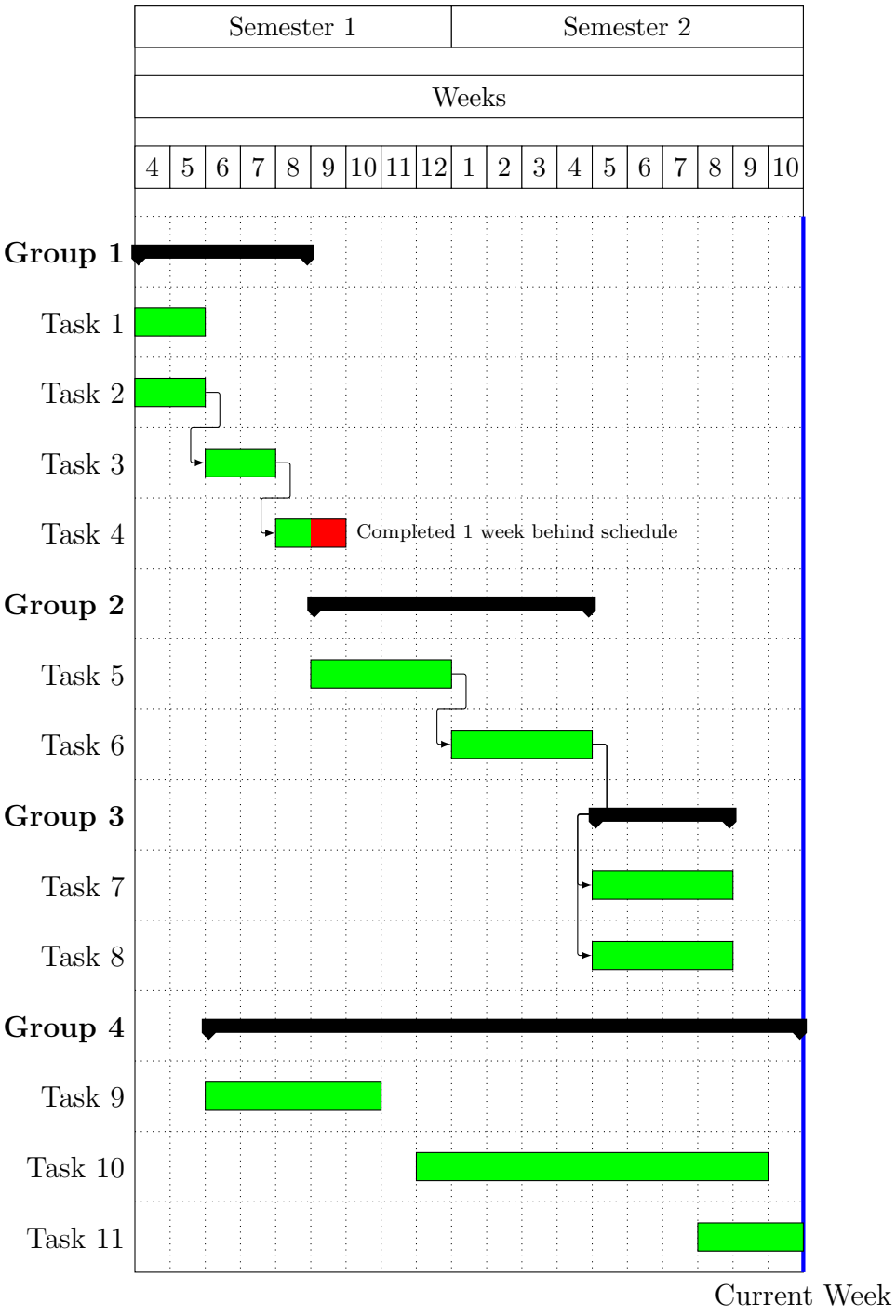
- Review the existing literature in model-based DRL in order to select the most suitable DRL algorithm in terms of sample-efficiency.
- Asses, in simulation, the expected control performance and sample-efficiency of the selected DRL algorithm for the process control problem considered.
- Attempt to control the real-world PBF manufacturing system using the selected DRL algorithm, and asses its performance.

Doing so, this project provides the following novel contributions to the fields of deep reinforcement learning and PBF additive manufacturing:

- This study is the first to demonstrate that DRL can be used for process control of physical systems. This landmark indicates that recently introduced model-based DRL algorithms are sufficiently sample-efficient for physical application domains, and can autonomously learn excellent control of very complex systems.

- This study is the first to demonstrate closed-loop thermal process control of a PBF process. For industrial applications, the approach proposed can learn on-the-fly without requiring expensive and time-consuming experiments to be performed in order to determine the optimal process parameters. For research applications, the ability to precisely control thermal process signatures is expected to be essential to better understand the formation of manufacturing defects in PBF.

## 1.3 Project Management




- Green: Completed
- Red: Completed behind schedule
- Gray: Uncompleted

Tasks may be completed during Christmas and Easter vacations.

**Group 1** Preliminary and simulation work

- **Task 1** Establish the manufacturing system requirements
- **Task 2** Choose DRL method
- **Task 3** Implement DRL method in Python
- **Task 4** Benchmark DRL implementation using CartPole environment

**Group 2** Practical work - main objectives

- **Task 5** Adjust DRL implementation to the real system
- **Task 6** Apply DRL implementation to the real system

**Group 3** Practical work - advanced objectives

- **Task 7** Perform a build with a different material
- **Task 8** Advanced modelling and & simulation (bootstrapping)

**Group 4** Report writing

- **Task 9** Write interim report
- **Task 10** Write dissertation
- **Task 11** Prepare oral presentation

## 1.4 Outline

The remaining of this dissertation is structured as follows: the background into the topics of deep reinforcement learning and powder bed fusion is provided in Chapter 2 and Chapter 3, the process control case study is introduced in Chapter 4, and the performance of PETS –the DRL algorithm used– is evaluated in Chapter 5 and Chapter 6, first in simulation and then on a physical manufacturing system.

**Chapter 2, Model-Based Deep Reinforcement Learning**: The existing literature on model-based DRL is reviewed, with an emphasis on sample-efficiency. The most suitable algorithm in terms of sample-efficiency, PETS, is presented.

**Chapter 3, Powder Bed Fusion Processes**: PBF additive manufacturing is introduced. Previous studies on PBF process monitoring and control are reviewed.

**Chapter 4, Introduction to the Case Study** The PBF process control problem is fully defined as a reinforcement learning problem.

**Chapter 5, Case Study: Modelling and Simulation** The PBF process is modelled in order to develop a simulation environment. The control performance and sample-efficiency of PETS are evaluated in simulation.

**Chapter 6, Case Study: Process Control**: The control performance of PETS is evaluated on the real manufacturing system and compared to the current industry standard where no control is used.

# Chapter 2

# Model-Based Deep Reinforcement Learning

This Chapter introduces the field of reinforcement learning, the core approaches to sample-efficient DRL, and the DRL algorithm used for the control case study.

Firstly, the reinforcement learning framework and the core methods for solving RL problems are introduced in §2.1. The description of these RL methods does not intend to be exhaustive, but rather a brief introduction in anticipation of the comprehensive literature review on model-based DRL provided in §2.2. The literature review provides justification, on the basis of sample-efficiency, for the chosen DRL algorithm, PETS, which is described in detail in §2.3.

## 2.1 The Reinforcement Learning Framework

The RL framework is formalised as a discrete-time Markov decision process [6], comprised of a set of *system states* $x \in X$, a set of *control inputs* $u \in U$, a *dynamics function* $f(x_{t+1} \mid x_t, u_t)$ and a *reward function* $\mathcal{R} : (x_{t+1}, u_t) \mapsto r_{t+1} \in \mathbb{R}$. Additionally, the *controller function* $\pi(u_t \mid x_t)$ maps system states onto a distribution of control inputs. The fundamental principle of RL is that the dynamics function is completely unknown to the controller, and thus a suitable controller function must be learnt purely from iteration with the system being controlled.

The control task is executed in discrete *trials* each lasting a finite number of time-steps $H$. At each discrete time-step $t$, the controller observes the system

state $x_t$ and samples a control input $u_t \sim \pi(x_t)$ using the controller function. As a result, the system transitions to a new state $x_{t+1} \sim f(x_t, u_t)$ according to the dynamics function. A *reward* $r_{t+1} = \mathcal{R}(x_{t+1}, u_t)$ is assigned to the observed state transition. The *return* of a trial is defined as the cumulative reward achieved over the entire time horizon $H$ of the control task:

$$R = \sum_{t=0}^{H-1} r_{t+1} \tag{2.1}$$

The goal of RL algorithms is to learn an optimal controller function $\pi^*$ which maximises the expected return:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\ \mathbb{E}[R \mid \pi] \tag{2.2}$$

There are two main approaches for solving RL problems: value function methods and policy search methods. Both methods can incorporate the use of models.

### Value Function Methods

Value function (VF) methods aim to estimate the *quality function* $Q$, that is, the expected return obtained by, starting at state $x$, applying the control input $u$ and then using the controller function $\pi$ to sample further control inputs:

$$Q^\pi(x, u) = \mathbb{E}[R \mid x, u, \pi] \tag{2.3}$$

The estimate of the $Q$ function is iteratively improved by interacting with the system, using the information gained from the observed state transitions $(x_t, u_t, x_{t+1}, r_{t+1})$. Once a sufficiently accurate $Q$ function is learned, the controller can act optimally by choosing, at each time step, the optimal control input $u_t^*$ which maximises the expected return from state $x_t$ :

$$u_t^* = \underset{u_t}{\operatorname{argmax}}\ Q^\pi(x_t, u_t) \tag{2.4}$$

In deep reinforcement learning, deep neural networks (DNN) are used as function approximators to estimate the $Q$ function [7]. DNNs scale well to high-dimensional

spaces, unlike the look-up tables used in traditional VF methods [8], and as a result the $Q$ function can be directly learned from raw high-dimensional observations. Traditional VF methods, on the other hand, require the use of hand-engineered feature extraction to reduce the dimensionality of the state space [9], in order to make the algorithms computationally tractable.

### Policy Search Methods

Policy search (PS) methods aim to learn suitable control by directly optimising a parametrised controller function $\pi_\theta$, where the optimisation objective is to maximize the expected return $\mathbb{E}[R \mid \theta]$ of the parametrised controller function. The parameters $\theta$ are iteratively updated using either gradient-free optimisation –most often genetic algorithms [10]– or gradient-based optimisation –such as gradient descent [11]– for differentiable controller functions.

In deep reinforcement learning, deep neural networks are used as the controller function and are most commonly optimised using the computationally-efficient backpropagation algorithm [12]. Due to the exceptional expressive power of DNNs, PS methods in DRL can learn remarkably complex controller functions, which is necessary to solve very intricate decision-making or control problems [9].

### Model-based Methods

Model-based RL methods learn an approximation $\widetilde{f}$ of the dynamics function $f$ using the dataset of state transitions $\mathcal{D} = \{(x_n, u_n), x_{n+1}\}_{n=1}^{N}$ collected by interacting with the system. The learned system dynamics can be integrated within RL methods in a variety of ways, as discussed in §2.2.

## 2.2 Sample-Efficiency and the Use of Models

The literature on DRL employs a suite of standard simulated benchmarks to allow the fair and direct comparison of different DRL algorithms. These benchmarks are of two types: classic Atari computer games [13] and robotic simulators of legged locomotion or pick-and-place tasks [14]. The DRL community uses two metrics to assess the performance of a given algorithm: the *asymptotic performance* and the

*sample-efficiency*. The asymptotic performance is the benchmark score obtained by the algorithm upon convergence and indicates the quality of the control solution found. On the other hand, the sample-efficiency is the number of interactions or samples required for the algorithm to converge to an optimal solution and indicates the learning speed of the algorithm.

While early studies on model-free DRL focused on maximising asymptotic performance irrespective of sample-efficiency [15–19], in recent times sample-efficiency has become a prominent research focus, with the aim of developing algorithms which can be used in physical application domains, where samples are often expensive and time-consuming to obtain. Although more sample-efficient model-free methods have been developed [20–22], model-based DRL has emerged as the dominant approach for sample-efficient DRL.

Research on model-based DRL revolves around three general techniques: value function methods, policy search methods, and methods employing deep models in combination with optimal control.

Model-based DRL uses purely data-driven models since a fundamental principle of RL is to assume no initial knowledge about the system dynamics. Nonetheless, some studies [23–25] have attempted to embed prior knowledge about the system dynamics –such as physical models– within model-based DRL. These methods are not considered in the provided literature review on model-based DRL, since, as opposed to the fundamental principles of RL, they require greater domain-specific knowledge and have limited applicability.

### 2.2.1 Model-based Value Function Methods

Model-based VF methods fit a model of the system dynamics using the data collected by interacting with the system, which is then used to simulate or *imagine* future state trajectories or *rollouts*. The existing literature on model-based VF algorithms has explored two approaches for integrating the use of imagination: directly as additional training data to fit the $Q$ function, or indirectly to improve value estimation –to better estimate the return expected from a given state–.

The former approach was introduced by Gu et al. [26], where a replay buffer with both real and imagined state transitions is used to iteratively fit $Q$. Sample-

efficiency was found to be substantially improved if a sufficiently accurate model of the system dynamics could be learned. However, performance degraded dramatically for imperfect models. Thus, local linear models were found to be superior to deep neural networks, since sufficient local accuracy could be achieved within fewer samples. To prevent performance degradation due to model inaccuracies, Kalweit et al. [27] built upon the work by Gu et al. by using uncertainty-aware Bayesian deep neural network models, and only incorporating imagined state transitions to the replay buffer when model uncertainty was low.

Imagination has also been used to improve value estimation: Feinberg et al. [28] used short imagination rollouts at every time step to better predict the expected return which could be achieved form the current state. This was shown to improve value estimation, resulting in fewer system interactions required to train a sufficiently accurate $Q$ function. Buckman et al. [29] further enhanced the speed at which the $Q$ function can be learned by, building upon the work of Feinberg et al., using an uncertainty-aware model to reject rollouts with high uncertainty. Finally, Stadie et al. [30] used a tangential approach in which imagined rollouts were not directly used for value estimation, but rather to incentivize the exploration of previously unseen states. This was shown to substantially reduce the number of samples required to learn a globally accurate $Q$ function.

Despite the improvements introduced by the studies presented above, value function algorithms share a common limitation which hinders their sample-efficiency: both "good" and "bad" states must be experienced in order to train an accurate $Q$ function [26]. Thus, model-based VF methods are generally only an order of magnitude more sample-efficient than model-free methods. The asymptotic performance of model-based VF methods is, nonetheless, similar to that of state-of-the-art model-free methods. This is because model-based VF methods are very robust to model errors, since the model is not directly used for decision-making, but rather to improve the $Q$ function estimates.

### 2.2.2 Model-based Policy Search Methods

The existing literature on model-based policy search shares a common approach: the data acquired by interacting with the system is used to build a model, which is

then used to directly optimise the controller function in simulation. Model-based policy search algorithms thus iterate between using the model to improve the controller function and using the controller function to interact with the system and acquire more data with which to improve the model.

Literature differs on the choice of model and optimisation algorithm. Ha and Schmidhuber [31] used a very deep recurrent neural network (RNN) –10 convolution layers– model together with the evolutionary optimisation algorithm CMA-ES [32]. While the use of a very deep model resulted in state-of-the-art performance in extremely complex benchmarks, the improvement of sample-efficiency was minimal due to the large amount of data required to train the model.

Therefore, other studies have employed shallower deep models for greater sample-efficiency: Kurutach et al. [33] used a model ensemble of neural networks and trust region policy optimisation (TRPO [16]), and Depeweg et al. [34] used a Bayesian neural network model and stochastic gradient descent. Both studies conclude that the use of uncertainty-aware models was critical to the asymptotic performance of these algorithms, as it prevented the models from overfitting in the early learning stages when the amount of available training data is low.

Finally, Levine and Koltun [35] introduced Guided Policy Search (GPS). Unlike previous methods, the controller is a neural network trained in a supervised learning setting, where the training targets are provided by a Linear Quadratic Gaussian (LQG) controller which uses local linear models learned in real-time. GPS was demonstrated by Levine et al. [1] and Chebotar et al. [36] to be sufficiently sample-efficient to learn control for real-world robotic pick-and-place tasks.

The studies reviewed (except Ha and Schmidhuber) all achieve up to two orders of magnitude improved sample-efficiency compared to model-based VF methods, since the controller function can be directly optimised without having to explore the entire state space. However, model-based policy search algorithms generally have worse asymptotic performance compared to model-free DRL, since model errors often lead to suboptimal controller functions [34], due to the controller functions being directly optimised using the trained model.

### 2.2.3 Optimal Control with Deep Models

Model-based value function and policy search methods learn, on top of a model of the system, either a $Q$ function or a controller function. Methods combining optimal control with deep models, on the other hand, aim to improve sample-efficiency by only learning a model of the system, which is then used to optimise the control inputs on-the-fly without requiring an explicit controller function.

Methods combining optimal control with deep neural network models have been developed in the context of robotic applications: Finn et al. [37] demonstrated the use of an LQG controller to solve a robotic manipulation task, and Wahlstrom et al. [38], Finn and Levine [39] and Lenz et al. [40] demonstrated the use of model predictive control (MPC) in pick-and-place robotic tasks. Despite the success of these methods on real-world tasks, they all achieved poor asymptotic performance in the very complex simulated benchmark environments used in DRL [41].

The recent work by Chua et al. [42] was the first study combining deep models and optimal control to achieve state-of-the-art asymptotic performance in DRL benchmarks. The algorithm presented by Chua et al., named Probability Ensembles with Trajectory Sampling (PETS), also achieves extraordinary sample-efficiency: it requires up to two orders of magnitude fewer samples than model-based policy search to find comparable benchmark solutions.

The success of PETS stems from the use of uncertainty-aware models, motivated by the insights of previous studies in model-based policy search such as Depeweg et al. [34] Kurutach et al. [33]: uncertainty-aware models are critical for preventing overfitting during the initial stages of learning, where the amount of training data is low. Indeed, overfitting was the main cause of the poor performance in benchmarks of previous algorithms combining optimal control with deterministic models, as overfitting in the initial learning stages leads to poor suboptimal solutions and can even cause highly unstable behaviour [42].

Due to the novelty of PETS, the algorithm had not previously been used in a real-world application. Nonetheless, it was decided to use PETS in this project due to its unprecedented sample-efficiency and very strong asymptotic performance. This choice was reassured by the fact that other methods combining optimal control with deep models had been applied to real-world robotic tasks.

## 2.3 The PETS Algorithm

The PETS algorithm has been shown to achieve state-of-the-art performance in standard simulated benchmarks, while being significantly more data-efficient than previous DRL methods. PETS was presented at the 2018 Conference on Neural Information Processing Systems (NIPS), shortly after the start of this project.

PETS (Algorithm 1) is formulated as an epistemic RL problem, where $K$ trials of $H$ time steps each are performed on the real system being controlled. At the end of each trial, the system is reset to its initial system state.

**Algorithm 1** Probability Ensembles with Trajecotry Sampling (PETS)

1: Initialise dataset $\mathbb{D}$ (e.g. data from a single trial using a random control inputs)
2: **for** Trial $k = 1$ to $K$ **do**
3: Train dynamics model $\widetilde{f}$ given $\mathbb{D}$.
4: **for** Time $t = 0$ to task time-horizon $H$ **do**
5: Obtain optimal control inputs $u^*_{t:t+T}$ using CEM and TS
6: Execute first control input $u^*_t$
7: Store outcome: $\mathbb{D} \leftarrow \mathbb{D} \cup \{x_t, u^*_t, x_{t+1}\}$

Before each trial, a model of the system dynamics is trained using the data collected thus far from the physical system. This model is then used to, at each time step, find the optimal sequence of control inputs for the current system state. Only the first control input of this sequence is executed on the real system, as the sequence is reoptimised at every time step. The modelling and control methods used are discussed in detail in §2.3.1 and §2.3.2

The dataset used to train the dynamics model is commonly initialised by performing a number of initial trials with random control inputs. This initialisation method is generally preferred as it assumes no previous knowledge about the system –a known control law or previously collected data– and often offers good coverage of the state and input spaces through random exploration [42]. Nonetheless, if it is unsafe to operate the system using random control inputs, a user-specified control law could also be employed during the initial roll-outs.

### 2.3.1 Modelling of the System

The system dynamics are modelled using an ensemble of $B$-many probabilistic neural networks (NN), each encoding a Gaussian probability distribution. The

inputs to the NNs are the system state $x_t$ and control input $u_t$ at the current time step $t$, while the NN outputs are the mean $\mu$ and diagonal covariance $\sigma$ encoding the next system state $x_{t+1}$:

$$\widetilde{f}(x_{t+1} \mid x_t, u_t) = Pr(x_{t+1} \mid x_t, u_t) = \mathcal{N}(\mu(x_t, u_t), \sigma(x_t, u_t)) \tag{2.5}$$

Each model within the ensemble has a different set of weights $\theta_b$ trained using its own datasets $\mathcal{D}_b$, which is obtained by sampling with replacement the dataset of observations $\mathcal{D} = \{(x_n, u_n), x_{n+1}\}_{n=1}^{N}$ collected thus. The loss function used for training the NNs is the negative log prediction probability.

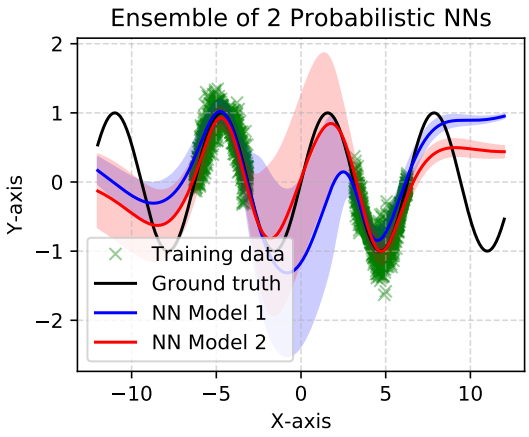



Figure 2.1: Illustration of a model ensemble.

The use of an ensemble of probabilistic models allows PETS to isolate two types of model uncertainty:

- Aleatoric uncertainty stems from the inherent stochasticity of the system dynamics and is directly modelled by the probabilistic NNs.
- Epistemic uncertainty arises from the "lack of sufficient training data to uniquely determine the underlying system dynamics" [42], and is inferred from the extent to which the NNs within the ensemble differ from each other. For instance, in Figure 2.1 the two models of the ensemble differ significantly around the origin, where there is no training data. However, both models are very similar in the regions populated with data points.

Currently, PETS does not directly make use of epistemic uncertainty to improve data-efficiency. In future work, however, epistemic uncertainty could be used

as a principled approach to guide exploration by incentivising the exploration of regions where this uncertainty is high.

### 2.3.2 Planning and control

PETS uses model predictive control (MPC) to, at each time step, find the set of control inputs which maximises the expected reward over the planning horizon $T$.

#### Computing the Expected Reward of an Input Sequence

**Algorithm 2** Cost evaluation of a control input sequence using TS

**Input:** $u_{t:t+T}$: Control input sequence, $x_t$: Current system state, $\widetilde{f}$: Ensemble of $B$-many probabilistic models, $P$: Number of particles for trajectory sampling
**Output:** Expected cost $J = -\sum_{\tau=t}^{t+T} \mathbb{E}_{\widetilde{f}}[\mathcal{R}(x_{\tau+1}, u_\tau)]$

**for** Particle $p = 1$ to $P$ **do**
  Initialise particle to current system state $x_t^p \leftarrow x_t$
  Uniformly sample a bootstrap $b \leftarrow \{1, \ldots, B\}$
  **for** Time-step $\tau = t$ to $t + T$ **do**
    Sample next system state $x_{\tau+1}^p \leftarrow \widetilde{f}_{\theta_b}(x_\tau^p, u_\tau)$
    **if** TS variant is *TS1* **then**
      Uniformly resample a bootstrap $b \leftarrow \{1, \ldots, B\}$
**return** Expected cost $J \leftarrow -\frac{1}{P} \sum_{p=0}^{P} \sum_{\tau=t}^{t+T} \mathcal{R}(x_{\tau+1}^p, u_\tau)$

Algorithm 2 shows the method used to predict the expected reward of a given control input sequence. Trajectory sampling (TS) is used to propagate the uncertainty captured by the learned dynamics model. This is done by initialising $p$-many particles, each with a randomly assigned bootstrap model $b$ within the ensemble. The bootstrap models are then, together with the provided control input sequence, used to propagate each of the particles over the planning horizon $T$. The expected return of each candidate input sequence is approximated as the average return of the trajectories propagated with that particular input sequence.

There are two variants of TS: TS1, where the assigned bootstraps are resampled at every time step, and TS$\infty$, where the assigned bootstraps are kept fixed throughout the entire trajectory. The best performing variant for a particular control problem is often determined empirically (see appendix B.3.1).

The use of uncertainty propagation has been shown to greatly improve performance, as it prevents overfitting in the early learning stages which can lead to

suboptimal control solutions. PETS uses TS because it is easily parallelised, and thus can be much more computationally-efficient than other techniques such as momentum matching or distribution sampling, while resulting in similar performance. A comparison between all uncertainty propagation techniques is provided in appendix B.3.1.

### Optimisation of Control Input Sequences

**Algorithm 3** Control input optimisation using CEM and TS

**Input:** MaxIters, $N$, $N_{\text{elites}}$, $\alpha$, $\epsilon$, $u_{\min}$, $u_{\max}$, $\mu_{u_{t:t+T}} \in \mathbb{R}^{U \cdot T}$, $\sigma^2_{u_{t:t+T}} \in \mathbb{R}^{U \cdot T}$

**for** $i = 1$ to MaxIters **do**
  Sample $N$ control input sequences $u^1_{t:t+T}, \ldots, u^N_{t:t+T} \sim \mathcal{N}(\mu_{u_{t:t+T}}, \sigma^2_{u_{t:t+T}})$
  Evaluate expected cost $J_1, \ldots, J_N$ of $u^1_{t:t+T}, \ldots, u^N_{t:t+T}$ using TS
  Pick the $N_{\text{elites}}$ control input sequences $u'^1_{t:t+T}, \ldots, u'^{N_{\text{elites}}}_{t:t+T}$ with lowest cost
  Compute mean $\mu'_{u_{t:t+T}}$ and variance $\sigma'^2_{u_{t:t+T}}$ of elites $u'^1_{t:t+T}, \ldots, u'^{N_{\text{elites}}}_{t:t+T}$.
  Update mean $\mu_{u_{t:t+T}} \leftarrow \alpha\mu_{u_{t:t+T}} + (1-\alpha)\mu'_{u_{t:t+T}}$
  Update variance $\sigma^2_{u_{t:t+T}} \leftarrow \alpha\sigma^2_{u_{t:t+T}} + (1-\alpha)\sigma'^2_{u_{t:t+T}}$
  Constrain variance $\sigma^2_{u_{t:t+T}} \leftarrow \min(\sigma^2_{u_{t:t+T}}, \frac{(\mu_{u_{t:t+T}} - u_{\min})^2}{4}, \frac{(u_{\max} - \mu_{u_{t:t+T}})^2}{4})$
  **if** Convergence criteria is met: $\max(\sigma^2_{u_{t:t+T}}) \leq \epsilon$ **then**
    **break**
**return** Optimal control input sequence $u^*_{t:t+T} \leftarrow \mu_{u_{t:t+T}}$

The control input sequences are optimised (Algorithm 3) using the cross-entropy method (CEM). This general optimisation framework consists on iteratively sampling a number of candidate solutions from a parametric distribution –often a Gaussian–, evaluating the cost of each candidate solutions, and refitting the distribution parameters to the best performing candidate solutions. The process of refitting the probability distribution is illustrated in Figure 2.2.

CEM can be easily parallelised, offering excellent computational-efficiency, and significantly better performance than simpler optimisation methods such as random shooting, particularly in high dimensional spaces. It is unclear if the use of more advanced optimisation techniques, such as CC-MAES, could lead to significant performance improvements.

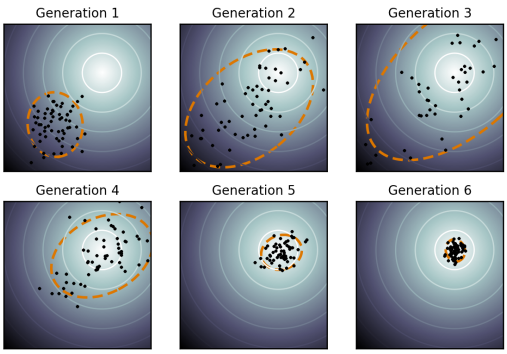



Figure 2.2: CEM Optimisation: evolution of the underlying probability distribution towards a minimum.

### 2.3.3 Software Implementation

The authors of PETS fully open-sourced an implementation of the PETS algorithm in Python using the machine learning library TensorFlow. Extensive changes to the original software package were introduced to conduct the modelling, simulation, and real-world control experiments performed in this project.

A notorious limitation of MPC is the high computational cost of optimising the control inputs at each time step, as is the case for PETS. As a result, the number of experiments which could be conducted in simulation as part of this study was severely limited by the very high run-time of these experiments.

# Chapter 3

# Powder Bed Fusion Processes

This Chapter introduces Powder Bed Fusion with the aim of highlighting the challenging nature of PBF process control: the dynamics of PBF are unknown, high-dimensional and very non-linear.

Firstly, the PBF manufacturing process is introduced in §3.1. Then, the existing literature on PBF and product quality is reviewed in §3.2. Finally, the merits and limitations of previous attempts at PBF process monitoring and control are discussed in §3.3.

## 3.1 Overview of Powder Bed Fusion

Laser powder bed fusion processes use a laser beam to fuse metallic powder in order to form the different layers of material comprising the object being manufactured. The powder particles are fused together by melting the powder with a laser beam, forming an area of molten material which re-solidifies upon cooling. The path which the laser follows when melting the powder –the scan path– is predefined by the PBF machine according to the geometry of the object being manufactured.

The entire PBF process (Figure 3.1) is as follows: first, a recoating blade sweeps the powder from the dispenser platform onto the build platform. Then, the laser fuses the powder particles using a predefined scan path. Afterwards, the build platform is lowered, and another layer of powder is swept on top of the last layer built, which is again fused with the laser beam. This process of stacking layers of material is repeated until the object is fully manufactured.

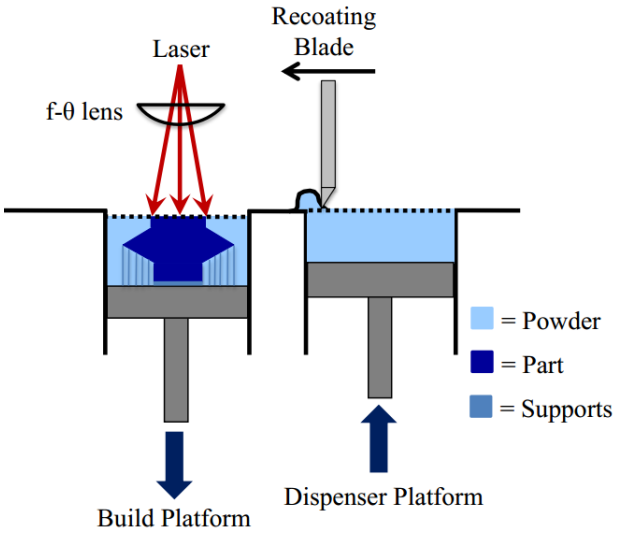



Figure 3.1: Diagram of the PBF process [43].

## 3.2 Build Quality and Process Dynamics

One of the key issues preventing AM from becoming a widely adopted technology is the difficulty of reliably producing high quality products. Thus, much research on PBF has focused on identifying relationships between PBF process parameters and build quality, with the aim of finding optimal process parameters which consistently result in high-quality products.

The existing literature on PBF distinguishes between 4 categories of process parameters [4, 43]: material-related (powder size, powder density, thermal properties, etc), laser-related (power, frequency, pulse, etc.), scan-related (speed, pattern, spacing), and environment-related (oxygen level, pressure, etc.). In total, there are over 50 parameters affecting PBF [44]. Furthermore, the powder melting, fusing, and cooling dynamics are very complex, since they are comprised of coupled non-linear thermal and metallurgic interactions [43].

The very high complexity of PBF processes has severely hindered the development of accurate physical models [45], and as a result the majority of literature has focused on identifying the relationship between processes parameters and product quality empirically. In these studies, different measures are used to determine the build quality, such as the product's mechanical properties [46–48] –strength, hardness, toughness, fatigue resistance, etc.–, physical properties – residual stress [49–51], surface roughness [46, 52, 53] and porosity [54, 55]–, and

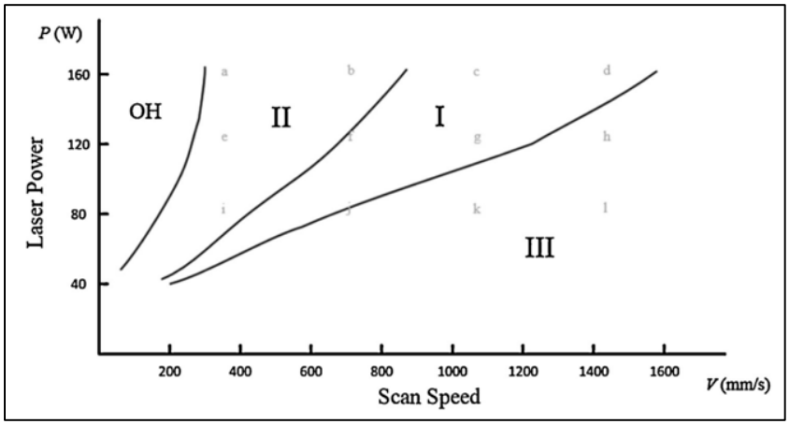



Figure 3.2: Example of energy density envelopes [61]. The optimal envelope is envelope I, while envelope OH leads to severe defect formation.

number of manufacturing defects –porosities [56] and cracks [57]–.

Despite the large number of empirical studies conducted, the extent to which most process parameters and process signatures contribute to the formation of defects is still unknown [44]. However, a large number of studies [46, 47, 49, 52, 54] agree that the dominant parameters affecting build quality are the laser power and scan speed. This insight was used by Aboulkhair et al. [58], Yadroitsev et al. [59] and Helmer et al. [60] to, through a process known as Design-of-Experiments, identify the optimal laser powers and scan speeds which maximise build quality.

The Design-of-Experiments (DoE) methods used by these authors consist on producing several objects –in the order of tens– with different laser power and scan speed configurations, in order to identify *energy density* envelopes where the formation of manufacturing defects is minimised (Figure 3.2). Deviating from the optimal energy density has been shown by Gong et al. [61] to lead to two types of defects: porosities due to excessive energy input [56] and lack-of-fusion defects due to insufficient energy input.

The input energy density $E$ (J/mm$^3$) is a function of the laser power $P$ (W), scan speed $v$ (mm/s), hatch spacing $h$ (mm) and layer thickness $t$ (mm):

$$E = \frac{P}{v \cdot h \cdot t} \tag{3.1}$$

Of these terms, only the laser power and scan speed are controllable mid-build, while the hatch spacing and layer thickness must be kept constant.

While DoE methods have been successfully used to enhance product quality,

they have substantial limitations. Firstly, DoEs require a significant number of objects to be built, which is an expensive and time-consuming process which may span several days and cost thousands of pounds [5]. This is aggravated by the fact that these DoEs must be repeated if different materials or geometries are to be used, as the PBF process dynamics are significantly altered by changes in material or geometry. Secondly, while the use of the optimal input energy density results in fewer manufacturing defects, it does not guarantee the absence of defects [61] due to its open-loop nature, since it is not possible to detect and correct process anomalies which may lead to the formation of defects [4].

## 3.3 Process Monitoring and Control

The aim of real-time monitoring is to predict the formation of manufacturing defects, either as a means in itself for quality control or for the subsequent introduction of corrective measures to prevent defect formation. Because defects are not directly observable during the build process, the literature on PBF has focused on identifying anomalies in measurable *process signatures* which may be indicative of the formation of defects [4, 43]. These process signatures can be either directly measured, such as the melt-pool temperature and geometry, or analytically derivable from observations, such as residual stresses or geometric irregularities.

Previous empirical research efforts [48, 50, 51, 62] have identified the main process signatures affecting defect formation to be the melt-pool temperature and melt-pool geometry. To measure the former, a wide range of measurement instrument configurations and algorithms have been developed. For instance, both melt-pool temperature and geometry can be measured using complementary metal-oxide semiconductor (CMOS) cameras [63, 64]. On the other hand, other technologies focus solely on acquiring temperature measurements. While pyrometers are capable of measuring the maximum melt-pool temperature [64, 65], Infra-red (IR) cameras are generally preferred due to their high accuracy [66, 67]. However, IR cameras require significantly more involved post-processing techniques.

This project focuses solely on temperature-related process signatures measurable by a pyrometer, as pyrometers are the only monitoring equipment available in

the manufacturing system used. Three thermal anomalies, identified by Chua et al. [4], are considered in this project: substantial deviations of the surface temperature from the optimal temperature, prolonged localised overheating, and highly uneven surface temperatures (large spatial variations of melt-pool temperature).

Despite the well-studied nature of defect formation in relation to melt-pool geometry and temperature, and the technological ability of monitoring said process parameters in real-time, research efforts on the use of closed-loop control have been extremely limited: only one study has demonstrated closed-loop control in a PBF process. The work by Craeghs et al. [68] aimed to keep the melt-pool size constant –as measured by an optical sensor–, by modulating the laser power. The authors characterised the dynamics of the system using a second-order model fit over previously collected system data and subsequently used the model to design a Proportional Integral controller. The use of real-time closed-loop control was shown to maintain the size of the melt-pool within its optimal range, preventing overheating and enhancing the overall surface finish of the product. This study demonstrates that effective closed-loop control of the process signatures could lead to substantial improvements in product quality.

However, no previous studies have demonstrated thermal control in PBF. The lack of research efforts on the use of control for PBF processes can be attributed to two factors: firstly, most PBF systems on the market lack the capability of changing process parameters mid-build [43], which in turn could also be attributed to the lack of research in process control for PBF. Secondly, PBF processes pose a remarkable control challenge: they are high-dimensional Multiple-Input Multi-Output systems with very non-linear dynamics. The design of suitable control is further hindered by the fact that a model of the processes dynamics does not exist due to the complexity of the process.

# Chapter 4

# Introduction to the Case Study

This Chapter presents the process control case study considered in the remaining Sections. The characteristics of the real-world manufacturing machine used are presented in §4.1. Subsequently, the PBF thermal control problem is defined as a reinforcement learning problem in §4.2. Finally, an overview of the software developed to conduct the real-time control experiments is presented in 4.3.

## 4.1 The AconityMINI Manufacturing System

The powder bed fusion Additive Manufacturing system considered in our industrial control case study is the AconityMINI by Aconity3D GmbH. This machine is well-suited for the implementation of closed loop control, as it offers both real-time process monitoring capabilities and, more critically, the ability to change some process parameters mid-build.

#### Process Monitoring using Pyrometers

The AconityMINI is equipped with two pyrometers which measure thermal radiation at the layer surface as the laser beam scans each layer. Thus, the pyrometer data comprises a set of 2D coordinates and the measurement readings obtained at each location. As a result, spatial variations of surface temperature can be precisely observed, unlike temporal variations of surface temperature, which can only be determined between different layers.

The pyrometer provides analog measurements of thermal radiation in mV,

which can be converted to absolute temperatures using the StefanBoltzmann law. The pyrometers are calibrated such that a change of 1 mV in the measurements corresponds to a change of 1 °C. Therefore, the researchers at the Mercury Centre, the research lab which owns the AconityMINI used, often work directly with the raw mV measurements, as it is sufficient for determining variations in temperature. For consistency and simplicity, our study also provides the temperature measurements in mV. Finally, the researchers at the Mercury Centre recommended to only use the second pyrometer, as it was believed to be the more accurate one.

The sample rate of the pyrometers is 100kHz and its measurements have 10-bit resolution. As a result, the amount of data recorded for a single layer generally is in the order of many megabytes. As discussed in §4.3, it may take up to a few seconds to read files of that magnitude, which can be problematic for real-time monitoring and control. The AconityMINI does not provide the functionality of reducing the sample rate or resolution of the pyrometers.

#### Changing the Process Parameters

The AconityMINI offers the functionality of changing some process parameters mid-build. Normally, multiple objects are built simultaneously, and different laser powers and scan speeds can be assigned to each of the objects being built. The maximum laser power and scan speed are 200W and 4m/s, respectively.

The processes parameters are changed using a Python API provided by Aconity3D GmbH. This API had not been previously used by researchers at the Mercury Centre, and substantial project time was spent developing and testing the software required to change the process parameters through the API.

## 4.2 Description of the Process Control Problem

In this section, the control problem to be considered in subsequent Chapters will be fully-defined as a reinforcement learning problem, according to the characteristics of the AconityMINI defined in §4.1 and the insights gained into the formation of manufacturing defects, reviewed in §3.2. To frame the process control problem within the RL framework, the following elements must be defined: the system

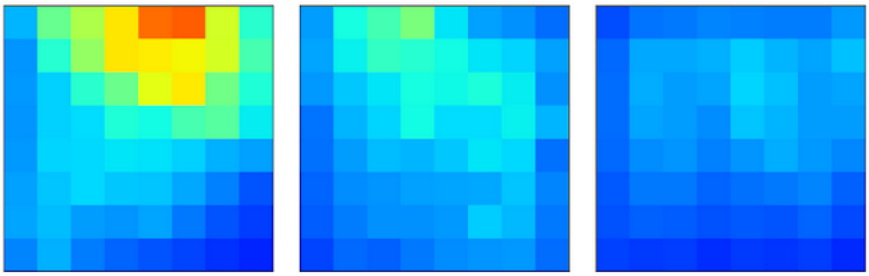


Figure 4.1: Example of different surface temperatures. These layers are divided into 64 cells. Localised hotspots can be detected (leftmost image).

states, the control inputs, the dynamics function and the reward function.

### 4.2.1 System States

The only form of feedback available in the AconityMINI is that of the pyrometer, which measures the temperature across the layer surface. To detect the presence of localised hotspots –or coldspots– the layer surface is divided into a number of cells and the mean temperature for each cell is computed (Figure 4.1). Naturally, a higher number of cells will improve the detection of localised hotspots due to the enhanced spatial resolution, but will also increases the dimensionality of the problem, which may be problematic from a computational perspective.

The choice of 16 cells was found to be a good compromise between sufficient spatial resolution and relatively low dimensionality. The impact of the number of cells on model performance is summarised in Table 4.1 (for further reference on the methodology used, see A.1). Two important insights can be gained: firstly, that the spatial resolution of the layer temperature is indeed key for building an accurate thermal model of the process (the model trained for 4 cells is substantially worse than the other two). Secondly, employing too many cells can also hinder the performance of the model.

Table 4.1: Model performance for different number of cells.

| Number of cells | MSE |
|---|---|
| 4 | 389±12 |
| 16 | **274±11** |
| 64 | 293±7 |

Therefore, 16 cells will be used, and the system state is thus the vector

$x = \{T_c\}_{c=0}^{16}$ where $T_c$ is the mean temperature at each cell $c$.

This state representation is superior to the one used in previous studies on PBF temperature monitoring, where the rich spatial temperature measurements provided by pyrometers are lumped into just two indicators: mean layer temperature and temperature variance. While the temperature variance provides a measure of temperature uniformity, a lack of temperature uniformity may not be caused by localised overheating phenomena but rather small temperature deviations across the entire build surface. Conversely, very localised hotspots in an otherwise uniform surface temperature may not affect the temperature variance significantly. Furthermore, temperature variance does not hold any information on the spatial location of the temperature irregularities, which in the future could be exploited to provide localised parameter corrections within the layer surface if machines with such capabilities enter the market.

Furthermore, the state representation chosen does not require the use of feature engineering to process the data from the pyrometer. This is desirable in order to assess the ability of PETS to control the PBF process using only raw measurements.

### 4.2.2 Control Inputs

The control inputs $u = \{P, v\}$ are the laser power $P$ and scan speed $v$. These are the only parameters that can be changed mid-build in the AconityMINI. Nonetheless, they have been shown to be sufficient to effectively control surface temperature [4]. If it were possible to change the process parameters in localised regions, the definition of the control input could be easily extended to $u = \{P_c, v_c\}_{c=0}^{16}$ where would $c$ denote the 16 cells defined earlier.

### 4.2.3 System Dynamics Function

The system dynamics function $f(x_{t+1} \mid x_t, u_t)$ models the surface temperature at a particular layer $t + 1$ given the control inputs used to build said layer and the temperatures observed at the previous layer $t$. The notation $t$ is used to describe discrete instants of time –each layer– for consistency with RL notation.

The system dynamics function of the PBF process considered is unknown. For simulated experiments, the dynamics function is approximated from process data using deep neural networks, as detailed in §5.

### 4.2.4 Reward Function

The reward functions provides a measure of what the controller should achieve, that is, prevent the formation of temperature-related defects by:

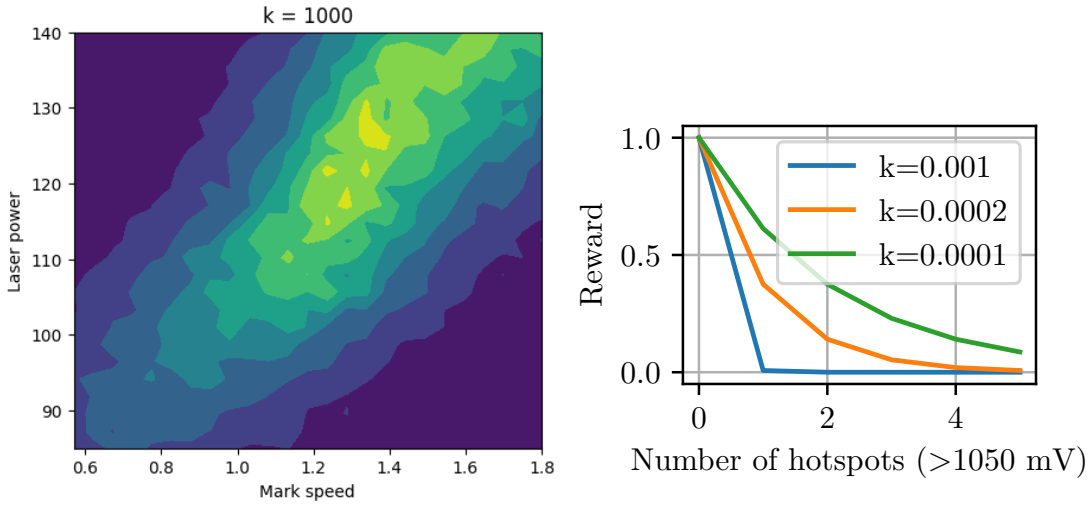
- Keeping the surface temperature close to its optimal-value, to prevent the formation of porosities and lack-of-fusion defects.
- Keeping a uniform temperature across the surface to prevent cracks.
- Preventing localised overheating, which can lead to severe deformations.

As a result of the state formulation employed, all three control objectives can be condensed to maintaining the temperature of each of the state cells at the optimal surface temperature. To prevent localised overheating, the reward function should be particularly aggressive in assigning low rewards even if a single cell within the layer overheats. The exponential negative quadratic error between the system state $x$ and the optimal temperature $T_{\text{opt}}$ is well suited for this task:

$$R(x) = e^{-k(x-T_{\text{opt}})^T(x-T_{\text{opt}})} \tag{4.1}$$

This reward function is only parametrised by a sensibility parameter $k$. A suitable parameter was selected by, using the simulation environment developed in §5, simulating the manufacture of objects with 625 different combinations of scan speed and laser power. The expected return for each parameter combination was then computed for different values of $k$, and compared to the energy density envelopes identified by previous PBF studies.

It was found that $k = 0.001$ resulted particularly well-defined energy density envelopes, shown in Figure 4.2 (a), comparable to those identified by Gong et al. [61] (Figure 3.2).Furthermore, the choice of $k = 0.001$ provides excellent hotspot rejection: if just one hotspot is detected, a reward of zero will be assigned to the corresponding system state, as shown in Figure 4.2 (b).




(a) (b)

Figure 4.2: (a) Energy density envelopes for $k = 0.001$. Darker colours indicate lower return. (b) Reward for different numbers of hotspots.

However, if no previous knowledge on the optimal energy envelopes had been available, a reasonable value of $k$ could have been arbitrarily selected given the range of expected system states, since $k$ does not need to be precisely tuned but rather have an appropriate order of magnitude so that the exponential function does not saturate. Saturation would cause the reward function to always assign either rewards of zero ($k$ too large) or one ($k$ to small).

The reward function could also include the control inputs. For instance, higher scan speeds could be favoured in order to achieve faster builds. However, this is beyond the scope of our process control case study.

The optimal surface temperatures for the powder materials tested were provided by the researchers at the Mercury Centre according to their in-house tests.

## 4.3 Software Developed for Real-Time Control

The case study considered is a real-time control problem, and thus the time required to compute the control inputs is of substantial importance. Upon consultation with expert researchers at the Mercury Centre, it was determined that it was undesirable for more than 15 seconds to pass between one layer being finished and the next one starting. This is due to excessive cooling of the layer surface. The target of 15 seconds could not be met in the real-world experiments conducted, with

the time between layers ranging between 25 and 40 seconds. Nonetheless, these relatively high times between layers likely did not significantly affect build quality, since the quality of the objects manufactured using fixed process parameters was similar to that of previous experiments conducted at the Mercury Centre.

Roughly, 80% of the computing time was spent reading and processing the pyrometer data, 15% optimising the control inputs using MPC and 5% was spent by the AconityMINI software changing the process parameters. Relative times are provided since the actual times will heavily depend on the number of manufactured objects. The time required to read and process the pyrometer data cannot be reduced unless Aconity3D GmbH provides the functionality necessary to reduce the pyrometer sampling rate. The time spent computing the control inputs with MPC could be drastically reduced to just a few milliseconds per object if the algorithm were to be run in a computer with a GPU.

The architecture of the software developed for real-time control of the AconityMINI is shown in Figure 4.3. The DRL algorithm is run on a computing cluster, due to the exceptionally underpowered CPU of the local computer connected to the AconityMINI. Communication between the remote cluster and the local machine is done using the SSH File Transfer Protocol (SFTP), which introduces a relatively small latency as only files in the order of a few kilobytes are transferred.

Two programs are run concurrently on the local machine to prevent the computationally expensive processing of the pyrometer data from blocking communication with the AconityMINI. Thus, one of the two programs is dedicated exclusively to communicating with the AconityMINI, while the other is used for reading and processing the pyrometer data and communicating with the computing cluster.

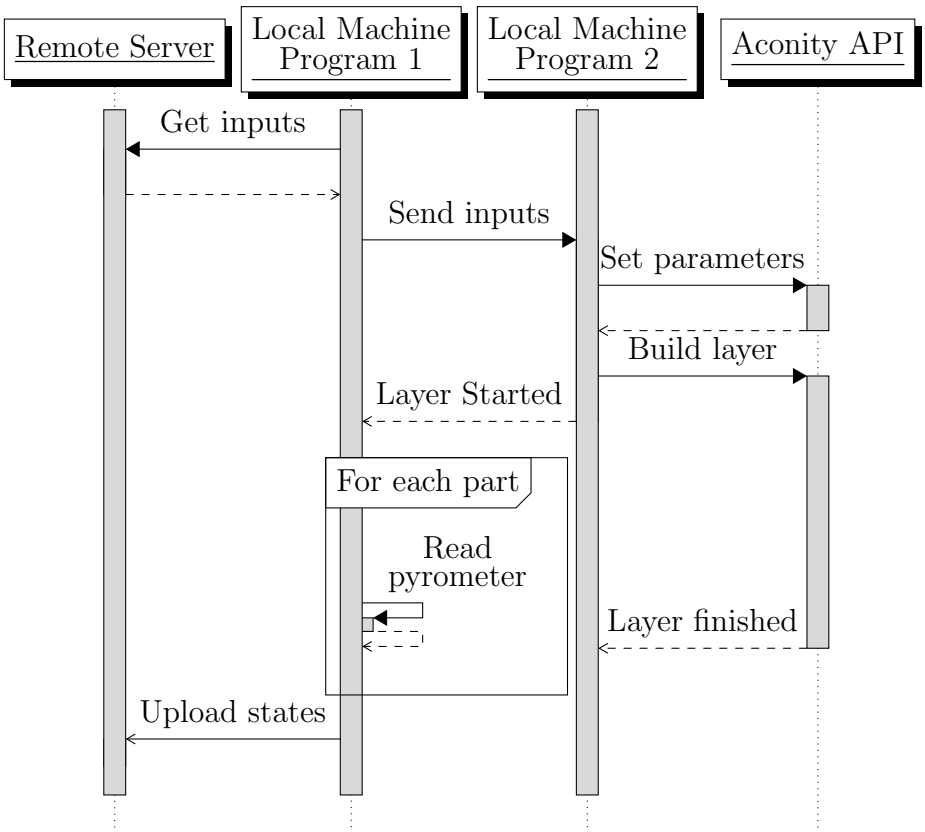



Figure 4.3: Sequence diagram of the software developed for real-time control.

# Chapter 5

# Case Study: Modelling and Simulation

This Chapter aims to evaluate, in simulation, the control performance and sample-efficiency of PETS for the thermal process case study considered. This simulation work was motivated by the inability to perform several experiments in the real manufacturing system. However, this work is performed solely to better evaluate the PETS algorithm, as DRL does not require system identification and controller design procedures, but rather learns autonomously by interacting with the system.

Firstly, the methodology used to collect the process data required to build a model of the process is presented in §5.1. Subsequently, the architecture of the deep model used to model the process dynamics is presented in §5.2. Thereafter, the simulation environment used to perform the simulation experiments is described in §5.3. Finally, the sample-efficiency and control performance of PETS evaluated using the simulation environment. The performance of PETS is compared to the use of optimal process parameters and to the use of Proportional Integral control.

Further information on the modelling and simulation experiments conducted is provided in appendix A.1 and appendix B.

## 5.1 Collection of Process Data

The aim of the data collection experiment was twofold: to collect as many samples as possible due to the data-hungry nature of deep models, and to achieve good

Figure 5.1: Top view of the 4 individual cubes forming a single object.

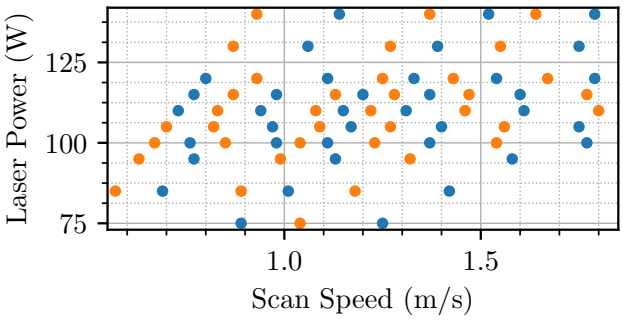



Figure 5.2: Parameters used for build 1 (blue) and build 2 (orange).

coverage of the state and input spaces. The material used was Haynes 282, and the geometry a cube, since researchers at the Mercury centre had performed extensive tests with this material and shape to determine the optimal process parameters.

### Dimension and Number of Printed Objects

Only two builds could be performed in the AconityMINI due to the limited availability of the machine. Thus, to fit as many objects as possible within each build, the minimum dimension achievable of 5x5x5mm was used. While up to 150 cubes of such dimensions could fit within the build plate, the AconityMINI can print a maximum of 40 objects. To overcome this issue, a single object was defined to be 4 individual cubes, as shown in Figure 5.1. The distance between cubes was set to be 2.5mm, sufficient to ensure the temperature of the individual cubes did not significantly affect each other. For each build, 35 objects fitted in the plate, totalling 280 cubes printed. For consistency with previous tests at the Mercury Centre, a layer thickness of 30$\mu$m was used, resulting in 167 layers.

It is important to note that because the 4 cubes are built as a single object, the laser scan strategy differs from that which would be used to print a single cube. It was believed that the process dynamics of each individual cube would nonetheless be very similar. This proved not to be the case, as discussed in appendix A.4.2.

### Laser Power and Scan Speed Used

The functionality of changing process parameters mid-build had not been made available at the time the data collection builds were performed. Thus, it was not

possible to vary the parameters over time, randomly or otherwise, to ensure good coverage of the state and input spaces. The parameters were therefore kept fixed.

The 70 laser power and scan speed combinations used are shown in Figure 5.2. The parameters are distributed around the optimal laser power of 110 W and scan speed of 1125 mm/s, as determined by researchers at the Mercury Centre. The ranges of the laser power and scan speed were 75-140 W and 0.57-1.8 m/s.

#### Interruptions of the Builds

The two builds were interrupted midway through the printing process. For the first one, the AconityMINI web application crashed for an unknown reason at layer number 112. To ensure the surface temperature completely cooled down, after two hours the build was restarted. The second build was interrupted at layer number 31 due to an automatic Windows update. It was restarted after three hours.

Because of the material used and the sufficient cooling time allowed, experts at the Mercury Centre concluded that the dynamics of the process upon restarting the build should be similar to that of starting the build from the base plate. Thus, for modelling purposes, we will consider four distinct builds: 1.1, 1.2, 2.1 and 2.2. The number of datapoints obtained in each build were 13972, 4776, 3948 and 8780.

## 5.2 Architecture of the Deep Model

The PETS algorithm uses an ensemble of deep neural networks to model the dynamics of the system being controlled. The NN inputs are the system state $x_t \in \mathbb{R}^{16}$ and control input $u_t \in \mathbb{R}^2$, while the outputs are the mean $\mu \in \mathbb{R}^{16}$ and covariance $\sigma \in \mathbb{R}^{16}$ encoding the next system state $x_{t+1} \sim \mathcal{N}(\mu, \sigma)$. Therefore, the input layer of the NN has 18 neurons and its output layer has 32 neurons. The number of hidden layers and the number of neurons per hidden layer were considered hyperparameters (tuned in §5.2.1). The NN layers are fully connected.

The weights of each neuron are initialised using a truncated normal distribution with zero mean and standard deviation equal to $0.5/\sqrt{n_{in}}$, where $n_{in}$ is the number of inputs to the layer. This generally improves model performance [69] by providing a stronger initial learning signal through the backpropagation [70].

L2 regularization is used to prevent the weights from becoming too large, which generally occurs when overfitting [71]. The Swish activation function is used, since it has been shown to generally perform better than the, until recently, popular ReLU [72]. The optimiser used to train the NN is Adam, a popular choice among the deep learning community due to its computational and memory efficiency [73].

The NN inputs were standardised, since this was shown to lead to substantially better model performance. Each of the three features –layer temperature, laser power and scan speed– serving as inputs to the NN were scaled independently. The methodology and full results of the tests performed are detailed in appendix A.1.

Several model-based RL studies [4, 74] report that learning the one-step differential dynamics of the system can lead to better model performance. This was found to be the case for the PBF process considered. Thus, the training targets of the predictive model are $(x_t, u_t) \rightarrow \Delta x_t$, where $\Delta x_t = x_{t+1} - x_t$, instead of $(x_t, u_t) \rightarrow x_{t+1}$. The methodology and full results of the tests conducted to determine the most suitable model target formulation are detailed in appendix A.2.

### 5.2.1 Hyperparameter Optimisation

Selecting the optimal set of hyperparameters for a deep NN often requires an extensive trial-and-error process guided by expert knowledge [75]. A poor choice of hyperparameters, however, can severely undermine the performance of the model in terms of its speed of learning and accuracy. The search for optimal hyperparameters is particularly problematic when the training time of the NN is very high, since it can drastically limit the number of hyperparameter combinations which can be tested. This is the case for our model, where the training time can be of up to several hours depending on the choice of hyperparameters.

In order to optimise the model hyperparameters within a reasonable timeframe, Bayesian optimisation was used, which has recently been shown to outperform, at a fraction of the number of hyperparameter evaluations, other popular hyperparameter search techniques –such as grid search– and to even surpass the performance of human experts in some instances [76]. To achieve high data-efficiency, Bayesian optimisation uses Gaussian Processes to iteratively model the objective function being optimised, which then employs to cheaply evaluate the

objective function at different candidate hyperparameter combinations [77].

The optimal set of hyperparameters found were: 3 hidden layers, 750 neurons per hidden layer, learning rate of $3.61 \cdot 10^{-4}$, batch size of 32, 45 training epochs, and weights decays of $8.21 \cdot 10^{-5}$ (input layer), $1.18 \cdot 10^{-5}$ (hidden layers) and $1 \cdot 10^{-5}$ (output layer). The methodology and full results of the tests conducted for hyperparameter optimisation are provided in appendix A.3.

## 5.3 The Simulation Environment

The simulation environment is as follows: cubes are printed one at a time, where the printing of a single cube is referred to as a *trial*. The number of layers of each cube corresponds to the trial time-horizon $H$. The model used to simulate the process dynamics is trained solely from the data collected in builds 2.1-2.2, since the data from builds 1.1-1.2 suffered from severe measurement noise (discussed in detail in appendix A.4.1). The initial system state is sampled from $x_0 \sim \mathcal{N}(870, 12)$, as this is representative of the temperatures observed for the first layer of builds 1.1-2.2. Thereafter, the model is used to propagate the system state $x_{t+1} \sim f(x_t, u_t)$ according to the control inputs $u_t$ used for each layer $t$.

The controller performance is measured using the reward function presented in §4.2.4. The *return* of a trial is the sum of the rewards associated with all system states $x_t$ observed during the trial.

The simulation environment was used to fine tune the parameters of the PETS algorithm (the methodology and results are detailed in appendix B) to ensure PETS would perform optimally in the very few experiments which could be conducted on the real manufacturing system. It was found that the parameters provided by the authors of PETS worked well for the case study considered, and only minor changes were required.

## 5.4 Evaluation of PETS Performance

The simulation environment was used to evaluate the sample-efficiency (§5.4.1) and control performance (§5.4.2) of PETS for the PBF process considered. This

was necessary due to the impossibility of conducting extensive tests on the real manufacturing system.

### 5.4.1 Data-Efficiency

The experiments conducted to measure the data-efficiency of PETS aimed to determine how many individual cubes had to be built before PETS was able to provide comparable control to using constant optimal process parameters, which is currently the industry standard. Generally, 13 objects are built during a standard DoE, the method used to empirically find the optimal process parameters.

The cubes simulated were of 50 layers each. This is the worst-case scenario from a sample-efficiency perspective, as extremely little data is obtained for each cube. In reality, the number of layers of an object is generally in the order of at least many hundreds of layers, often thousands.

The experiment results are provided in Figure 5.3. The most significant result is that PETS provides better thermal control than the use of constant process parameters after just 5 printed cubes, substantially fewer than the 14 objects required to empirically determine the optimal process parameters.

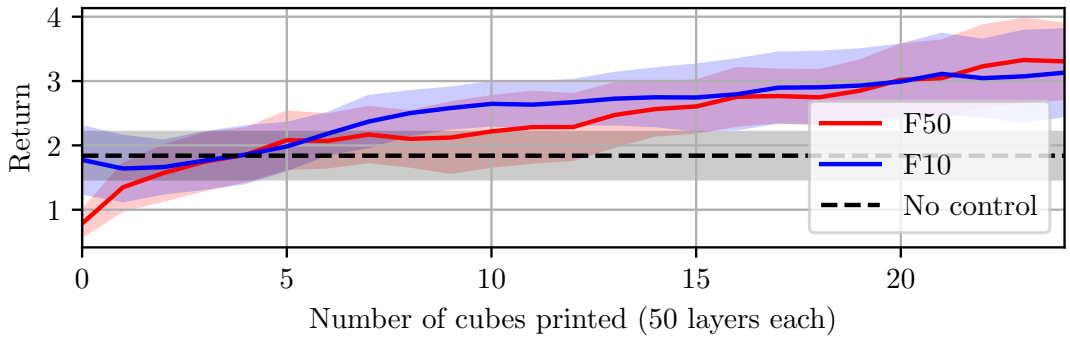



Figure 5.3: PETS control performance with respect to the number of cubes printed. The solid lines represent the mean return obtained for the 10 runs performed, while the shaded areas represent one standard deviation. "No control" refers to the use of constant optimal parameters.

It was also found that training the model multiple times per cube resulted in both higher sample-efficiency and better asymptotic control performance. Remarkably, when the model was trained every 10 layers, the thermal control provided by PETS on the very first cube was comparable to the use of constant

optimal process parameters.

Therefore, the simulation results obtained indicate that PETS results not only in significantly better thermal control of the PBF process, but also that much fewer objects have to be manufactured to train the controller compared to the use of DoE methods. Further experimental results for different parameter combinations of PETS can be found in appendix B.3.1 and in appendix B.3.2.

### 5.4.2 Control Performance

The performance of PETS was compared to that of using constant optimal process parameters (*no control*), which were determined through trial and error in simulation. Furthermore, the performance of a Proportional-Integral controller was also benchmarked. The proportional and integral gains of the PI controller were hand-tuned, as detailed in appendix B.5.

Both types of control are relevant to the case study: constant process parameters are used by DoE methods –the current industry standard–, and PI control is the only type of closed-loop control previously demonstrated in a PBF process [68].

The control performance of each of the three approaches –constant optimal parameters, PI controller and PETS– is summarised in Table 5.1. The PETS algorithm was trained with the data of 50 cubes in order to asses the asymptotic performance of the algorithm. The metrics provided are:

- RMSE: root mean squared error between the observed cell temperatures (16 different temperatures per layer, see §4.2.1) and the target temperature.
- $> +50$ (%): Percentage of the observed cell temperatures which were hotspots, where a hotspot is defined as 50 mV hotter than the target temperature.
- StdDev: the standard deviation of the observed cell temperatures of each layer. Gives a measure of temperature uniformity across the layer.
- Return: sum of the rewards obtained in each layer.

The best performing control strategy across all metrics is PETS, which according to the simulation results, results in layer temperatures significantly closer to the target temperature, fewer hotspots and higher temperature uniformity. Thus, all control objectives embedded in the reward function are met.

The simulation results obtained also indicate that PI control is insufficient to reduce the occurrence of hotspots and to enhance temperature uniformity across the layer. This is due to its Single Input Single Output nature, since the feedback signal is the mean temperature in the layer, which does not provide any information about the spatial distribution of the surface temperature.

Table 5.1: Metrics for the Haynes 282 experiment (over 20 trials).

| | RMSE | $> +50$ (%) | StdDev | Return |
|---|---|---|---|---|
| No control | 35.1±3.6 | 1.0±0.4 | 12.6±0.4 | 1.21±0.56 |
| PI control | 26.6±1.1 | 1.0±0.6 | 14.2±0.5 | 1.55±0.27 |
| PETS control | **24.9±0.6** | **0.1±0.1** | **12.0±0.3** | **4.71±0.92** |

The time responses for each of the control approaches tested are shown in Figures 5.4, 5.5 and 5.6. The mean surface temperature observed at each layer is provided, together with the control inputs used at each layer, if applicable.

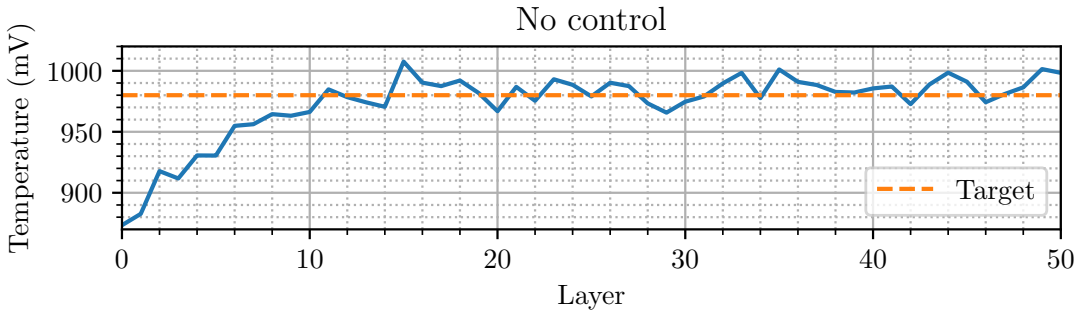



Figure 5.4: Time response of fixed optimal process parameters.

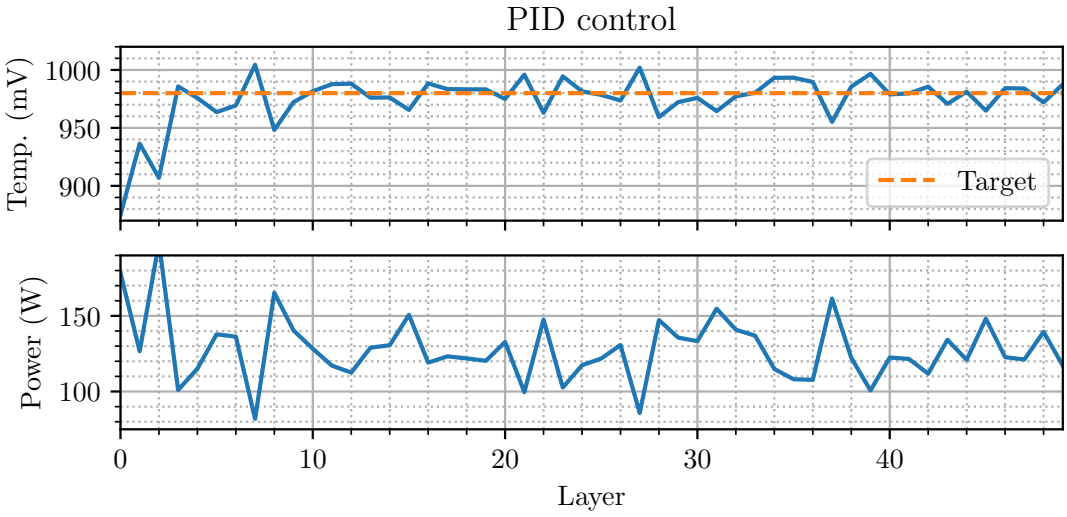



Figure 5.5: Time response for PID control.

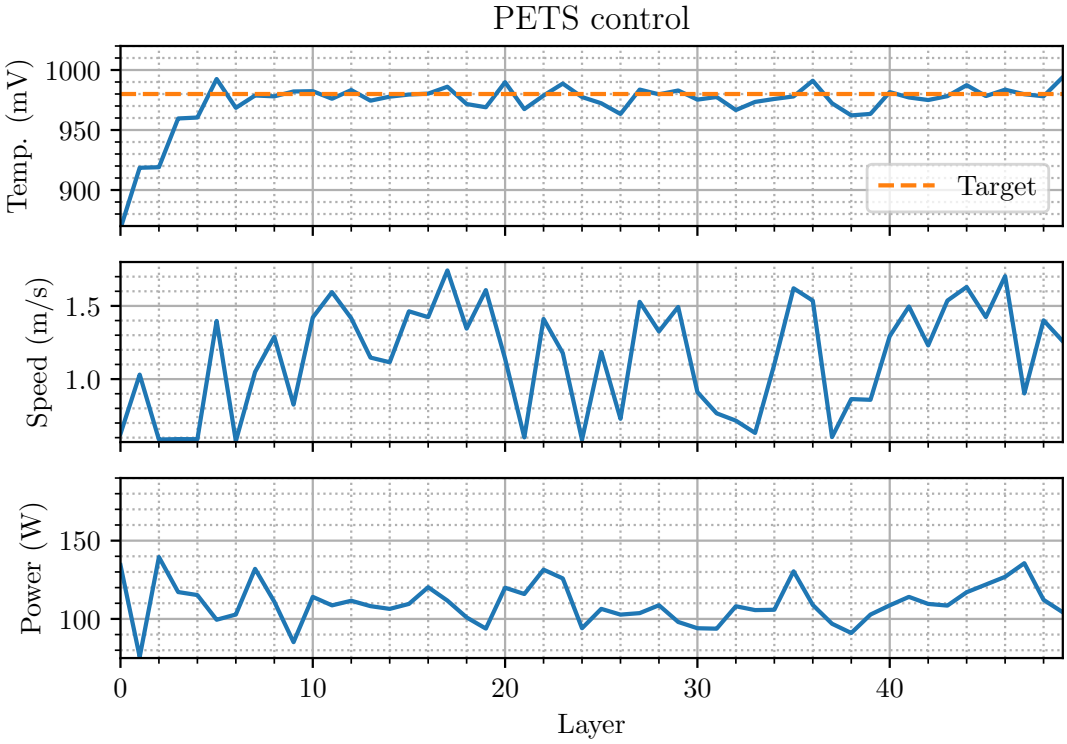



Figure 5.6: Time response for PETS control.

# Chapter 6

# Case Study: Process Control

This Chapter aims to evaluate the real-world control performance and sample-efficiency of PETS for the the PBF thermal control case study considered. Two experiments are performed for two different materials: CM247 (§6.1) and Haynes 282 (§6.2). For each experiment, the performance and sample-efficiency of PETS are evaluated and compared to the use of constant optimal process parameters, the current industry standard in PBF manufacturing. An initial analysis of the quality of the manufactured products is also provided.

## 6.1 CM247 Experiment

The CM247 material was selected for this experiment due to its extreme tendency to crack. The use of optimal process parameter has been shown to not be sufficient to prevent the formation of cracks, and it has been hypothesised that maintaining an optimal surface temperature could reduce crack formation.

Twenty cuboids of dimension 10x10x5mm (250 layers) were manufactured simultaneously, 10 using PETS and 10 using the optimal process parameters previously determined by researchers at the Mercury Centre through Design of Experiment (DoE) methods. To improve sample-efficiency and reduce computational time, a single model of the process dynamics is learnt with the data collected from all the cubes being controlled using PETS, rather than training an individual model for each cube. This approach could also be used in commercial additive manufacturing settings, where objects are generally produced in batches.

### 6.1.1 Control Performance

The temperature responses obtained from the experiment are shown in Figure 6.1. For simplicity, only the response of two cuboids is shown, one using constant optimal process parameters, and the other controlled using PETS. The responses for the remaining 18 cuboids are provided in appendix C.1

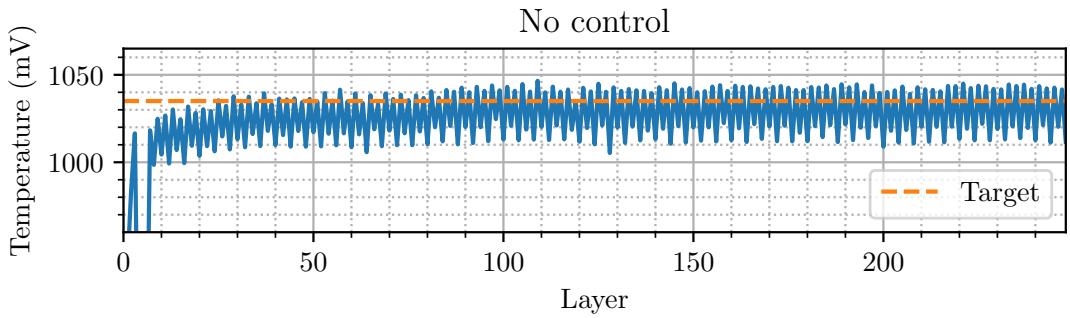
The first observation which can be made form the results is the extraordinary speed at which PETS learned to control the system. Having no previous knowledge about the system dynamics, after just 10 layers PETS is able to maintain the surface temperature close to the optimal temperature. This is remarkable, as the cuboids built without control do not reach steady state until roughly layer number 30. Furthermore, the uncontrolled cuboid also exhibits oscillatory behaviour at steady state, which PETS prevents.

The metrics computed for both control approaches are presented in Table 6.1. As clearly observed in the time responses, PETS reduces the RMSE substantially. However, the number of hotspots is higher for the cuboids built using PETS, although the percentage of hotspots is overall very low (1%). Furthermore, PETS did not succeed in increasing surface temperature uniformity. Indeed, the StdDev metric was higher for the samples built using PETS, although the difference with the cuboids built without control is relatively small.

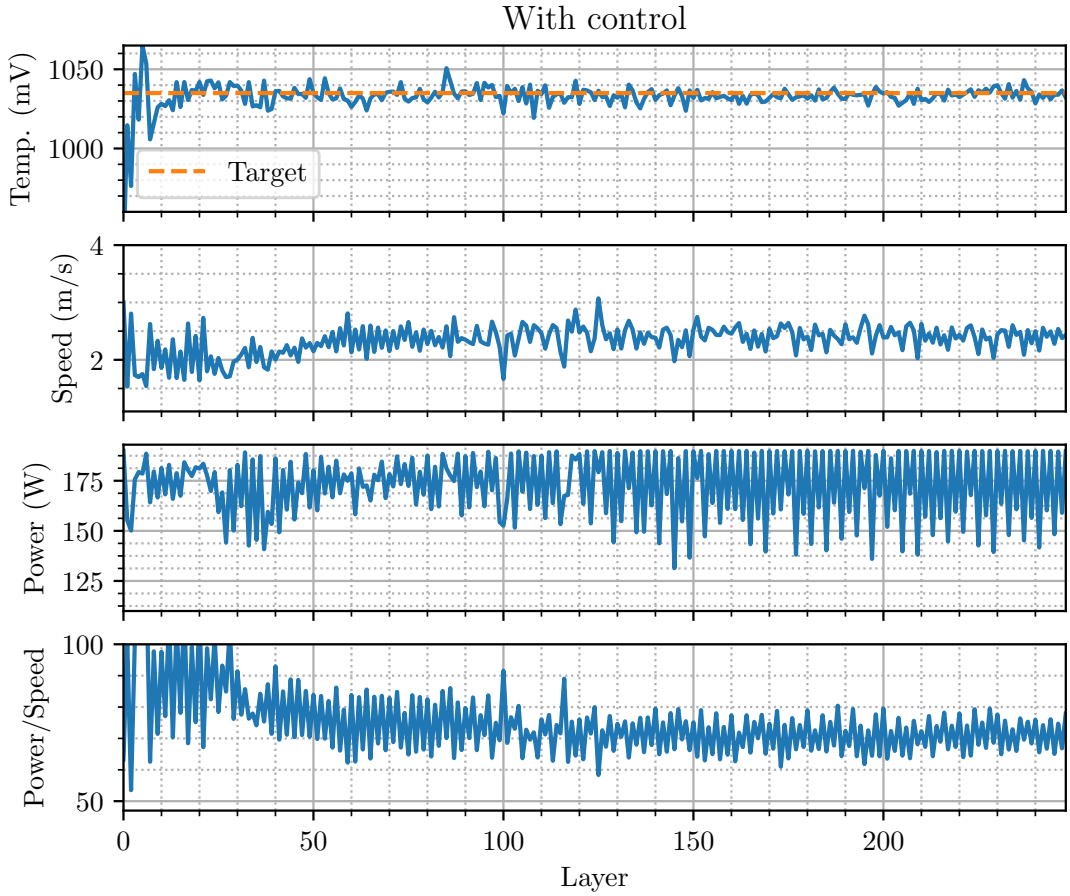
Table 6.1: Metrics for the CM247 experiment.

| | RMSE | $> +50$ (%) | StdDev | Return |
|---|---|---|---|---|
| No control | 33.7±0.8 | **0.1±0.2** | **17.8±0.4** | 0.31±0.08 |
| Control | **23.0±1.1** | 1.2±0.6 | 19.9±0.8 | **1.28±0.58** |

The return obtained for the cuboids built with PETS is higher to that of the cuboids printed without control. Thus, the reward function may have favoured a significant reduction of RMSE at the expense of a relatively small increase in the number of hotspots and a decrease in temperature uniformity. It is unclear if this behaviour is the most adequate for enhancing build quality, but the relative weight given to each control objective could nonetheless be easily altered by changing the parameter $k$ of the reward function.




(a) No Control



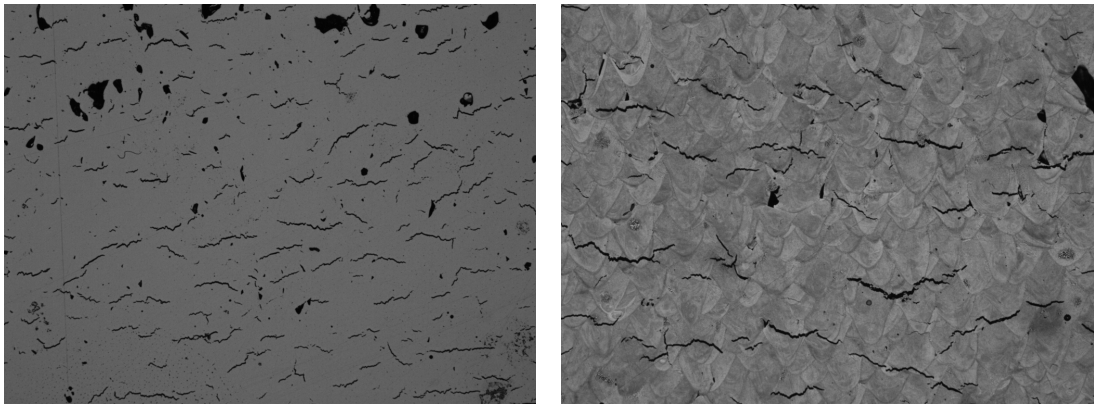

(b) With Control

Figure 6.1: Time responses for the CM247 experiment.

### 6.1.2 Quality of the Manufactured Objects

The analysis of the build quality of a manufactured object is a time-consuming process, and the researchers at the Mercury Centre were only able to obtain preliminary results. Because the key manufacturing defect of CM247 is cracking, the surface of the cuboids was examined to attempt to asses if the samples built with

thermal control had fewer cracks. The images obtained under a microscope are provided in Figure 6.2. Despite the absence of quantitative data, it is possible to qualitatively determine that both images show a similar number of cracks.



(a) No Control (b) With Control

Figure 6.2: Object surface under 1000x magnification.

Figure 6.3: One of the manufactured cuboids.

Thus, it is likely that the use of thermal control did not reduce, nor enhance, the formation of cracks. This may be due to the inability of PETS to improve surface temperature uniformity, as this has been previously linked to the formation of cracks. However, further experiments are required to determine if this is the case.

## 6.2 Haynes 282 Experiment

The second experiment was done using a different material, Haynes 282, in order to asses if PETS was capable of providing effective control for multiple materials with different dynamics. Haynes 282 was selected because it is a very well stud-

ied material, and very good product quality can be obtained through the use of optimal process parameters.

Twenty cubes of dimension 5x5x5mm (167 layers) were manufactured simultaneously, 10 using PETS and 10 using the optimal process parameters previously determined by researchers at the Mercury Centre through Design of Experiment (DoE) methods. However, the DoE was conducted for 10x10x5mm cubes instead of 5x5x5mm cubes. It was believed the dynamics would be similar for both dimensions, and that it was not necessary to repeat the DoE.

### 6.2.1 Control Performance

The temperature responses obtained from the experiment are shown in Figure 6.4. For simplicity, only the response of two cuboids is shown, one using constant optimal process parameters, and the other controlled using PETS. The responses for the remaining 18 cuboids are provided in appendix C.2

The first observation which can be made form the results is, once again, the extraordinary speed at which PETS learned to control the system. Having no previous knowledge about the system dynamics, after just 5 layers PETS is able to provide very good thermal control. The second key insight is, unlike PETS, the use of no control resulted in a very substantial steady-state error. This is because the DoE experiments used to select the optimal process parameters were performed in cubes of a different dimension. This highlights that the use of constant optimal process parameters cannot correct for process uncertainties or changes in the process dynamics, unlike PETS.

The metrics computed for both control approaches are presented in Table 6.3. As clearly observed in the time responses, PETS substantially reduces the RMSE. Furthermore, the number of hotspots was also dramatically reduced. However, PETS did not succeed in increasing surface temperature uniformity. Indeed, the StdDev metric was very slightly higher for the samples built using PETS.

### 6.2.2 Quality of the Samples Printed

The researchers at the Mercury Centre analysed the density of the cubes printed. The results are shown in Table 6.3. A higher density indicates better quality,

Table 6.2: Metrics for the Haynes 282 experiment.

| | RMSE | > +50 (%) | StdDev | Return |
|---|---|---|---|---|
| No control | 54.2±8.8 | 43.8±14.1 | **17.8±1.7** | 0.27±0.14 |
| Control | **25.6±2.1** | **5.0±1.7** | 18.2±0.8 | **2.00±0.94** |

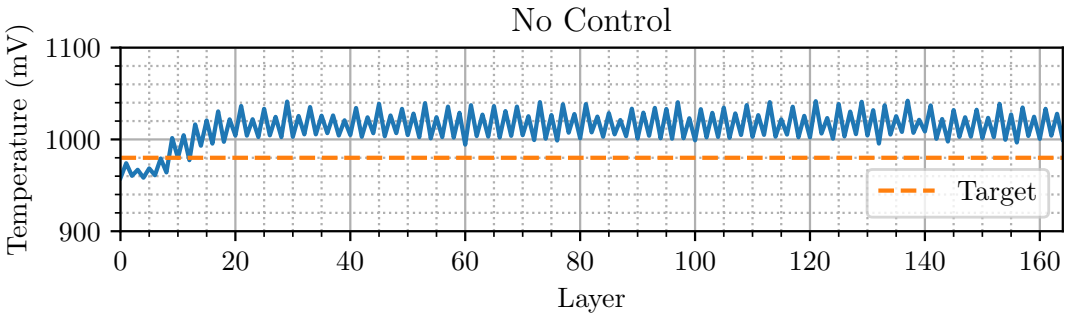
as defects such as porosities or cracks reduce the product density. All measured densities were above 95% indicating very high quality for the samples built.

Table 6.3: Density of the samples for the Haynes 282 experiment.

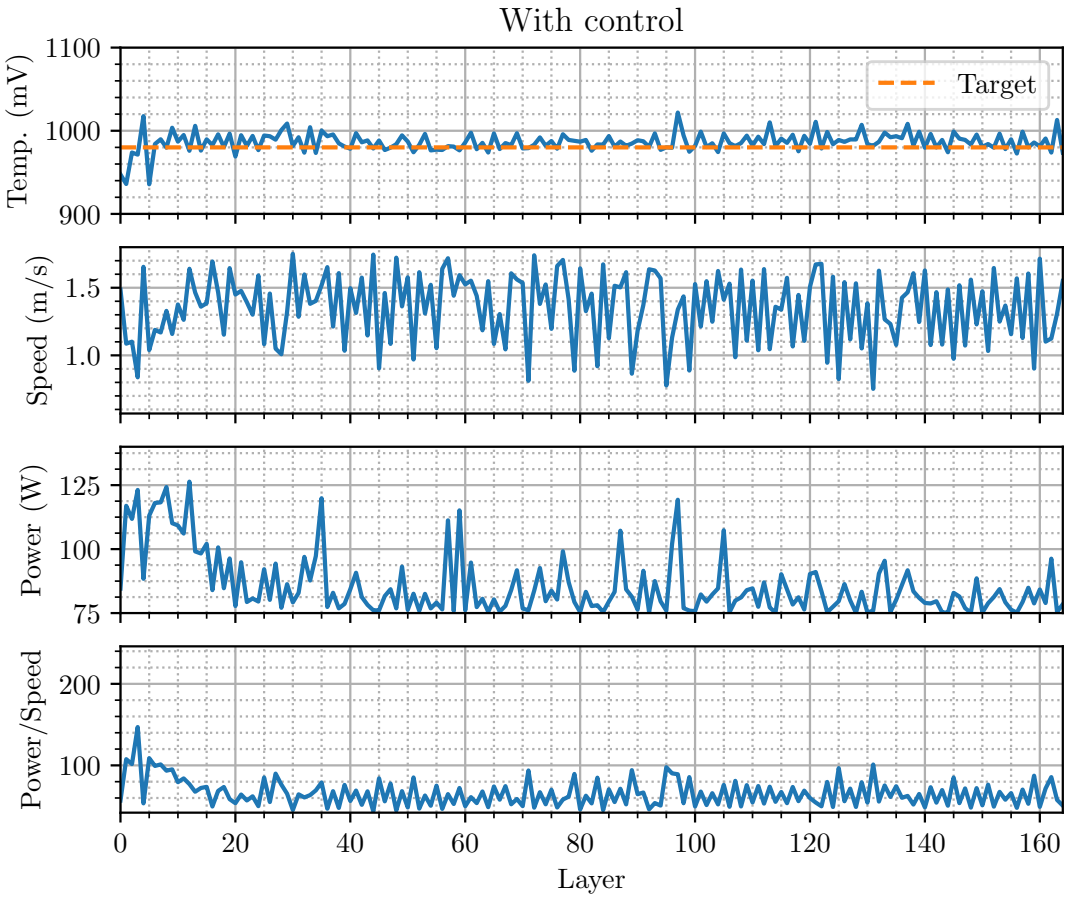
| | Density (%) | | |
|---|---|---|---|
| | Mean | Minimum | Maximum |
| No control | **97.9±0.6** | 97.2 | 98.7 |
| With control | 95.9±0.7 | 95.4 | 97.2 |

Nonetheless, the cubes manufactured using PETS exhibited a significantly lower density, despite the very large temperature error observed for the samples built using constant optimal process parameters. It is possible that surface temperature may not be a good indicator of build quality, however, further tests are required to asses this hypothesis. Nonetheless, this highlights the substantial impact that thermal process control could have in future research on PBF and build quality, since past studies have had great difficulty in finding direct links between thermal signatures and build quality due to the inability of precisely controlling the thermal signatures of the process.

On the other hand, it is possible that the manner in which PETS controls the process parameters, with extremely large changes between layers, may have a negative impact on the quality of the manufactured object. However, although the individual parameters change substantially in between layers, the overall input energy density (see the power/speed ratio in Figure 6.4) does not change as violently. Nonetheless, additional constraints could be easily placed within the MPC algorithm to limit the change in process parameters in between layers.




(a) No Control




(b) With Control

Figure 6.4: Time responses for the Haynes 282 experiment.

# Chapter 7

# Conclusions

### Deep Reinforcement Learning

This study demonstrated that state-of-the-art model-based DRL is sufficiently sample-efficient to control complex physical systems. The case study considered was especially challenging from a DRL perspective, as the cost of acquiring data from the system was remarkably large.

Nonetheless, PETS achieved exceptional sample-efficiency, learning to control the system within only a few layers. Furthermore, the system was shown to provide exceptional control performance compared to the use of constant optimal process parameters and Proportional Integral control.

The case study considered reflects the potential of DRL for real-world process control applications: PETS was able to, directly from raw observations and without incorporating any previous knowledge about the system, autonomously learn to control a complex processes with dynamics which were unknown, very non-linear and high-dimensional.

### Powder Bed Fusion

This study demonstrated closed-loop thermal control of a PBF process. The use of the laser power and scan speed as control inputs was sufficient to control the surface temperature to its optimal value. However, closed-loop control did not improve surface temperature uniformity, and only reduced the formation of hotspots in one of the two real-world experiments conducted. It is possible that altering the reward function used by the PETS algorithm might solve this issue.

The objects manufactured using closed-loop control did not exhibit higher quality. Compared to the objects manufactured using optimal process parameters, the CM247 samples had comparable levels of cracking, while the Haynes 282 samples had significantly lower density –albeit still considered very good density–. Thus, it is possible that the control of thermal process signatures may not have the impact on product quality previously thought. Another factor which could have impacted build quality was the drastic changes in control inputs between layers. In the future, the change of control inputs between layers could be constrained.

Finally, the ability to control the thermal process signatures will likely facilitate future research efforts aiming to characterise the formation of manufacturing defects in PBF, since previous studies were unable to precisely control the thermal process signatures.

# Appendix A

# System Modelling Experiments

## A.1 Input Scaling

The scalers tested –MinMaxScaler, MaxAbsScaler, StandardScaler, RobustScaler– were taken from the Scipy scientific computing toolbox for Python. Only the data collected during Build 1.1 is used to build the predictive model, in order to reduce the training time. The metrics used to compare each scaling method are the Mean Squared Error (MSE) and the R-Squared ($R^2$). 5-fold cross-validation is used, and the metrics reported are those of the testing set. The mean and standard deviation of the metrics for the 5 folds are provided in Table A.1.

The model hypoerparameters used are: 2 hidden layers, 500 neurons per hidden layer, learning rate of $10^{-3}$, training batch size of 32, 64 training epochs, and weights decays $10^{-4}$ (input layer), $2.5{\cdot}10^{-4}$ (hidden layers), $5{\cdot}10^{-4}$ (output layer).

Table A.1: Test results for different input scalers.

| Scaler | MSE | $R^2$ |
|---|---|---|
| No scaler | 1219±62 | 0.4574±0.0207 |
| MinMaxScaler | 1226±68 | 0.4543±0.0271 |
| MaxAbsScaler | 1425±68 | 0.3676±0.0271 |
| StandardScaler | **966±45** | **0.5701±0.0168** |
| RobustScaler | 988±80 | 0.5604±0.0267 |

## A.2 Model Targets

The two formulations tested for the training targets of the predictive model are $(x_t, u_t) \rightarrow \Delta x_t$, where $\Delta x_t = x_{t+1} - x_t$, and $(x_t, u_t) \rightarrow x_{t+1}$. Furthermore, the performance impact of standardising the model targets was also tested.

For this experiment, 5 fold cross-validation (CV) was used and the MSE and $R^2$ metrics were computed over the testing set. The mean and standard deviation of the metrics for the 5 folds are reported in Table A.2. Both learning the differential dynamics of the system and standarising the targets led to improved metrics.

Table A.2: Test results for the training targets of the model.

| Target $\Delta x$ | Targets standarised | MSE | $R^2$ |
|---|---|---|---|
| No | No | 1236±68 | 0.449±0.022 |
| No | Yes | 973±57 | 0.566±0.022 |
| Yes | No | 1015±60 | 0.548±0.023 |
| Yes | Yes | **957±36** | **0.575±0.027** |

## A.3 Hyperparameter Optimisation

The optimisation method used was Bayesian Optimization. First, an appropriate hyperparameter space to optimise was selected by performing preliminary tests. Thereafter, Bayesian optimisation was used to optimise the hyperparameters.

### A.3.1 Selecting the Hyperparameter Space

It is crucial to select an appropriate hyperparameter space for the optimisation not to result in a suboptimal solution. If the hyperparameter space is too narrow, the optimal solution may lie outside of it, whereas if it is too wide the optimisation may not converge to the optima within the desired number of hyperparameter evaluations. Thus, several tests were conducted to determine the most suitable hyperparameter space for which to perform Bayesian optimization. In the tests, the data collected in buils 1.1, 1.2, 2.1 and 2.2 was used to train the model.

The reference hyperparameters used in the PETS paper are 2 hidden layers, 500 neurons per hidden layer, learning rate of $10^{-3}$, training batch size of 32, 64

training epochs, and weights decays $10^{-4}$ (input layer), $2.5{\cdot}10^{-4}$ (hidden layers), $5{\cdot}10^{-4}$ (output layer). The first test kept all hyperparameters constant but the number of hidden layers and neurons per layer. The MSE for the test set following 5-fold CV is reported in Table A.3. A higher number of layers resulted in worse performance, and thus it was decided to only search $\{2, 3\}$ hidden layers. While a high number of neurons performed well, it was exceptionally expensive to train and thus it was decided to only search $\{250, 500, 750\}$.

Table A.3: Test for number of layers and neurons per layer

| | | Number of neurons | | | | | |
|---|---|---|---|---|---|---|---|
| | | 100 | 200 | 350 | 500 | 750 | 1000 |
| Hidden layers | 2 | 687 | 680 | 675 | 675 | **667** | 676 |
| | 3 | 698 | 704 | 699 | 696 | 693 | 692 |
| | 4 | 720 | 727 | 711 | 710 | 699 | 698 |
| | 5 | 740 | 732 | 725 | 715 | 714 | 724 |

Thereafter, the best performing 2 hidden layers and 750 neurons per layer, three magnitudes of the learning rate were tested: $10^{-5}, 10^{-4}$ and $10^{-3}$, with resulting MSE over the testing set in 5-fold CV of 741, 681 and 669, respectively. A very computationally-expensive 128 training epochs were used. Due to the exceptionally bad performance of a learning rate of $10^{-5}$, it was decided to only search the range $[10^{-4}, 10^{-3}]$.

The batch size was also tested for the following values: 16, 32, and 64, with resulting MSE over the testing set in 5-fold CV of 670, 668 and 675, respectively. Due to the small performance differences, it was decided to not include the batch size in the hyperparameter optimization and use a batch size of 32. The range of weight decay searched was $[10^{-5}, 10^{-4}]$ as recommended by the authors of PETS.

### A.3.2 Results of the Bayesian Optimisation

Bayesian optimisation was performed using the Skopt software package for Python. The objective function to be minimised was defined as the MSE over the testing set for 5-fold CV. The number of training epochs used were 64, as this ensured relatively fast training times while being sufficiently large number of epochs for the training to converge, and sufficiently low not to overfit excessively. The algorithm

was run with 10 initial random hyperparameter evaluations and 30 iterations of the optimization algorithm. The total run-time was of 18 hours (with GPU).

The optimal hyperparameters found are presented in Table A.4. This results in a MSE of 635 over the testing set in 5-fold CV, a modest improvement of 6% over the hyperparameters provided by the original authors of PETS.

Table A.4: Results of the hyperparameter optimisation.

| Hyperparameters | Range searched | Optima found |
|---|---|---|
| Hidden layers | {2, 3} | 3 |
| Neurons per layer | {250, 500, 750} | 750 |
| Learning rate | $[10^{-4}, 10^{-3}]$ | $3.61 \cdot 10^{-4}$ |
| Weight decay - input layer | $[10^{-5}, 10^{-4}]$ | $8.21 \cdot 10^{-5}$ |
| Weight decay - hidden layers | $[10^{-5}, 10^{-4}]$ | $1.18 \cdot 10^{-5}$ |
| Weight decay - output layer | $[10^{-5}, 10^{-4}]$ | $1.00 \cdot 10^{-5}$ |

Finally, in order to select the optimal number of training epochs for the hyperparameters found, the model was trained using the following training epochs: 25, 45 and 75, with resulting MSE over the testing set in 5-fold CV of 666, 624 and 639, respectively. Thus, the number of training epochs which will be used subsequently is 45. Nonetheless, we can see that the weight decay used prevents the model from overfitting excessively for higher number of training epochs.

## A.4 Insights onto the PBF Process Dynamics

### A.4.1 Model Performance Comprarisson for Different Builds

The performance of the model for each of the builds is summarised in Table A.5. The metrics are computed over the testing set for 5-fold CV. Build 3 and build 4 were performed during the process control experiments detailed in §6.

There is a very substantial difference in model performance between builds 1.1-1.2 and builds 2.1-2.2 due to the inability of the model to learn the extreme overheating dynamics observed in builds 1.1-1.2. This limitation is seen in Figure A.1 (a), where for temperatures over 1100 mV the model cannot make accurate predictions. Upon consultation with the researchers at the Mercury Centre, they assured us that the extreme temperature readings were likely to have been caused by measurements errors in the pyrometer. The pyrometer had been recalibrated

Table A.5: Model performance for the different builds performed.

| Build | Datapoints | MSE | $R^2$ |
|---|---|---|---|
| 1.1 | 13972 | 844±94 | 0.627±0.021 |
| 1.2 | 4776 | 621±143 | 0.675±0.029 |
| 2.1 | 3948 | 204±38 | 0.834±0.038 |
| 2.2 | 8780 | 232±57 | 0.807±0.035 |
| 3 | 6280 | 50±5 | 0.849±0.024 |
| 4 | 4820 | 54±5 | 0.840±0.028 |

before builds 2.1-2.2, where this overheating behaviour was not observed. However, we further investigate in A.4.2 if these overheating dynamics can be learned.

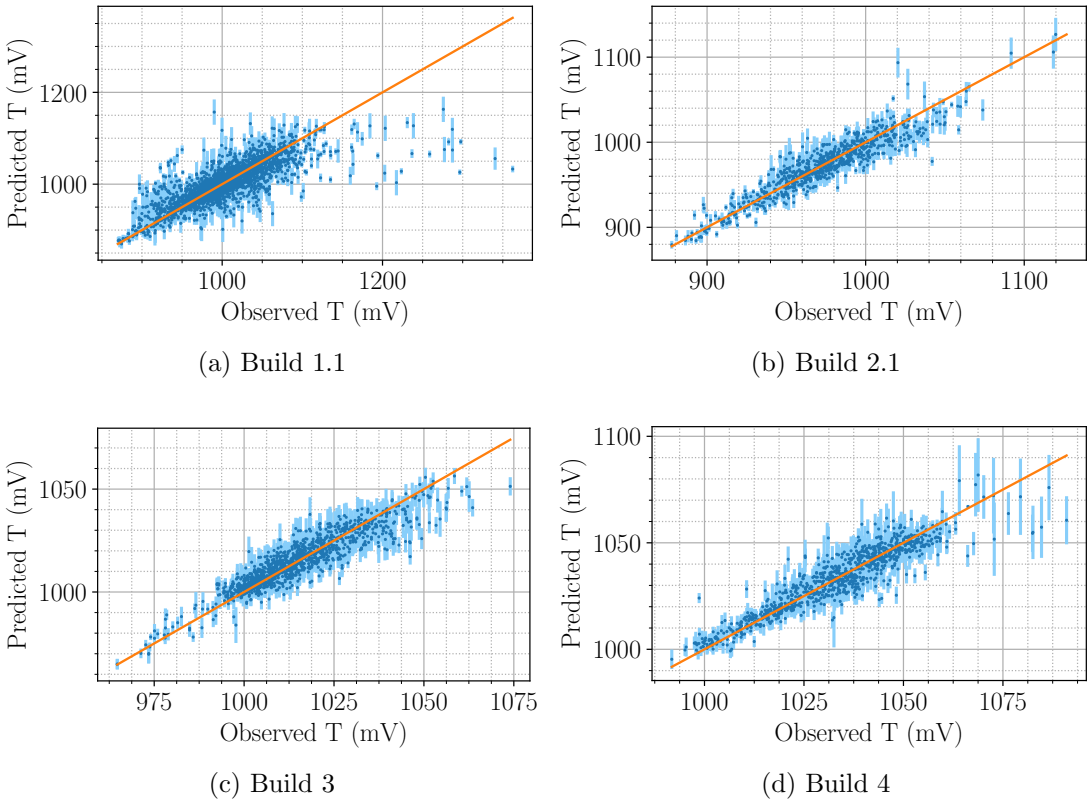
There is a fundamental difference between the data collected in builds 1.1-2.2 and builds 3-4: in the former, data was collected for as much of the state and input space as possible in order to build a general model of the process, whereas the data collected in builds 3-4 corresponds to the local dynamics of the process around its optimal operating temperature. As we would expect, a more accurate model of the local dynamics in builds 3-4 can be learned (lower MSE).

### A.4.2 Bootstrapping for Balancing the Dataset

The inability to model the overheating observed in build 1 could be attributed to the fact that the dataset used is not balanced, that is, most data corresponds to the system in its normal operational temperature –under 1050 mV–. Thus, the model will tend to better model this region of operational temperatures. However, if a more balanced dataset also fails to learn these overheating dynamics, this could indicate that in fact overheating occurs due to external disturbances or errors in the measurement device that cannot be accounted for.

A common and simple to implement technique to balance datasets is the use of bootstrapping-based oversampling and undersampling [78]. For both methods, the dataset is separated into the dominating and minority subsets, where the former contains all observed system states with no hotspots and the latter contains all system states where a hotspot occurred. A hotspot occurs if the temperature in any of the 16 cells comprising the system state is greater than 1050.

The oversampling method consists in resampling from the minority set until the desired minority:majority ratio is achieved. Some noise is generally added to




(a) Build 1.1

(b) Build 2.1

(c) Build 3

(d) Build 4

Figure A.1: Predicted vs Observed Temperature for different builds. The probabilistic model outputs the mean and standard deviation of a Gaussian distribution. In the graphs, the dark blue dots represent the output mean and the light blue bars represent $\pm$ 1 output standard deviation.

the resampled datapoints. We used random uniform noise of 1% as recommended by Batista et al. [25]. On the other hand, undersampling consists on removing datapoints from the majority set until the desired minority:majority ratio is achieved.

Bootstrapping-based undersampling, unlike oversampling, can lead to significant information loss as the datapoints discarded may contain unique information about the process. On the other hand, because the size of the dataset is reduced, the training time is also reduced. Naturally, the opposite is true for oversampling.

To evaluate if bootstrap-based undersampling or oversampling can be used to better model the overheating dynamics observed in build 1, 80% of the datapoints of build 1 were used for training and for testing. The training data was then split into the majority and minority sets. Thereafter, undersampling or oversampling was applied. Then, the RMSE over the testing data is computed for the minority set, the majority set, and both sets combined. Three different trials were performed, and the mean of the metrics accross the trials is reported in Table A.6.

The percentage of the majority set deleted or added is also reported.

Table A.6: Results of bootstrapping-based oversampling and undersampling.

| Undersampling | | | | | Oversampling | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | MSE | | | | | MSE | | |
| Min:Maj | % Del. | All | Min | Maj | Min:Maj | % Add. | All | Min | Maj |
| 0.05 | 4.60 | 731 | 3511 | 483 | 0.05 | 5 | 721 | 3505 | 472 |
| 0.15 | 13.67 | 766 | 3525 | 520 | 0.15 | 17 | 727 | 3433 | 484 |
| 0.25 | 22.73 | 770 | 3347 | 540 | 0.25 | 32 | 726 | 3511 | 476 |
| 0.40 | 36.13 | 772 | 3110 | 563 | 0.40 | 72 | 729 | 3381 | 491 |
| 0.50 | 45.38 | 831 | 3110 | 636 | 0.50 | 120 | 739 | 3328 | 507 |
| 0.60 | 54.42 | 870 | 2958 | 683 | 0.60 | 213 | 759 | 3213 | 539 |
| 0.77 | 60.49 | 945 | 2780 | 781 | 0.77 | 348 | 755 | 3111 | 544 |

Both techniques did reduce the MSE of the minority set significantly, however, not to such an extent where it could be considered that the dynamics could be accurately modelled. However, both methods did significantly increase the MSE of the majority set, which is very undesirable. For undersampling, this was particularly evident, which could be attributed to a significant loss of information, as discussed earlier. On the other hand, for the oversampling method the training time was more than doubled in some instances.

In summary, both undersampling and undersampling did not improve the model accuracy for the overheating dynamics enough to justify the significant reduction in model accuracy for the regular process dynamics. This could indicate that the overheating behaviour observed is caused by external disturbances or errors in the measurement device that cannot be accounted for in the model.

### A.4.3 Effect of the Scan Pattern on the Process Dynamics

Several studies have shown that the choice of scanning pattern in SLM processes can significantly influence the temperature dynamics of the process [79]. For the builds performed, the corner of each individual cube from which the layer starts to be built is changed at every layer, since this has been shown to reduce the formation of hotspots at the corners.

Thus, if the scan pattern is a very substantial process signature affecting the process dynamics, then fitting an individual dynamics model for each of the 4 possible starting corners should enhance the model accuracy. To test this hypothesis,

Table A.7: Metrics for each starting corner.

| Corner | MSE | $R^2$ |
|---|---|---|
| All | 274±11 | 0.794±0.008 |
| Top left | 321±26 | 0.782±0.014 |
| Top right | 322±71 | 0.770±0.044 |
| Bottom left | 230±14 | 0.775±0.009 |
| Bottom right | 226±16 | 0.757±0.018 |

Table A.8: Metrics for each cube location.

| Location | MSE | $R^2$ |
|---|---|---|
| All | 274±11 | 0.794±0.008 |
| Top left | 177±12 | 0.871±0.011 |
| Top right | 192±29 | 0.841±0.025 |
| Bottom left | 143±17 | 0.870±0.019 |
| Bottom right | 230±20 | 0.856±0.015 |

the dataset for build 2 was split into 4 subsets according to corner from which each layer started to be built. A model was fit for each subset, and the mean MSE and $R^2$ of the testing data for 5-fold CV is reported in Table A.7.

Compared to not accounting for the scan pattern in the model, the MSE significantly decreased for the top corners, but increased for the bottom corners. On the other hand, the $R^2$ metric worsened for all corners. Because the metrics computed did not improve on all instances, it is likely that the corner in which the layer starts printing is not a dominant signature affecting the process dynamics.

### A.4.4 Cube Location and the Process Dynamics

The cubes built in build 1 and build 2 were printed in groups of 4 forming a single object, as explained in §5.1. The relative location of each of the 4 cubes may affect their dynamics, due to factors such as different cooling rates. To test this hypothesis, the data collected in build 2 was divided into two subsets, one for each possible cube position with respect to the centre of the object: top left, top right, bottom left and bottom right. A model was fit for each subset, and the mean MSE and $R^2$ of the testing data for 5-fold CV is reported in Table A.8.

Both the MSE and $R^2$ improve in all instances. This could indicate that each of the cubes has substantial different dynamics due to its relative location within the object. Furthermore, it is likely that the assumption that the dynamics of the 4 cubes printed as a single object are similar those of a cube printed as an object in itself is not a valid assumption. This is an important insight, because the simulations of the process dynamics do make this assumption.

### A.4.5 Number of Cells and Model Performance

As described in §4.2, the layer surface was divided into a number of cells, of which the mean temperature was taken as the system state. The impact of the number of cells on model performance is summarised in Table A.9.

Two important insights can be gained: firstly, that the spatial resolution of the layer temperature is indeed key for building an accurate thermal model of the process. The model trained for 4 cells is substantially worse than the other two. This reinforces our choice of system state over the sole use of the mean and the variance of the temperature, as previous studies considered. Our approach is superior not only because hotspots can be directly detected, but also because a fundamentally more accurate model of the system dynamics can be built.

On the other hand, employing too many cells can also hindering the performance of the model, albeit not as significantly. However, this may be not because of the system state representation but rather because of the higher dimensionality of the problem, which may require a different hyperparameter configuration.

Table A.9: Model performance for different number of cells.

| Number of cells | MSE | $R^2$ |
|---|---|---|
| 4 | 389±12 | 0.673±0.011 |
| 16 | 274±11 | 0.794±0.008 |
| 64 | 293±7 | 0.801±0.005 |

# Appendix B

# Simulation Experiments

The choice of the model predictive control parameters is of substantial importance for the proper functioning of the algorithm. The planning horizon is the most critical parameters: the predicted trajectories will be dominated by bias for low planning horizons, and by variance for high planning horizons. The rest of parameters pertain the CEM optimisation algorithm: number of particles to propagate, the popsize or number of actions that are sampled and the maximum number of iterations of the algorithm.

To compare the performance of different parameter configurations, the main metric used is the return obtained by the simulation environment. Other metrics which may be provided are the MSE, the % of layers where the mean temperature was $\pm 20$ mV of the target temperature, and the % of layers where the temperature was greater than 70 mV over that target temperature (considered a hotspot).

## B.1 No Control (Fixed Parameters)

The metrics obtained using MPC are compared to the metrics obtained by keeping the process parameters constant, where the parameters tested are those found through the experimental DoE (1.125 m/s and 110W) and those which lead to return as determined through grid search of the parameter space (1.3m/s and 117W). These metrics are provided in Table B.1.

Table B.1: Metrics for no control (fix parameters).

| Scan speed | Laser power | Return | MSE | % ±20 | % +70 |
|---|---|---|---|---|---|
| 1.125 m/s | 110W | 4.6±0.6 | 20.7±1.5 | 61.2±4.0 | 0.8±0.5 |
| 1.300 m/s | 117W | 6.7±0.6 | 16.5±0.8 | 70.9±1.4 | 0.1±0.0 |

## B.2 MPC with a Fully Known Model

To asses the best possible performance which can be achieved for the simulated environment considered, MPC is performed where the model used for optimising the control inputs is the underlying transition model of the simulated environment.

### B.2.1 Deterministic Model

This section considers the MPC for a deterministic dynamic model of the system dynamics. It was observed that computational time constraints (number of iterations and popsize of the CEM algorithm) led to an "optimal" plan horizon of just 5 time steps. The comparison of different plan horizons (using a fully know model) are given in Table B.2.

Table B.2: Performance comparison of different plan horizons (deterministic model).

| Plan horizon | Return | MSE | % ±20 | % +70 |
|---|---|---|---|---|
| 1 | 28.3±2.0 | **10.9±1.0** | **89.0±1.2** | 0.0±0.0 |
| 5 | **30.9±0.9** | 11.4±0.4 | 87.6±0.8 | 0.1±0.0 |
| 10 | 28.6±1.0 | 11.4±0.3 | 87.9±0.5 | 0.0±0.0 |
| 15 | 25.6±1.1 | 11.7±0.4 | 86.8±1.0 | 0.0±0.0 |
| 25 | 24.7±2.0 | 12.1±0.6 | 85.1±1.5 | 0.0±0.0 |
| 50 | 20.9±0.9 | 12.7±0.6 | 83.2±1.4 | 0.0±0.0 |
| 75 | 21.5±0.7 | 12.9±0.9 | 84.2±1.4 | 0.0±0.0 |

### B.2.2 Stochastic Model

This section considers the MPC for a stochastic model of the system dynamics. The results are provided in Table B.3. As expected, the optimal plan horizon is lower than for a deterministic model, as it is not possible to accurately predict far into the future due to the high stochasticity of the dynamics model.

Table B.3: Performance comparison of different plan horizons (stochastic model).

| Plan horizon | Return | MSE | % ±20 | % +70 |
|---|---|---|---|---|
| 1 | **27.3±2.2** | **11.0±0.5** | **87.9±1.3** | **0.0±0.1** |
| 3 | 25.1±2.5 | 12.0±0.5 | 85.5±1.6 | 0.1±0.1 |
| 5 | 19.9±1.7 | 12.4±0.4 | 84.0±1.0 | 0.1±0.1 |
| 10 | 15.5±0.9 | 13.3±0.5 | 80.9±1.2 | 0.1±0.1 |

To determine if the low value of the planning horizon was caused by the the choice of CEM parameters, grid search was conducted over the following search space: $T = \{1, 3, 5\}$, $N_{CEM} = \{150, 500, 2000\}$, $N_{\text{elites}} = \{5\%, 15\%\}$, $B = \{1, 5, 20\}$. Due to the large number of combinations, and the fact that lower plan horizons were used, the task horizon (number timesteps) was reduced to 50. A selection of the results is provided in Table B.4, with the best performing parameter combination being $T = 3$, $N_{CEM} = 2000$, $N_{\text{elites}}$=5 and $B = 20$. Generally, it was observed that when $N_{CEM}$ and $B$ are small –fewer computational time due to fewer candidate solutions– low plan horizon works best, as this reduces the dimensionality of the optimisation problem. It is thus very likely that if more computational resources were available, better solutions could be found by using higher $T$, $N_{CEM}$ and $B$.

Table B.4: Performance comparison of different CEM parameter combinations.

| $T$ | $N_{CEM}$ | $N_{\text{elites}}$ (%) | $B$ | Return |
|---|---|---|---|---|
| 1 | 150 | 5 | 1 | 8.2±1.1 |
| 1 | 2000 | 5 | 5 | 9.0±1.2 |
| 3 | 500 | 5 | 20 | 10.3±1.2 |
| 3 | 2000 | 5 | 5 | 10.5±0.8 |
| 3 | 2000 | 5 | 20 | **10.9±0.8** |
| 5 | 150 | 5 | 1 | 7.2±0.8 |
| 5 | 2000 | 5 | 5 | 9.9±0.6 |

## B.3 MPC with a Learned Model

This test assesses the performance of the algorithm on the standard epistemic RL problem, where the system dynamics are assumed to be unknown. The model can be learned at the end of each trial (eac 50 leayhers) or at a higher rate.

### B.3.1 Model Learned After Each Trial

The key conclusions obtained from this tests are that a plan horizon of 1 performs substantially better than a plan horizon of 3, and that the use of a probabilistic model leads to substantially better performance compared to that of a deterministic model.

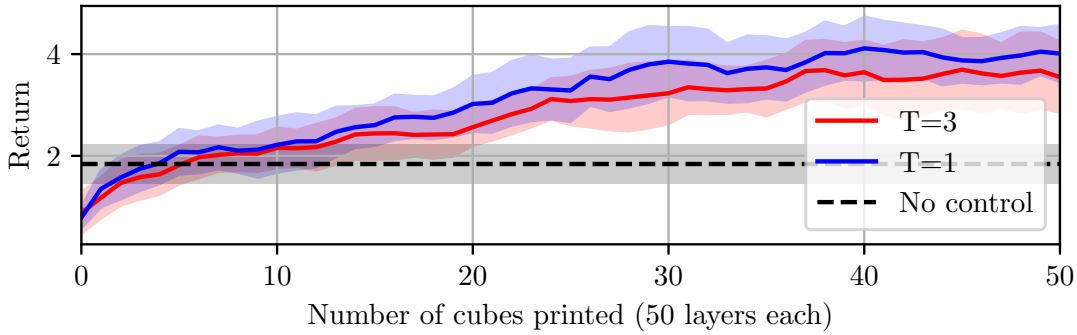
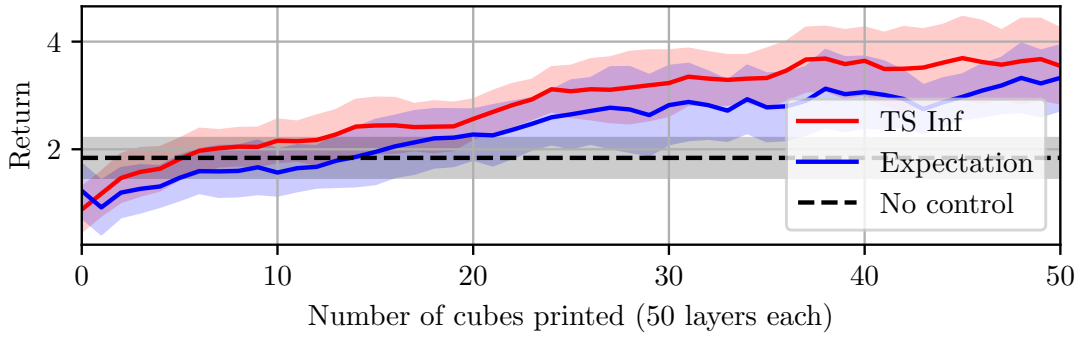
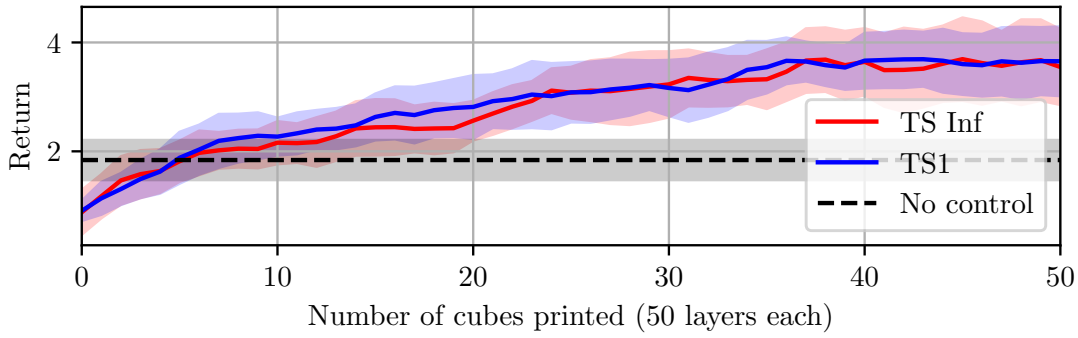


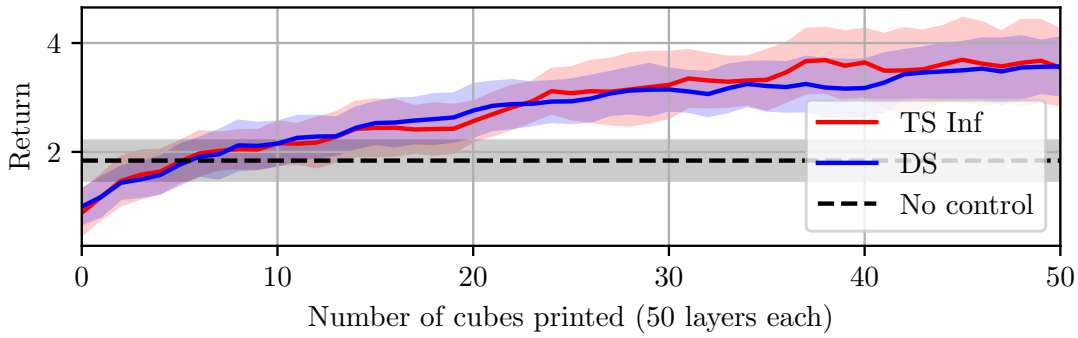

Figure B.1: Comparison of plan horizon.



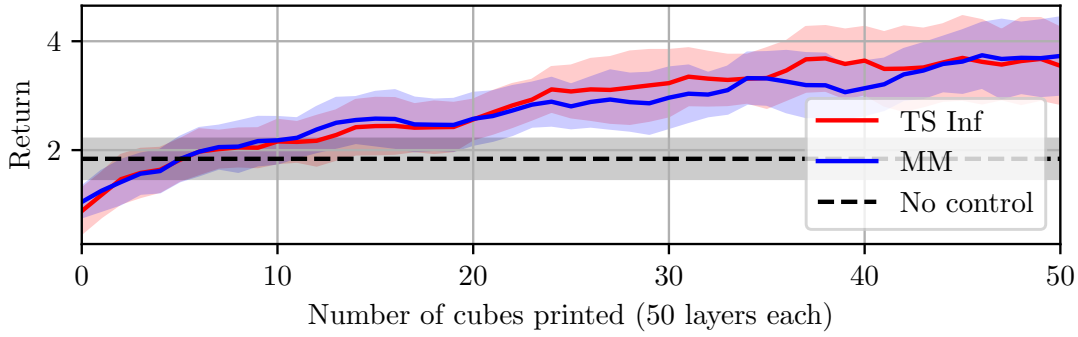

(a) TS∞ vs Deterministic (expectation) propagation.




(b) TS∞ vs TS1 propagation.




(c) TS∞ vs Distributed Sampling (DS) propagation.

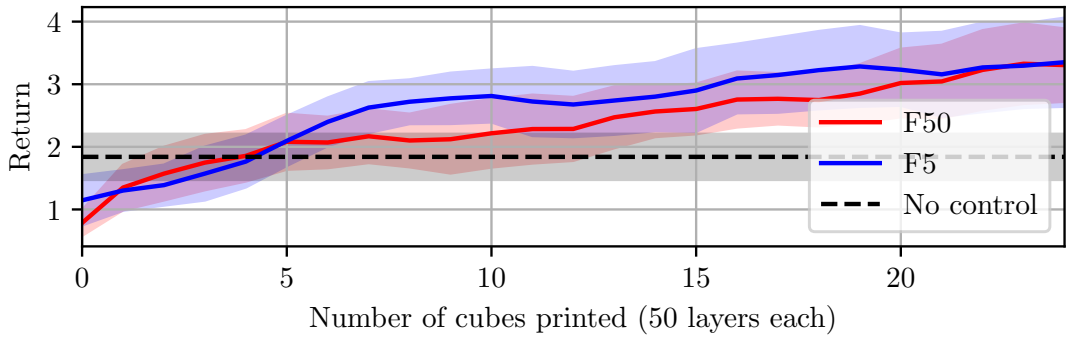



(d) TS∞ vs Momentum Matching (MM) propagation.

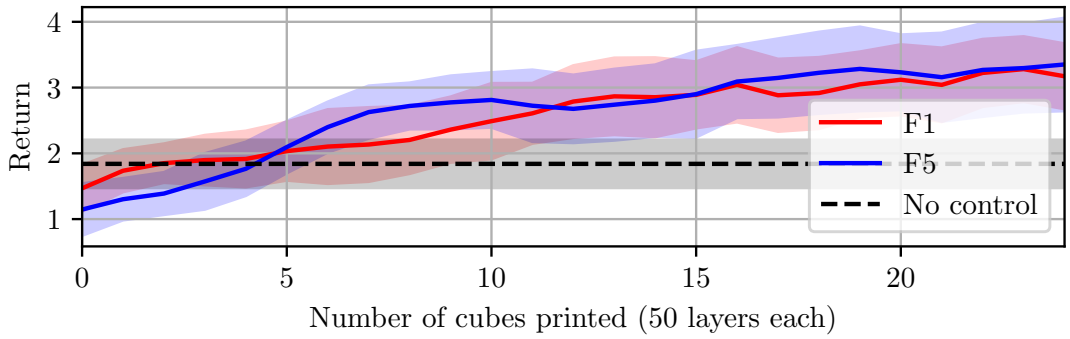
Figure B.2: Comparison of state propagation methods.

### B.3.2 Model Learned Multiple Times Each Trial

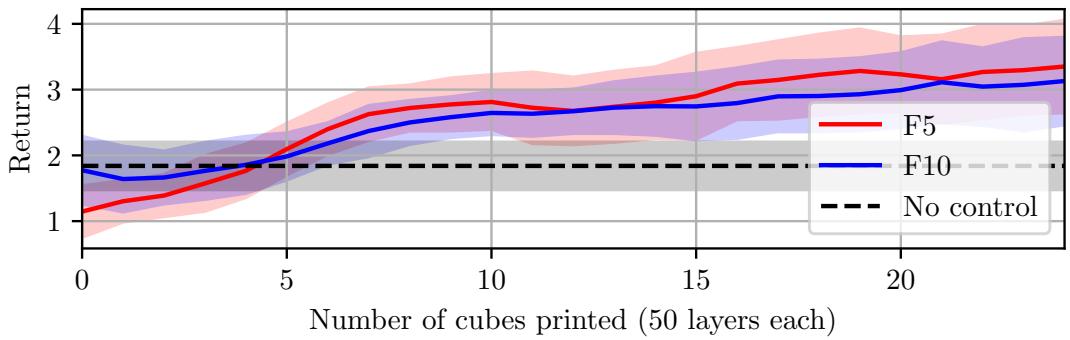
To improve sample-efficiency, the PETS algorithm was slightly altered, so that the model is trained not at the end of each trial, but at a higher rate (multiple times per trial). A high training frequency or number of training epochs causes the model to overfit initially, while a low training frequency or number of training epochs may cause to model to learn slowly, also degrading performance.




(a) At the end of each cube (50 layers) vs every 5 layers.




(b) Every layer vs every 5 layers.




(c) Every 5 layers vs every 10 layers.

Figure B.3: Comparison of model training frequency.

## B.4 MPC for a Scenario Alike the Real System

The previous tests have considered that cubes are built one by one. However, this is not the case for the real-world experiments conducted, where multiple samples are built simultaneously. Furthermore, only a single episode is considered as only one build is performed, albeit with a larger time horizon of $H = 160$. Different parameters are tested in simulation, with the results shown in Table B.5. The differences between parameter configurations are significant but relatively low.

Table B.5: Performance of different parameters (alike real experiment).

| $T$ | $N_{CEM}$ | $f_{\text{train}}$ | $N_{\text{b}}$ | $B$ | Return |
|---|---|---|---|---|---|
| 1 | 500 | 5 | 5 | 20 | 9.1±1.2 |
| 1 | 2000 | 5 | 5 | 20 | 9.2±1.0 |
| 1 | 500 | 1 | 5 | 20 | **9.9±1.2** |
| 1 | 500 | 10 | 5 | 20 | 8.5±1.0 |
| 1 | 500 | 5 | 10 | 20 | 9.2±1.0 |
| 1 | 500 | 5 | 10 | 40 | 9.0±0.9 |

## B.5 Proportional-Integral Control

The only form of closed-loop control previously demonstrated in literature is Proportional-Integral (PI) control, where the laser power $P$ was modulated according to a feedback measure of the melt-pool size, with the aim of maintaining the melt-pool size at a constant reference target. This approach was adapted to our case study by changing the feedback signal to the mean layer temperature $\bar{x}$, and the reference target to the optimal layer temperature $r_T = 980$ mV.

Thus, the laser power is computed according to the control law :

$$P = K_p \cdot (r_T - \bar{x}) + Ki \cdot I_k \tag{B.1}$$

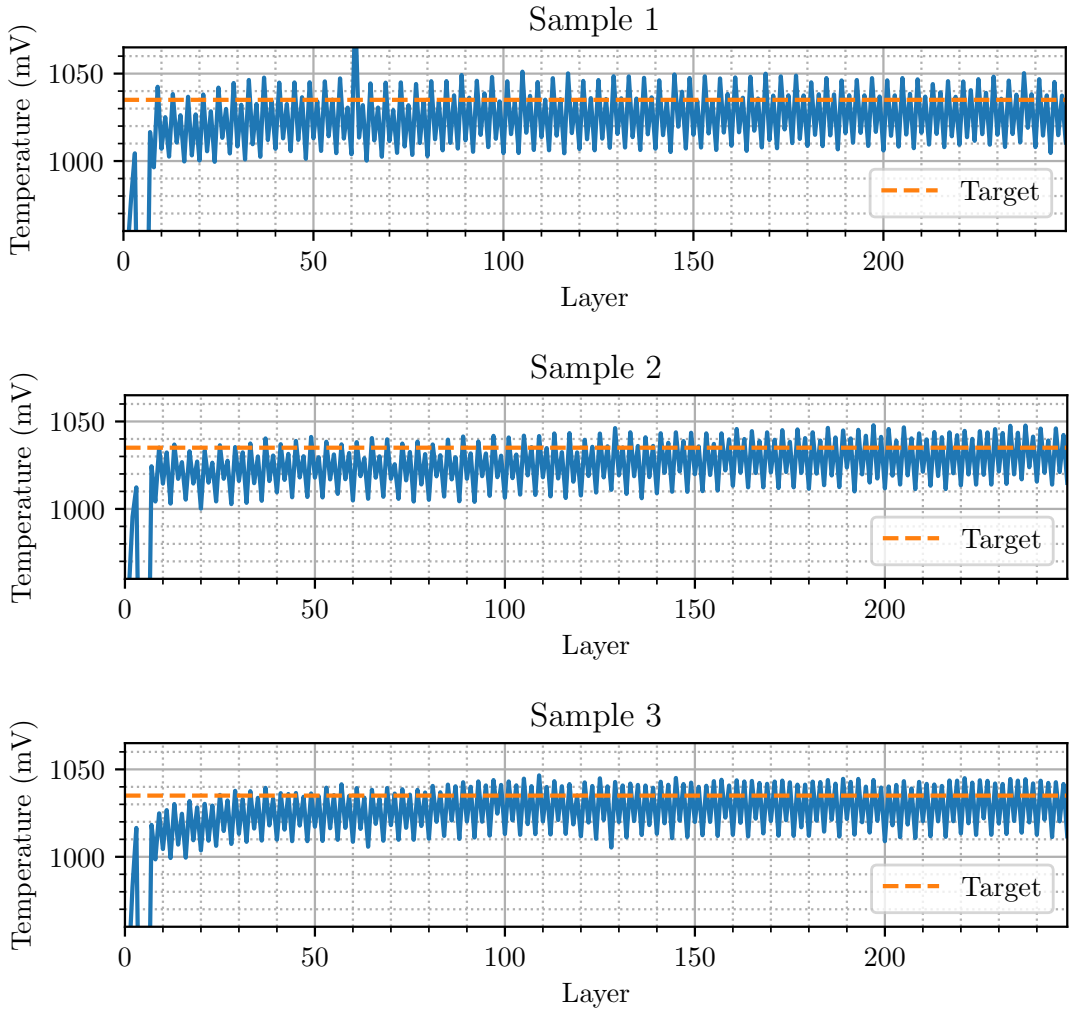
where $I_k = I_{k-1} + (r_T - \bar{x})$ and $I_0 = 0$. The gains $K_p = 1.2$ and $K_i = 0.5$, as well as the constant scan speed $v = 1.4$ m/s, were manually tuned.

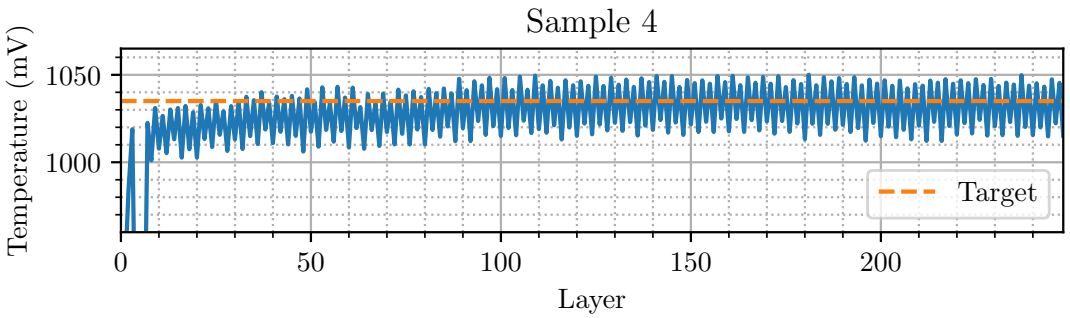
# Appendix C

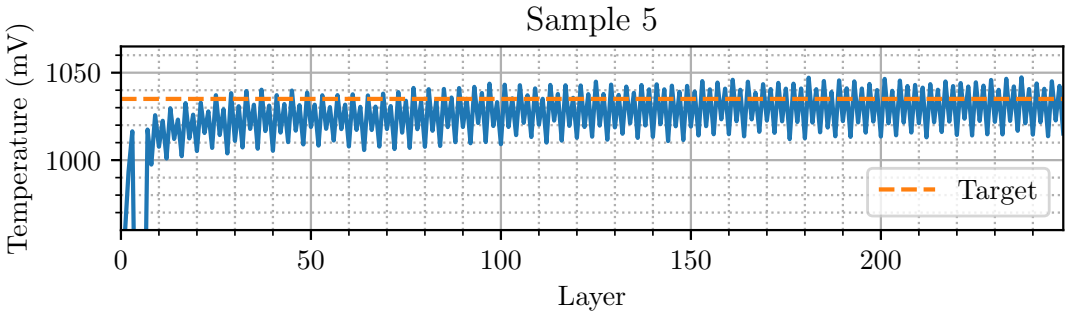
# Real-world Experiments

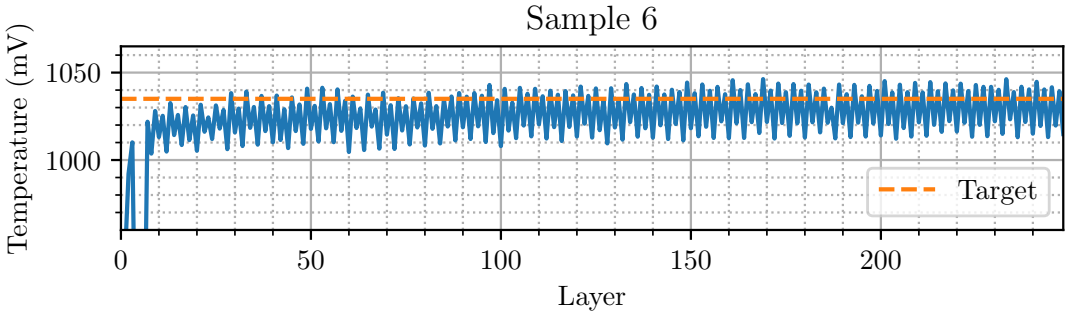
## C.1 CM247 Experiments

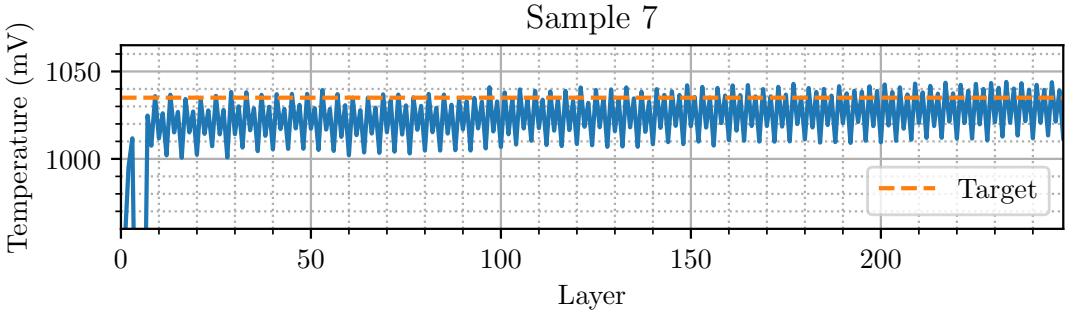
### C.1.1 No Control

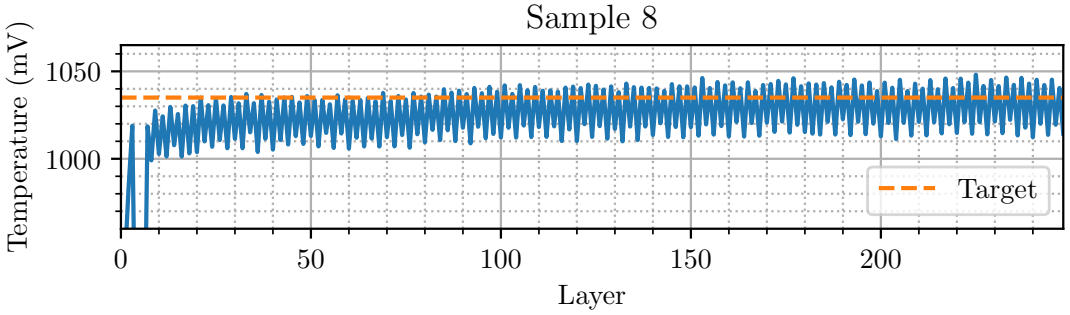


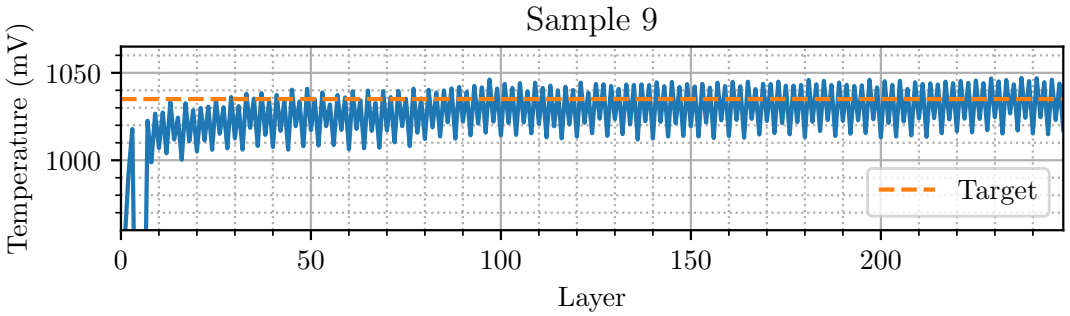

Sample 4
Temperature (mV)
1050
1000
0
50
100
150
200
Layer
Target

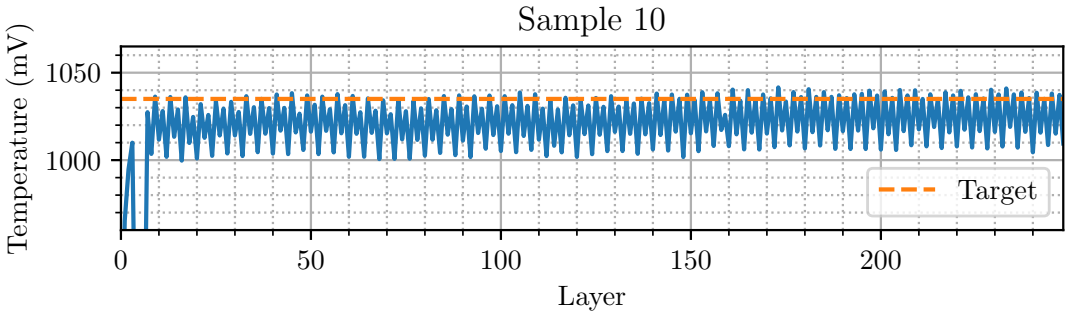

Sample 5
Temperature (mV)
1050
1000
0
50
100
150
200
Layer
Target


Sample 6
Temperature (mV)
1050
1000
0
50
100
150
200
Layer
Target

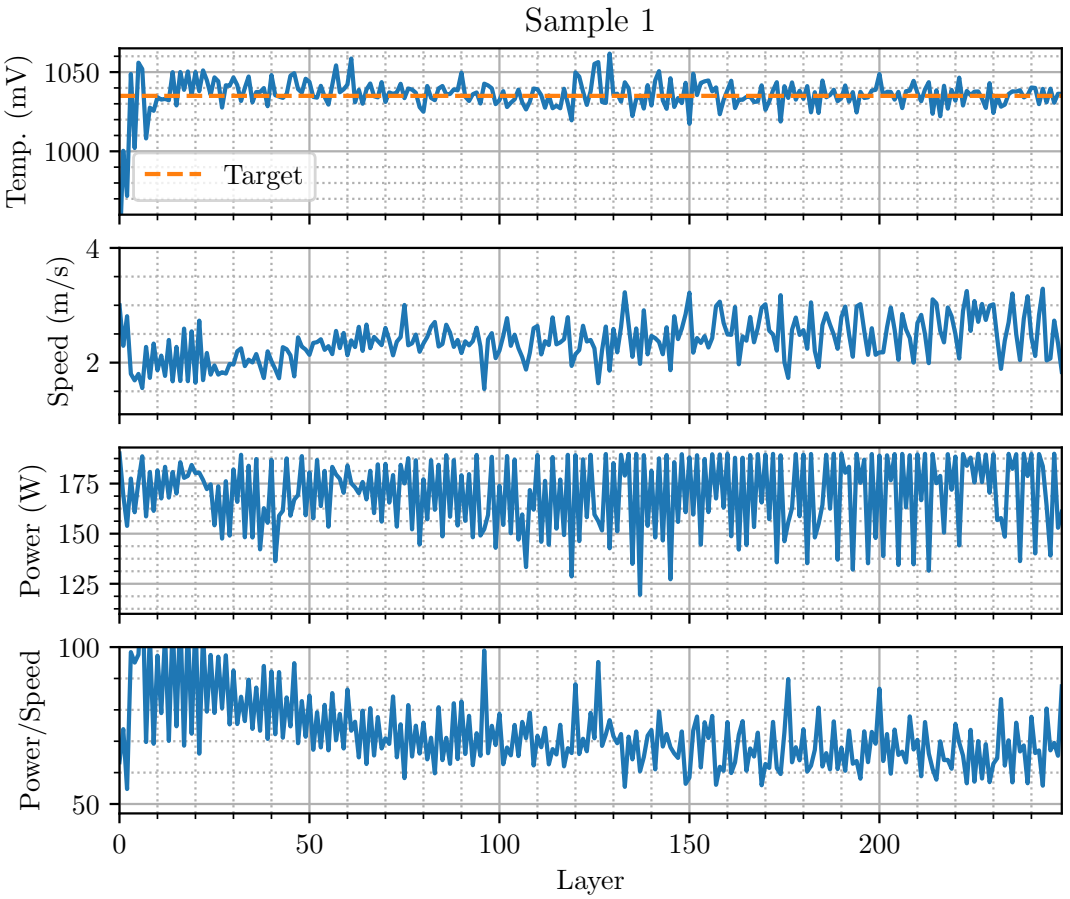

Sample 7
Temperature (mV)
1050
1000
0
50
100
150
200
Layer
Target

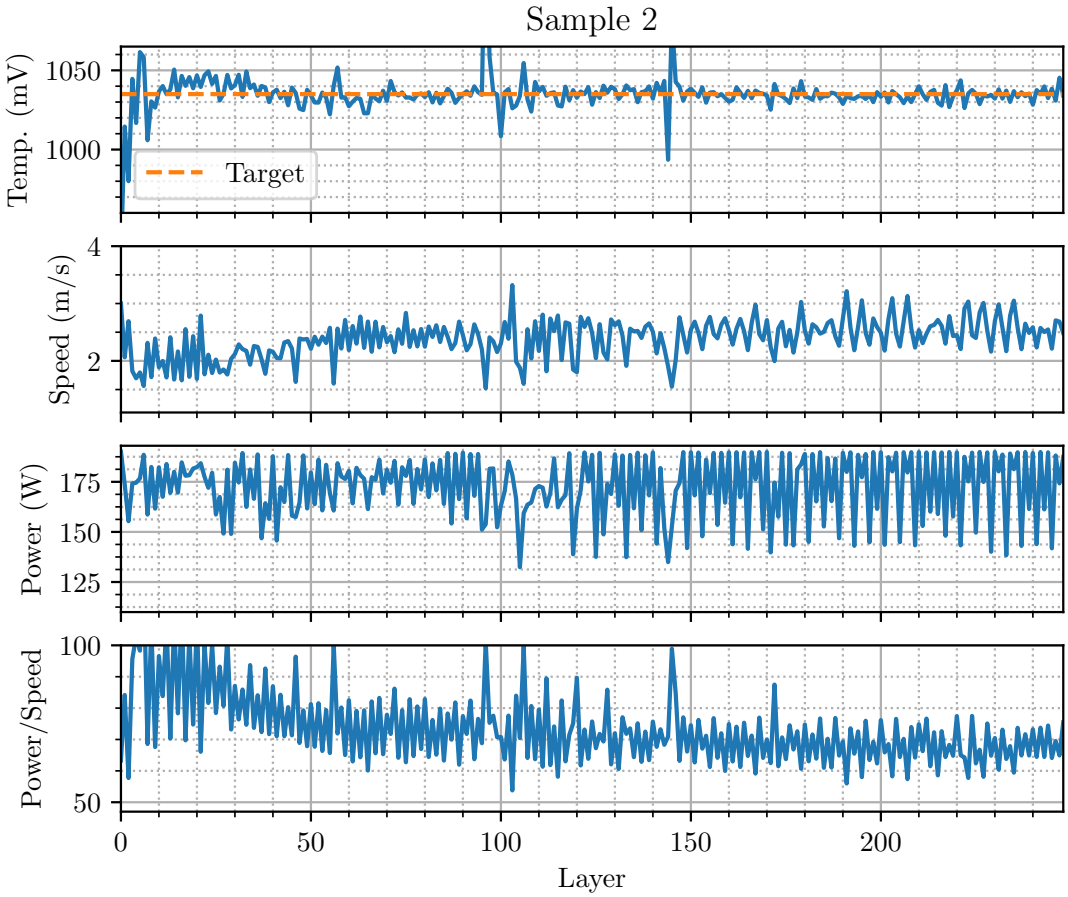

Sample 8
Temperature (mV)
1050
1000
0
50
100
150
200
Layer
Target

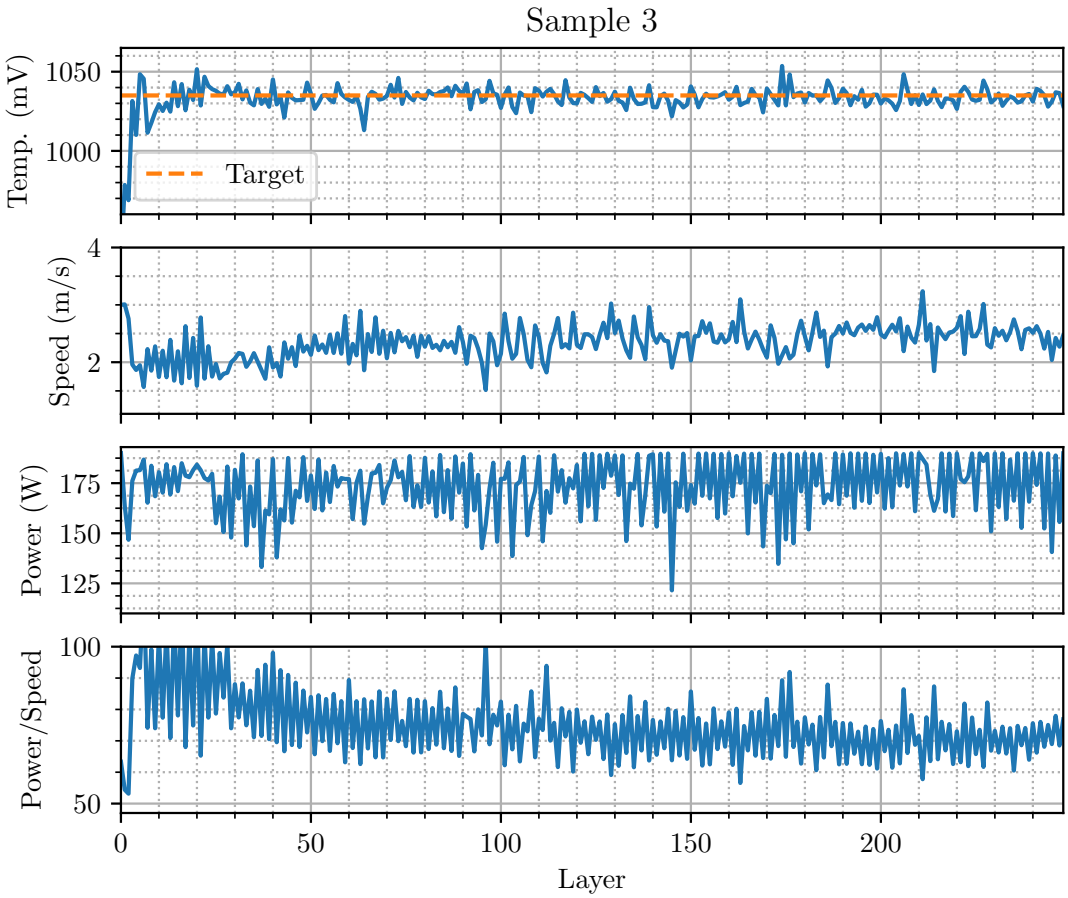


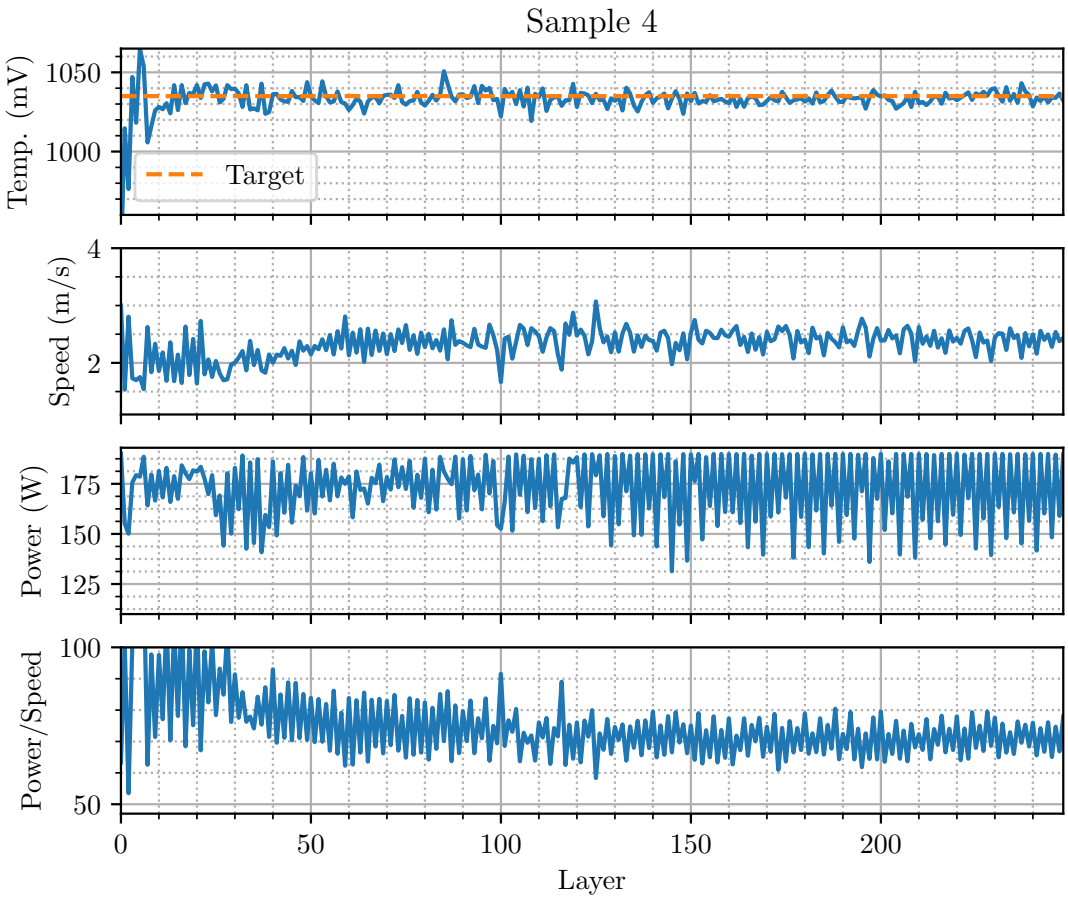



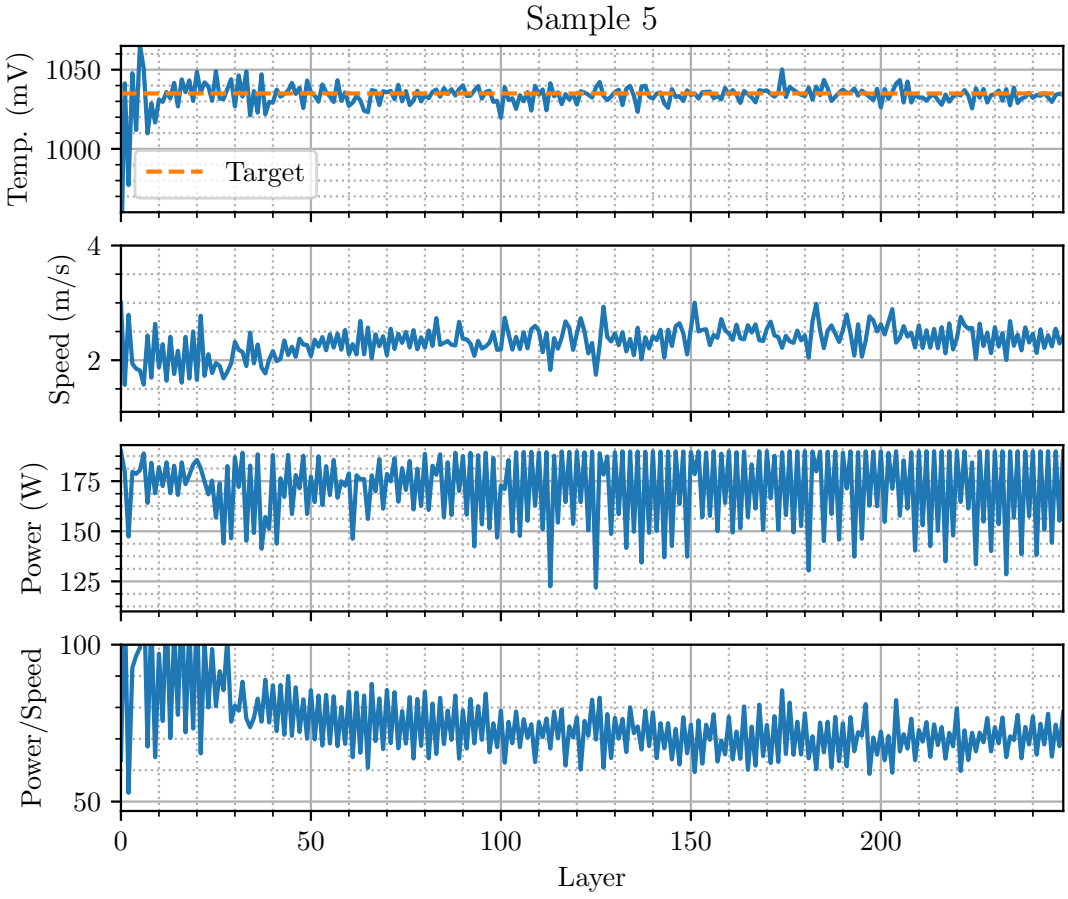

### C.1.2 With Control

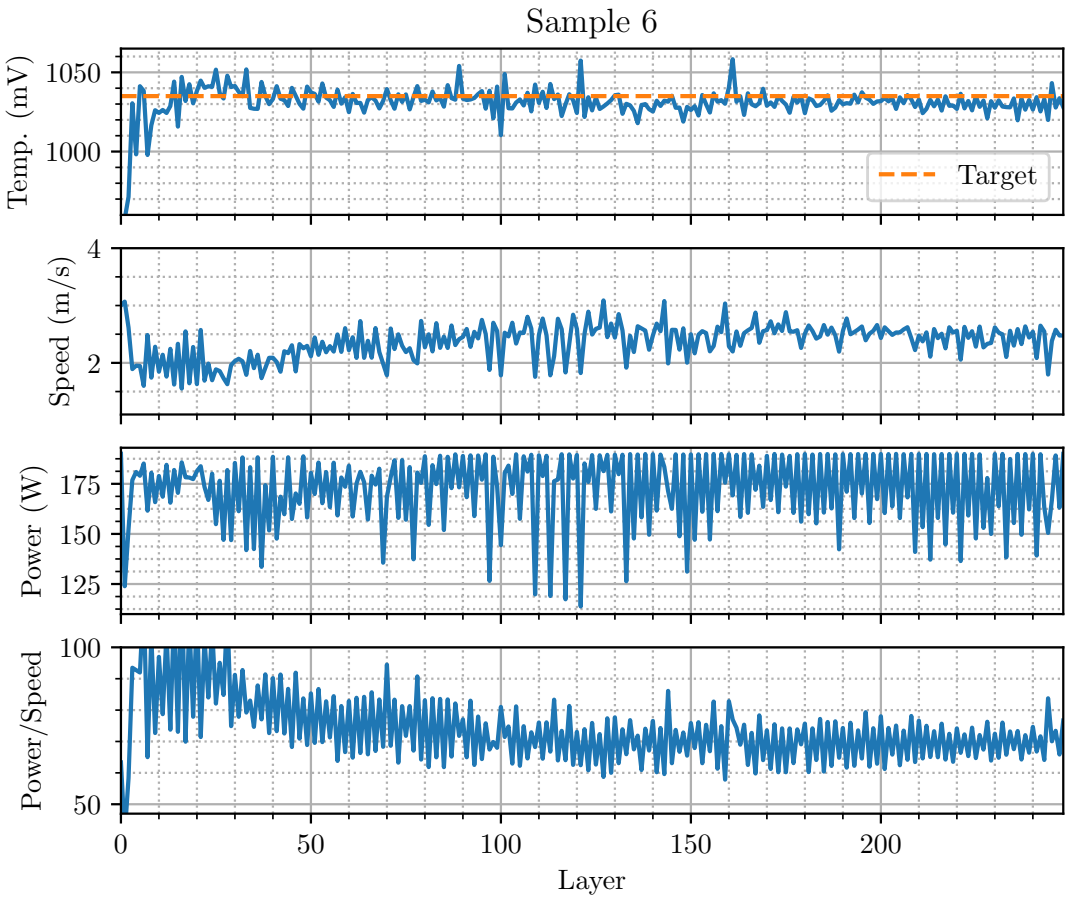


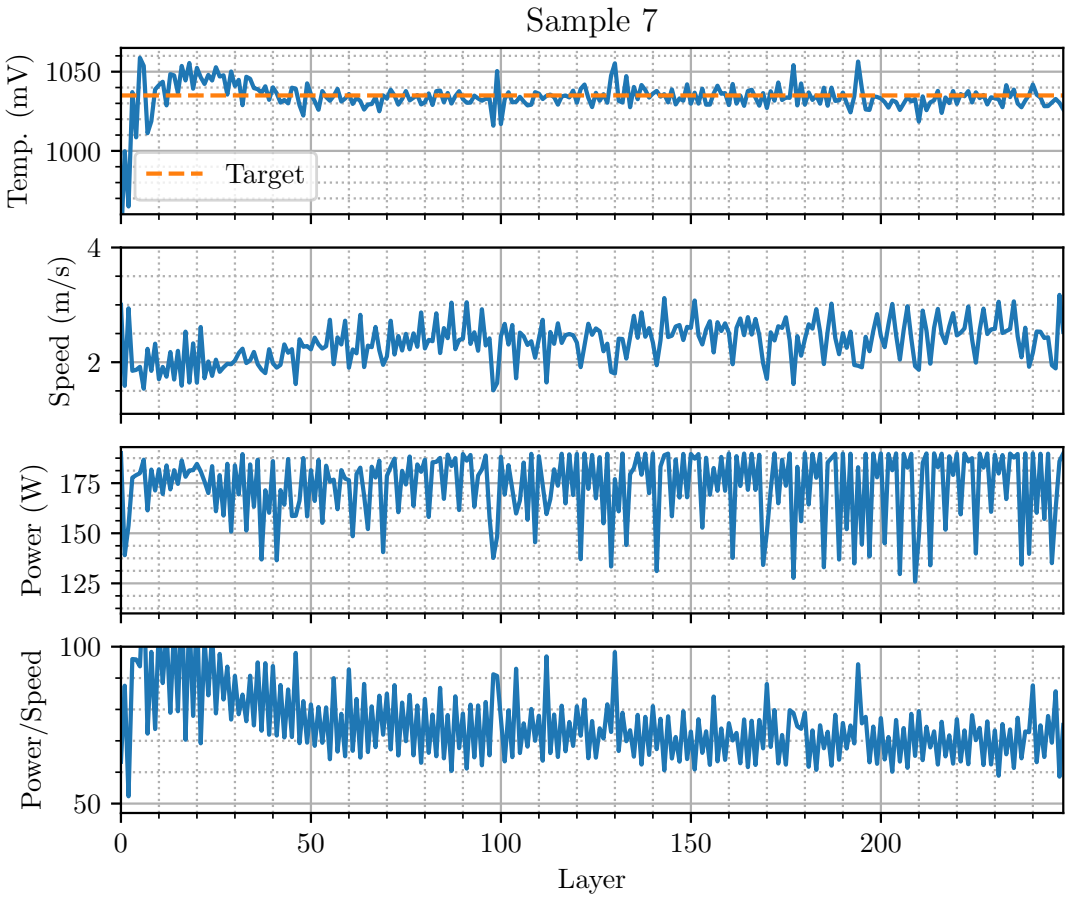

Sample 2
Temp. (mV)
1050
1000
Target
Speed (m/s)
4
2
Power (W)
175
150
125
Power/Speed
100
50
0
50
100
150
200
Layer

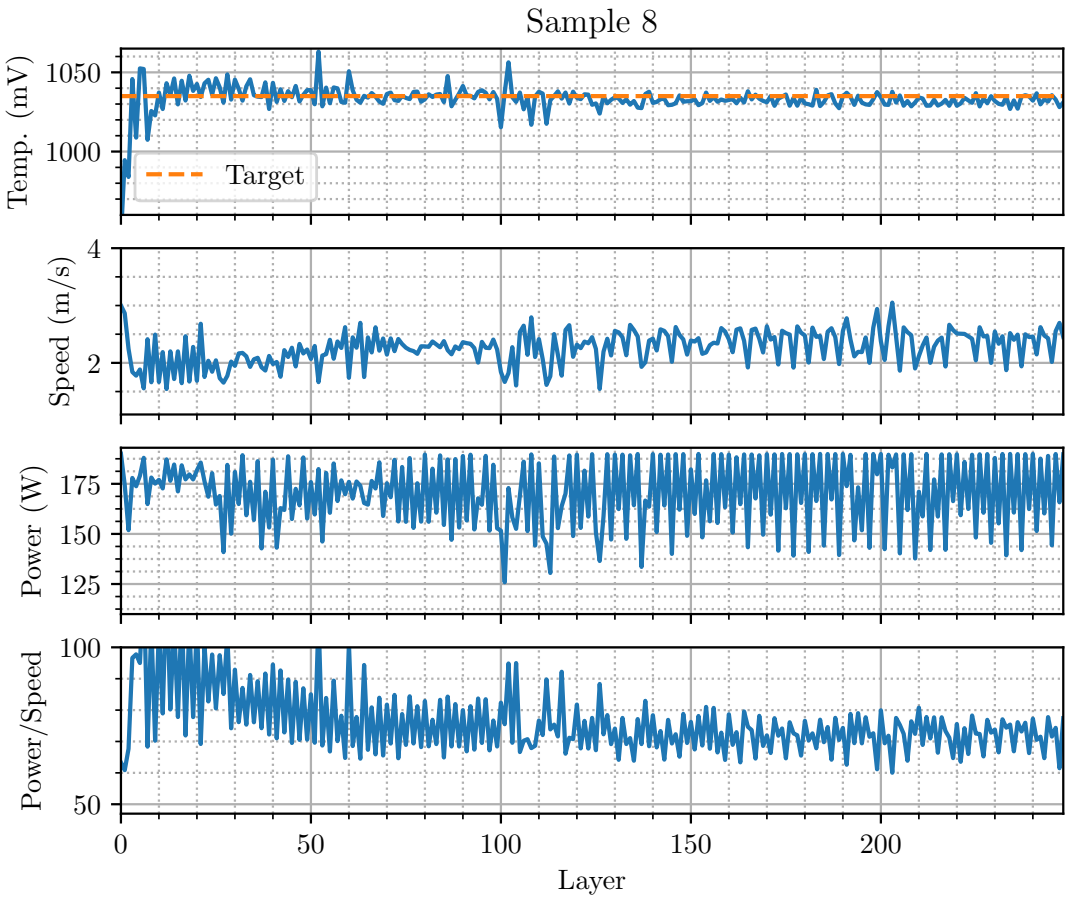

Sample 3
Temp. (mV)
1050
1000
Target
Speed (m/s)
4
2
Power (W)
175
150
125
Power/Speed
100
50
0
50
100
150
200
Layer

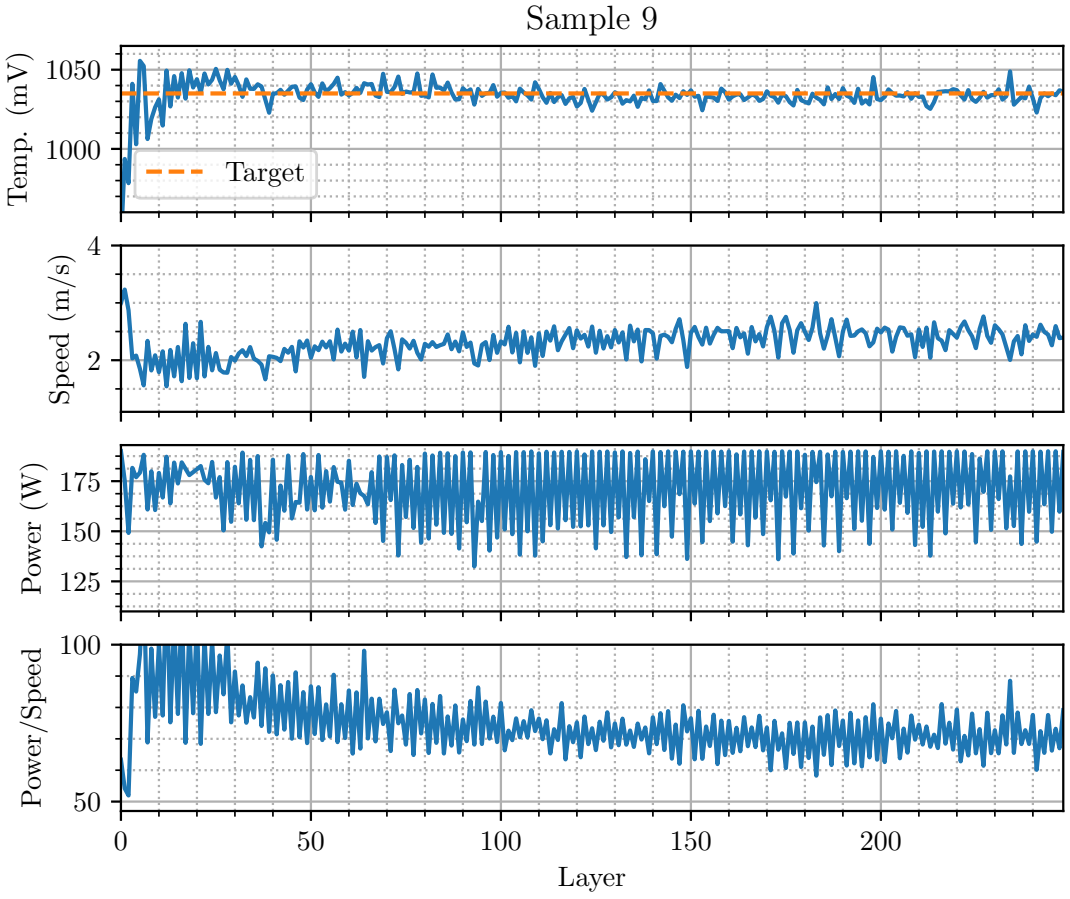

Sample 4
Temp. (mV)
1050
1000
Target
Speed (m/s)
4
2
Power (W)
175
150
125
Power/Speed
100
50
0
50
100
150
200
Layer

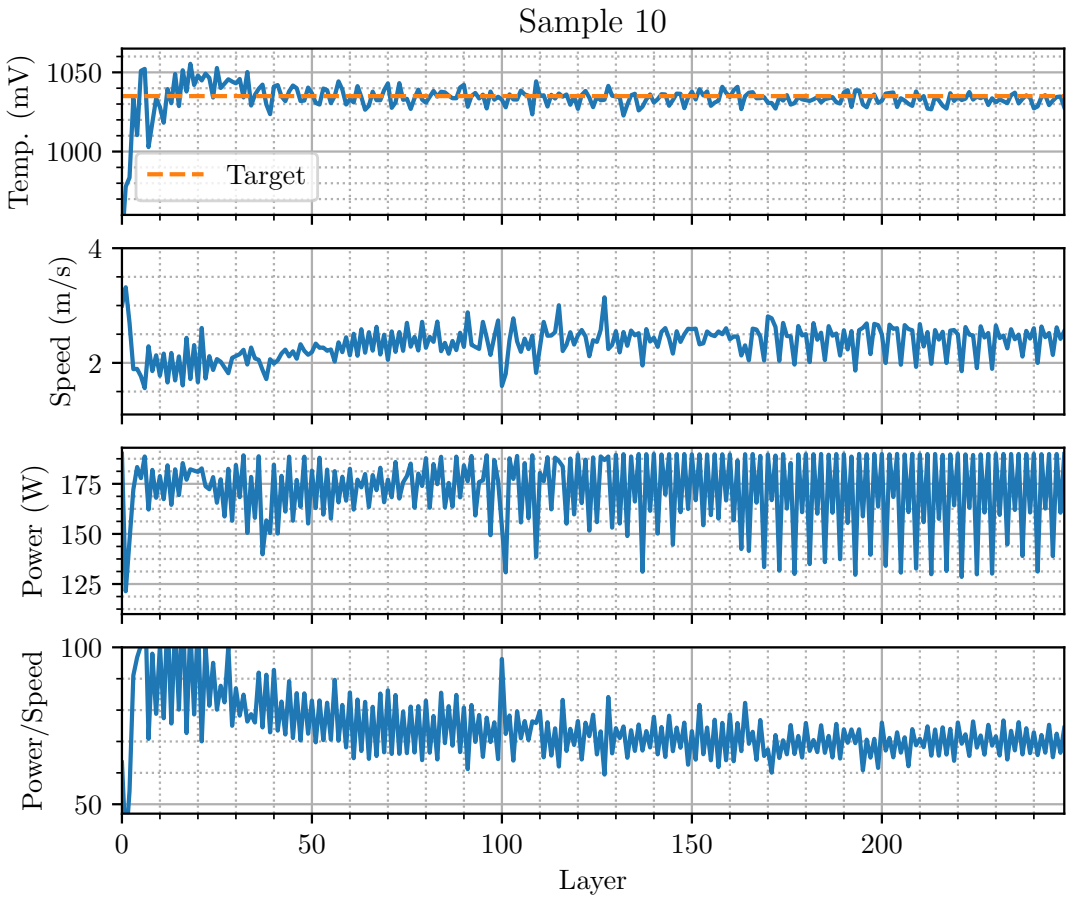

Sample 5
Temp. (mV)
1050
1000
Target
Speed (m/s)
4
2
Power (W)
175
150
125
Power/Speed
100
50
0
50
100
150
200
Layer


Sample 6
Temp. (mV)
1050
1000
Target
Speed (m/s)
4
2
Power (W)
175
150
125
Power/Speed
100
50
0
50
100
150
200
Layer


Sample 7
Temp. (mV)
1050
1000
Target
Speed (m/s)
4
2
Power (W)
175
150
125
Power/Speed
100
50
0
50
100
150
200
Layer

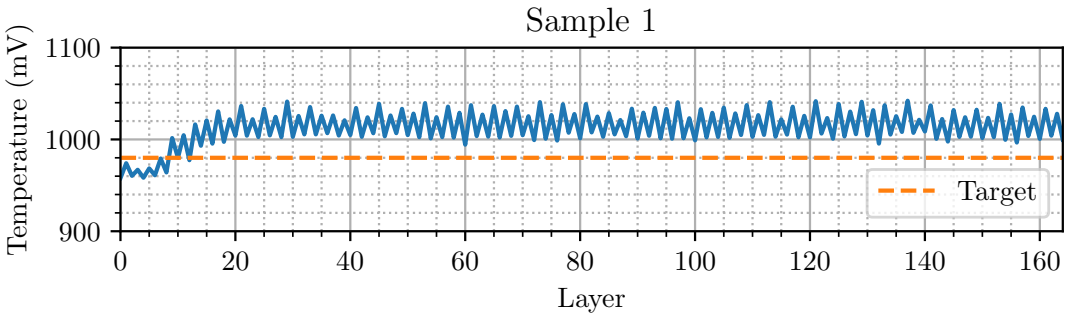

Sample 8
Temp. (mV)
1050
1000
Target
Speed (m/s)
4
2
Power (W)
175
150
125
Power/Speed
100
50
0
50
100
150
200
Layer

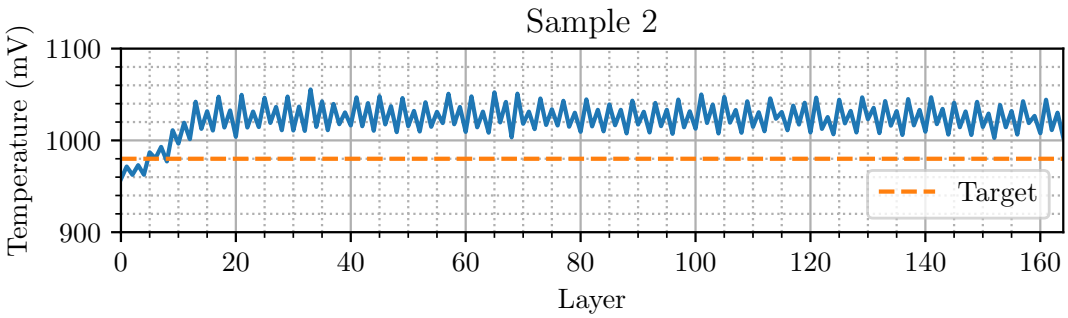

Sample 9
Temp. (mV)
1050
1000
Target
Speed (m/s)
4
2
Power (W)
175
150
125
Power/Speed
100
50
0
50
100
150
200
Layer

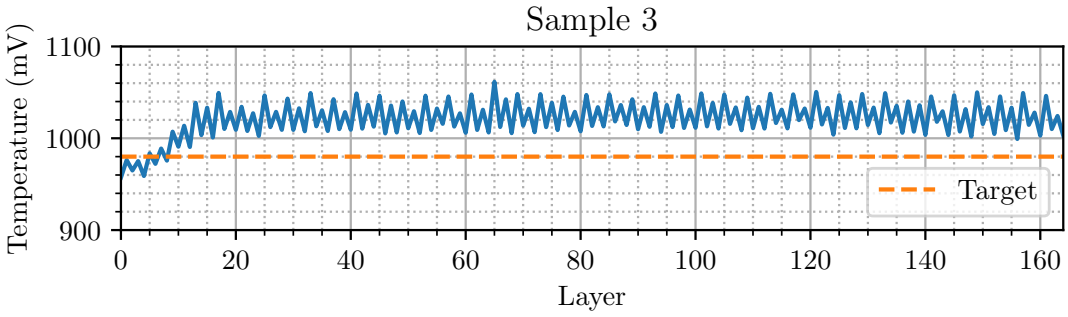


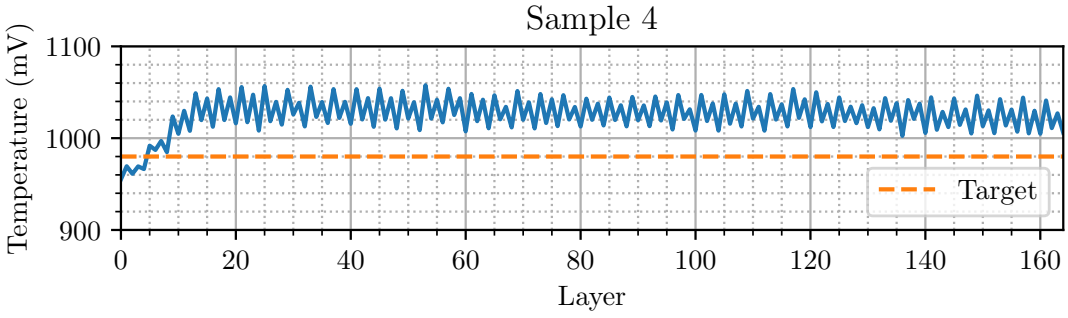

## C.2 Haynes 282 Experiments

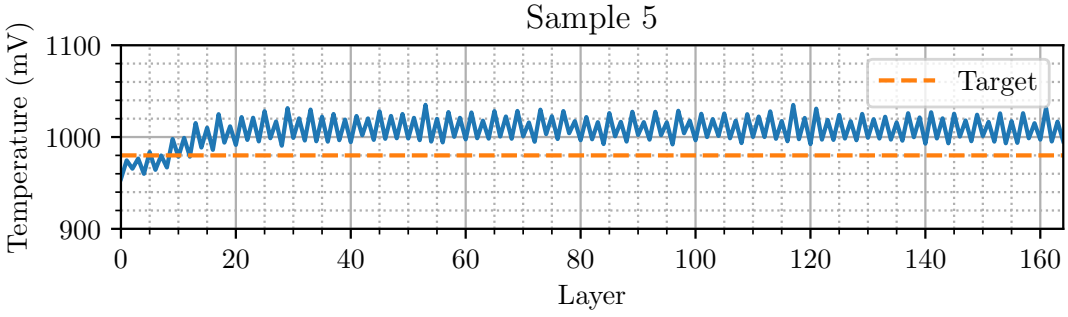
### C.2.1 No Control

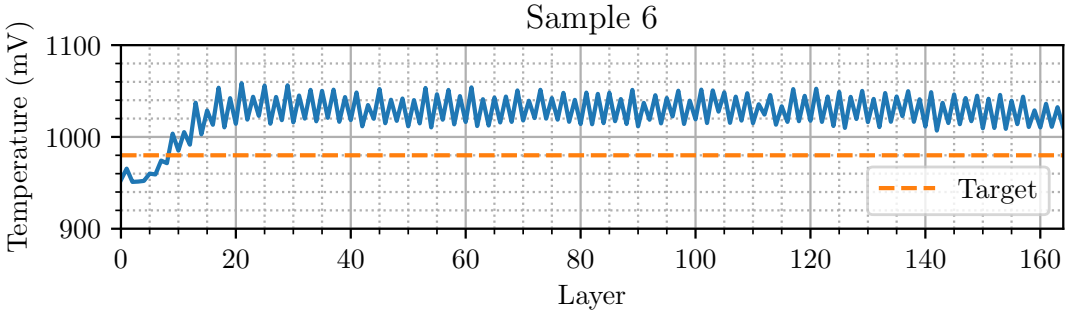


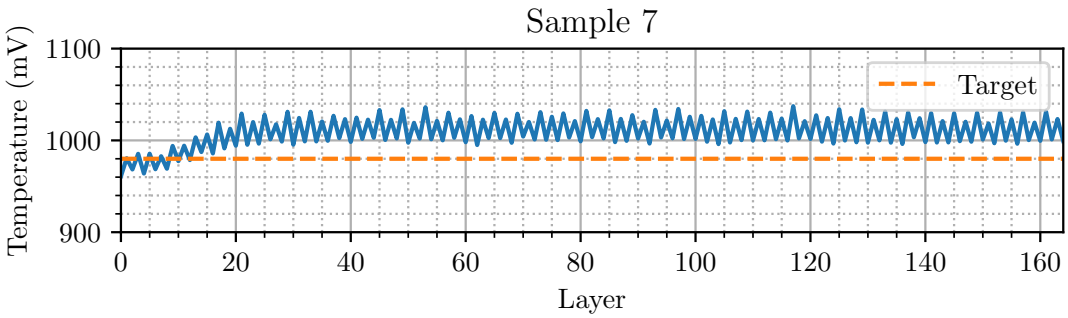

Sample 2
Temperature (mV)
1100
1000
900
0
20
40
60
80
100
120
140
160
Layer
Target

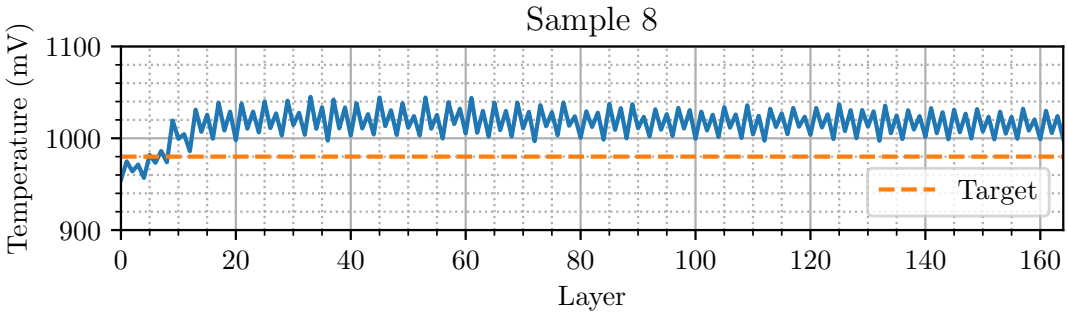

Sample 3
Temperature (mV)
1100
1000
900
0
20
40
60
80
100
120
140
160
Layer
Target

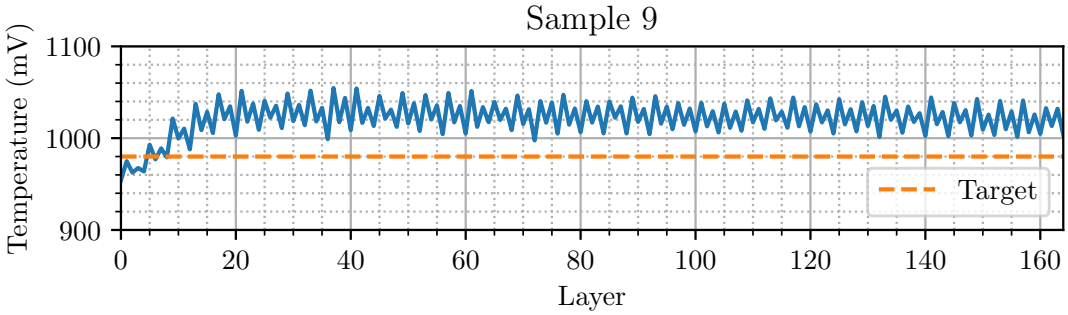

Sample 4
Temperature (mV)
1100
1000
900
0
20
40
60
80
100
120
140
160
Layer
Target

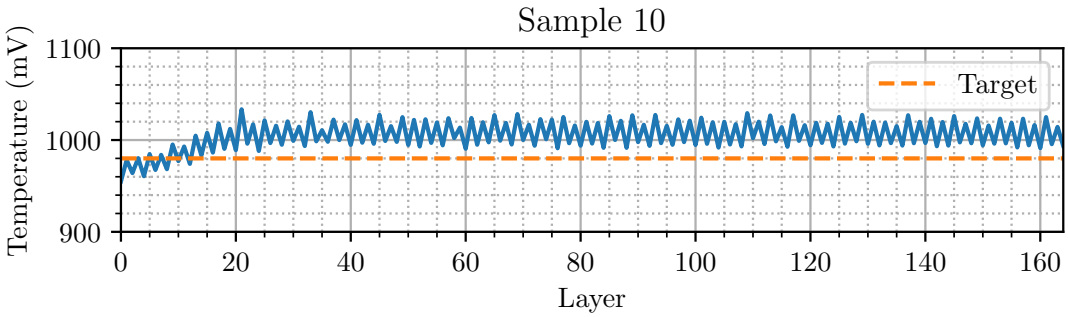

Sample 5
Temperature (mV)
1100
1000
900
0
20
40
60
80
100
120
140
160
Layer
Target


Sample 6
Temperature (mV)
1100
1000
900
0
20
40
60
80
100
120
140
160
Layer
Target

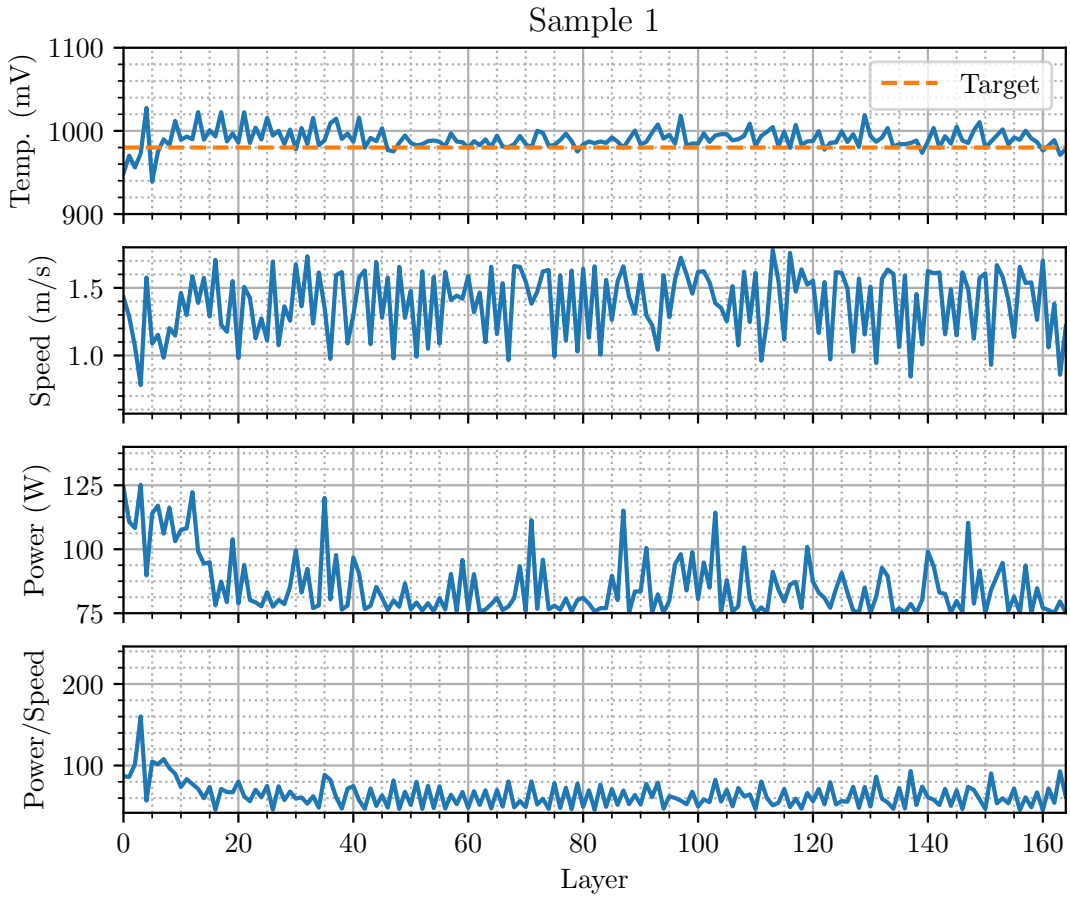

Sample 7
Temperature (mV)
1100
1000
900
0
20
40
60
80
100
120
140
160
Layer
Target

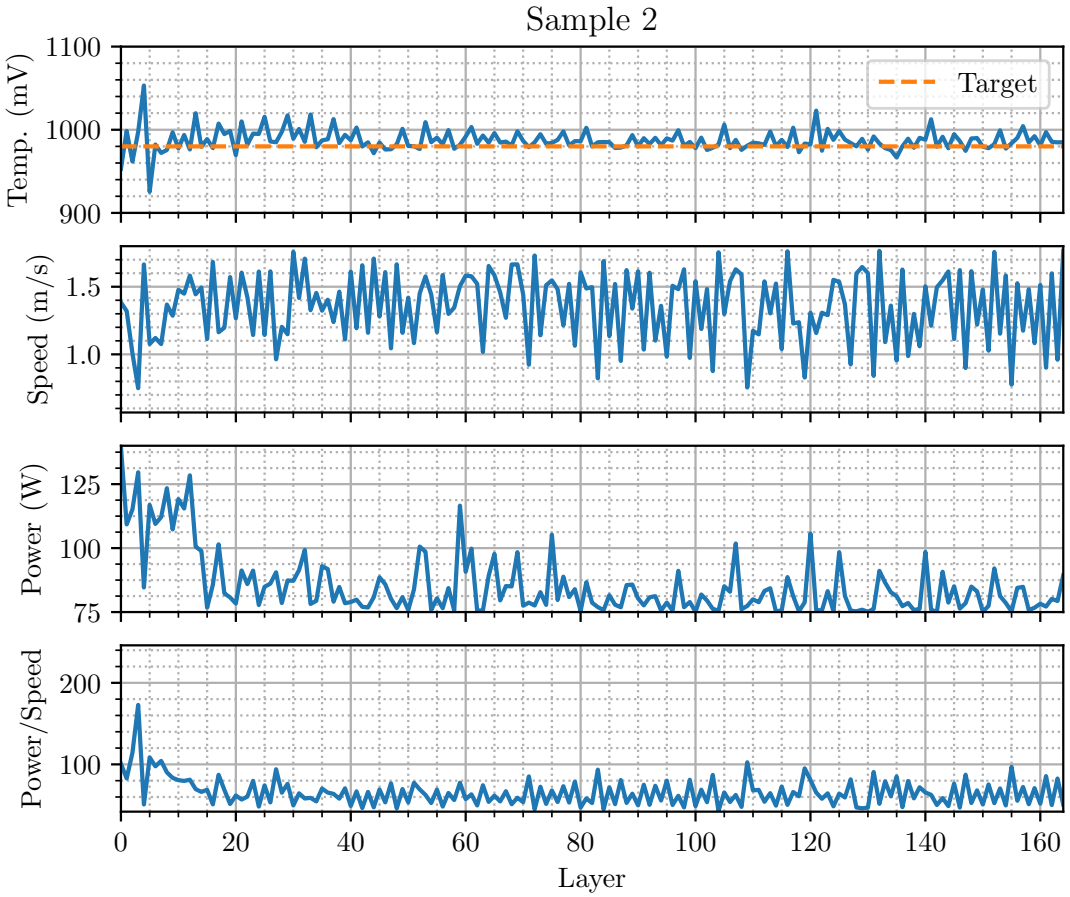

Sample 8
Temperature (mV)
1100
1000
900
0
20
40
60
80
100
120
140
160
Layer
Target

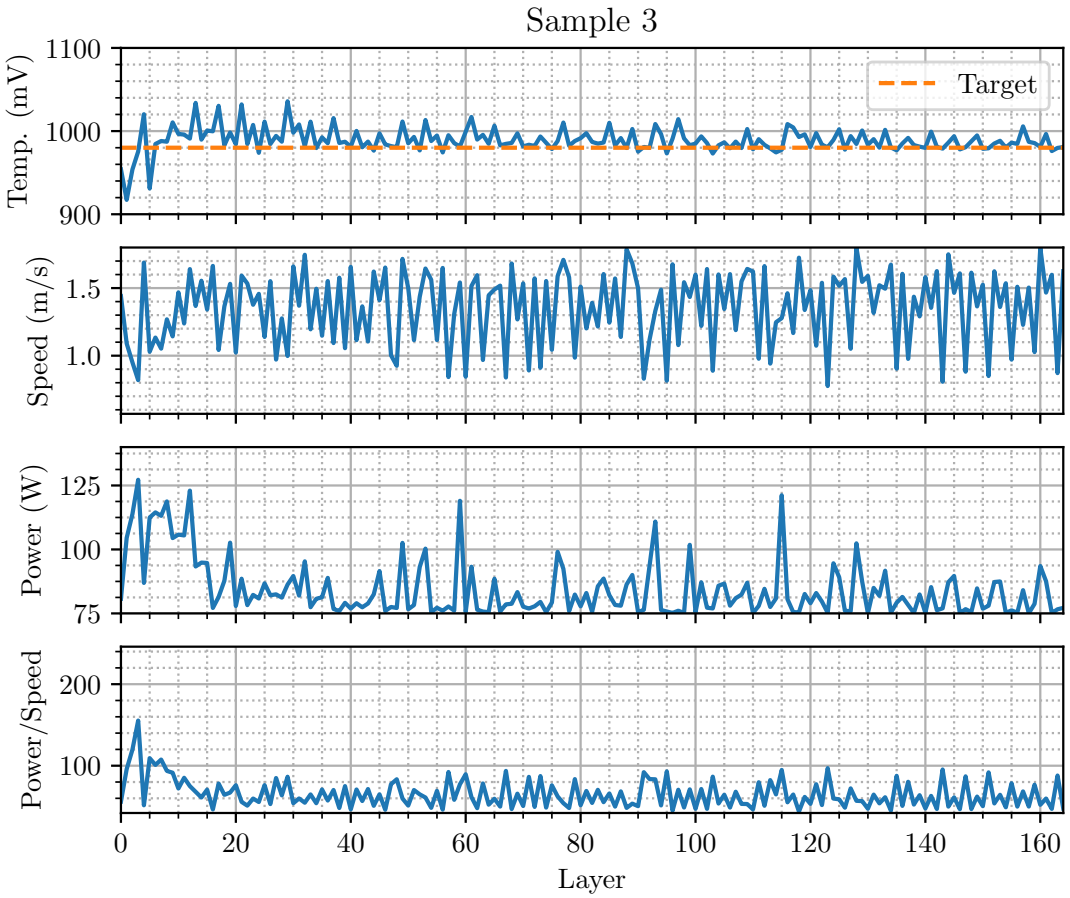

Sample 9
Temperature (mV)
1100
1000
900
0
20
40
60
80
100
120
140
160
Layer
Target

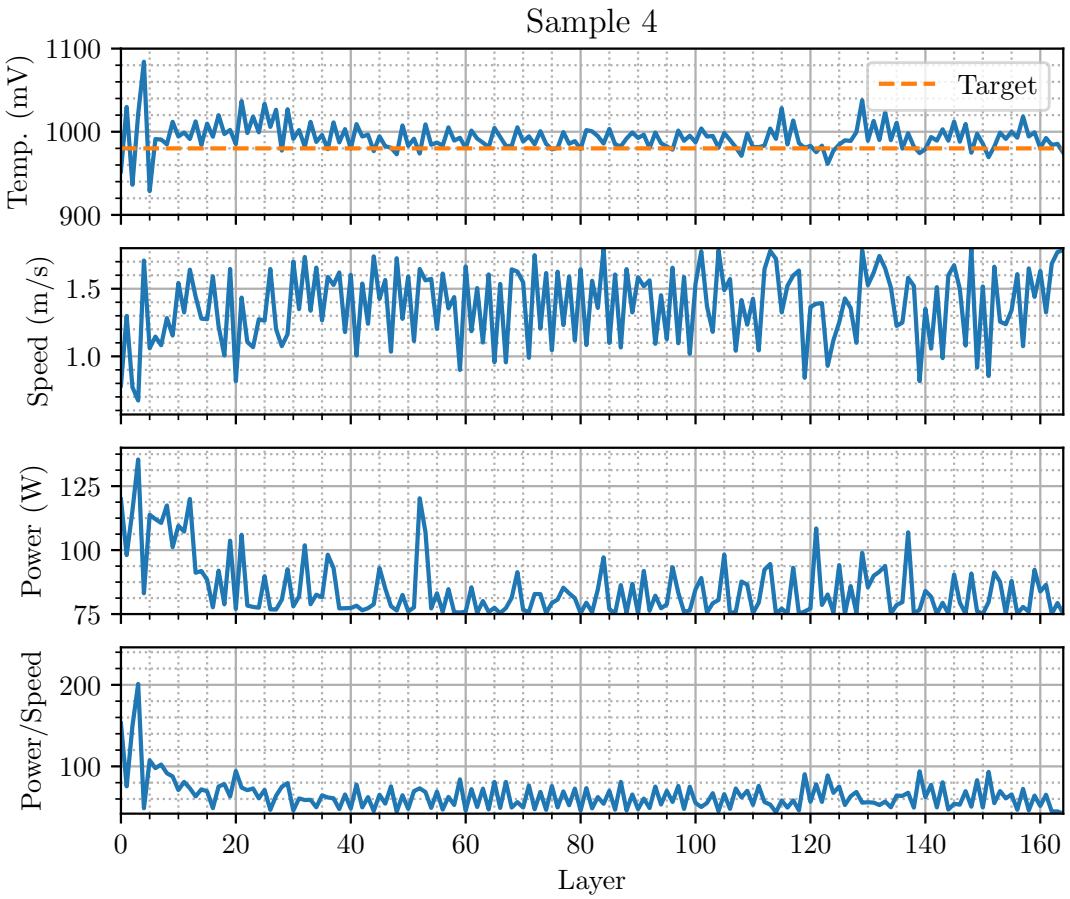

Sample 10
Temperature (mV)
1100
1000
900
0
20
40
60
80
100
120
140
160
Layer
Target

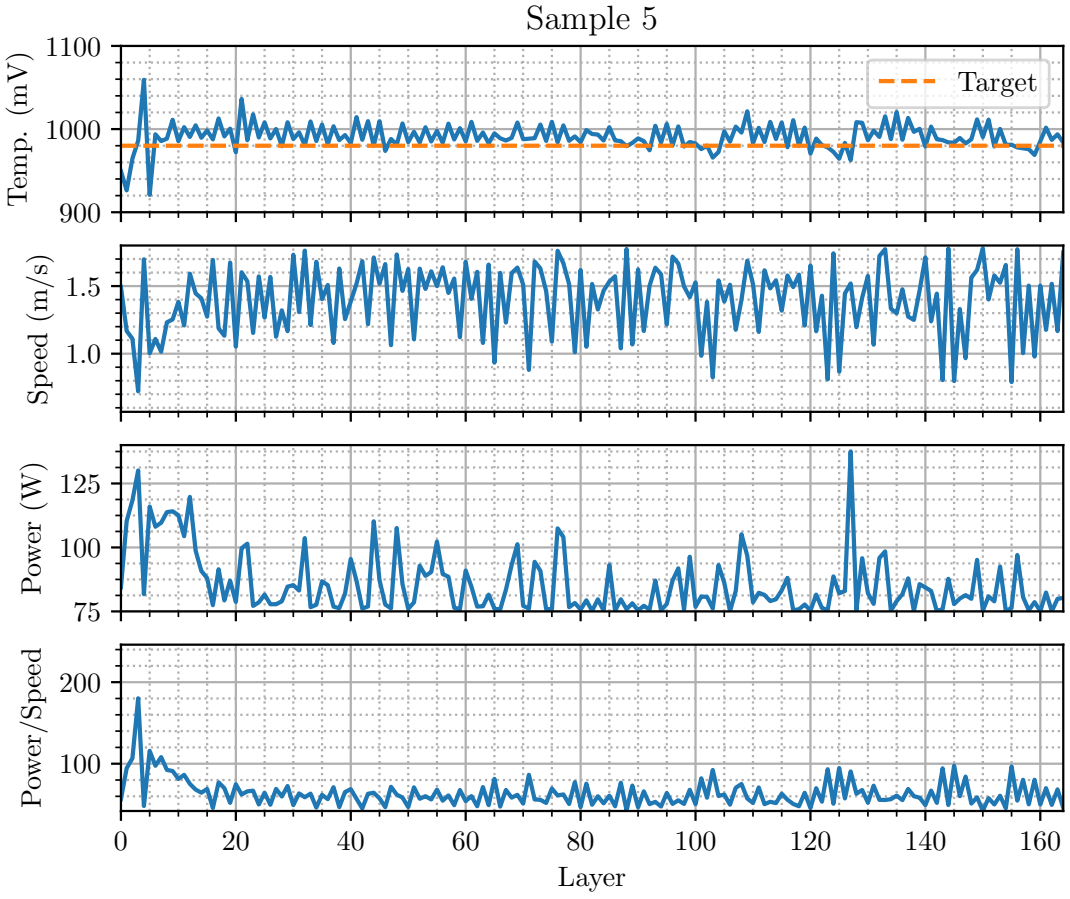
### C.2.2 With Control

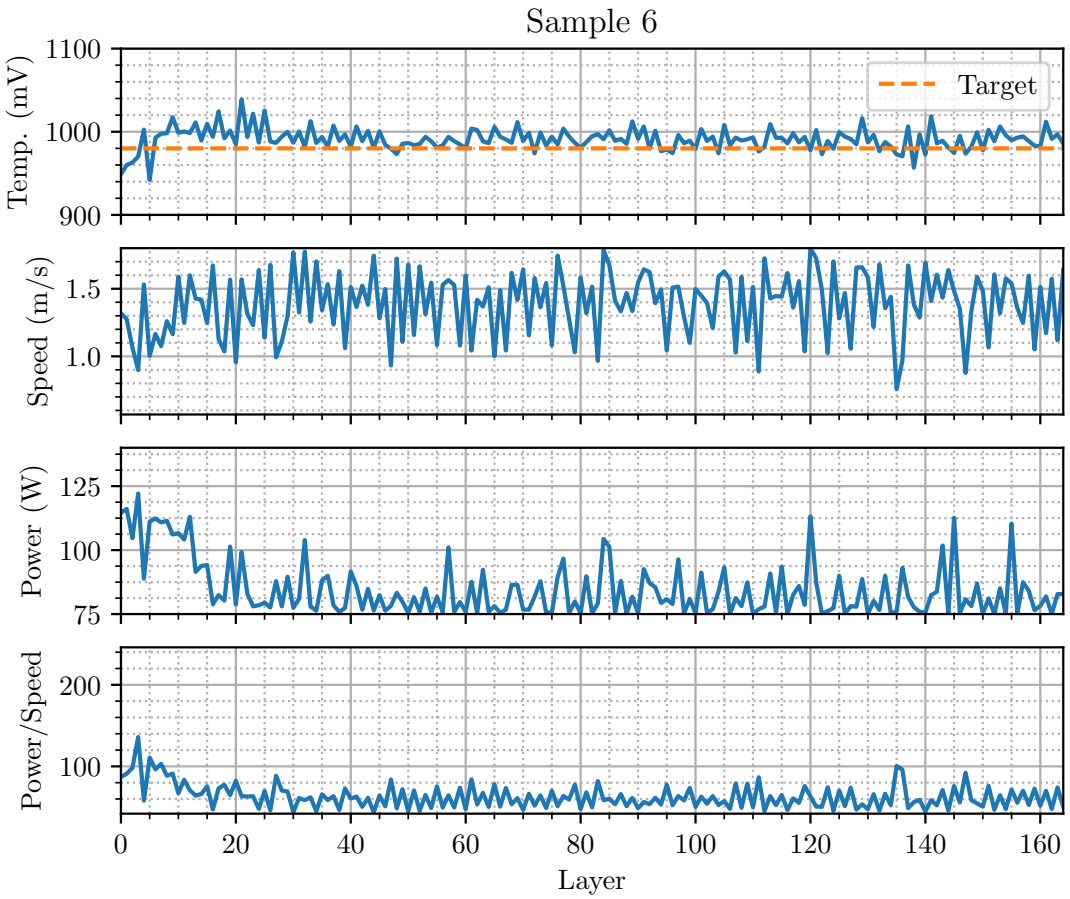


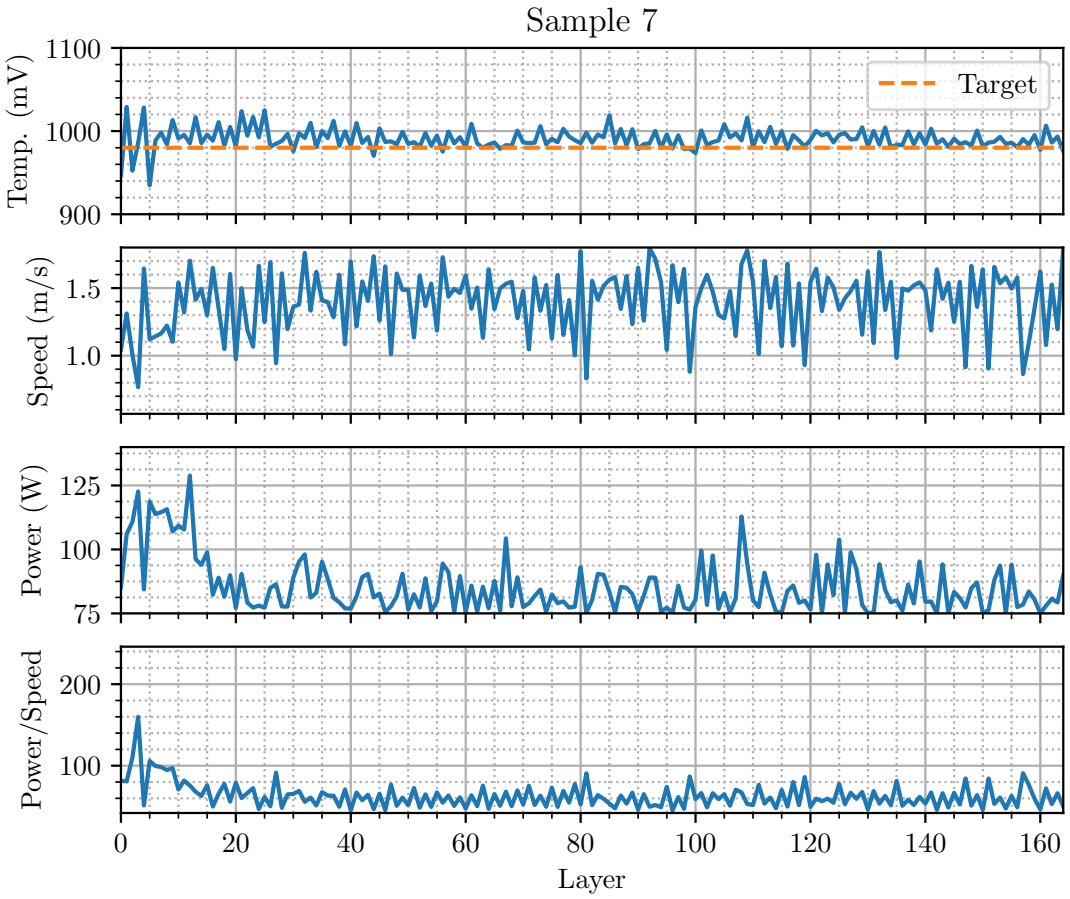

Sample 2
Temp. (mV)
1100
1000
900
Target
Speed (m/s)
1.5
1.0
Power (W)
125
100
75
Power/Speed
200
100
0
20
40
60
80
100
120
140
160
Layer

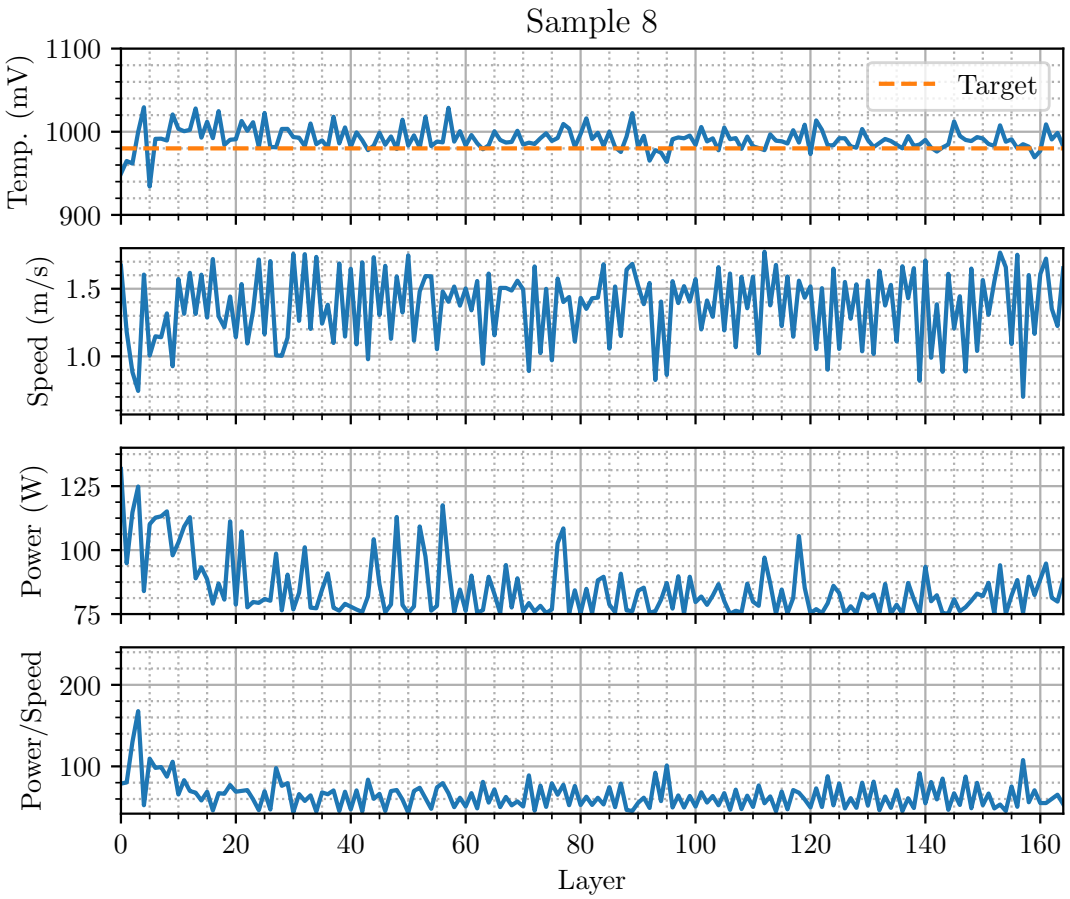

Sample 3
Temp. (mV)
1100
1000
900
Target
Speed (m/s)
1.5
1.0
Power (W)
125
100
75
Power/Speed
200
100
0
20
40
60
80
100
120
140
160
Layer

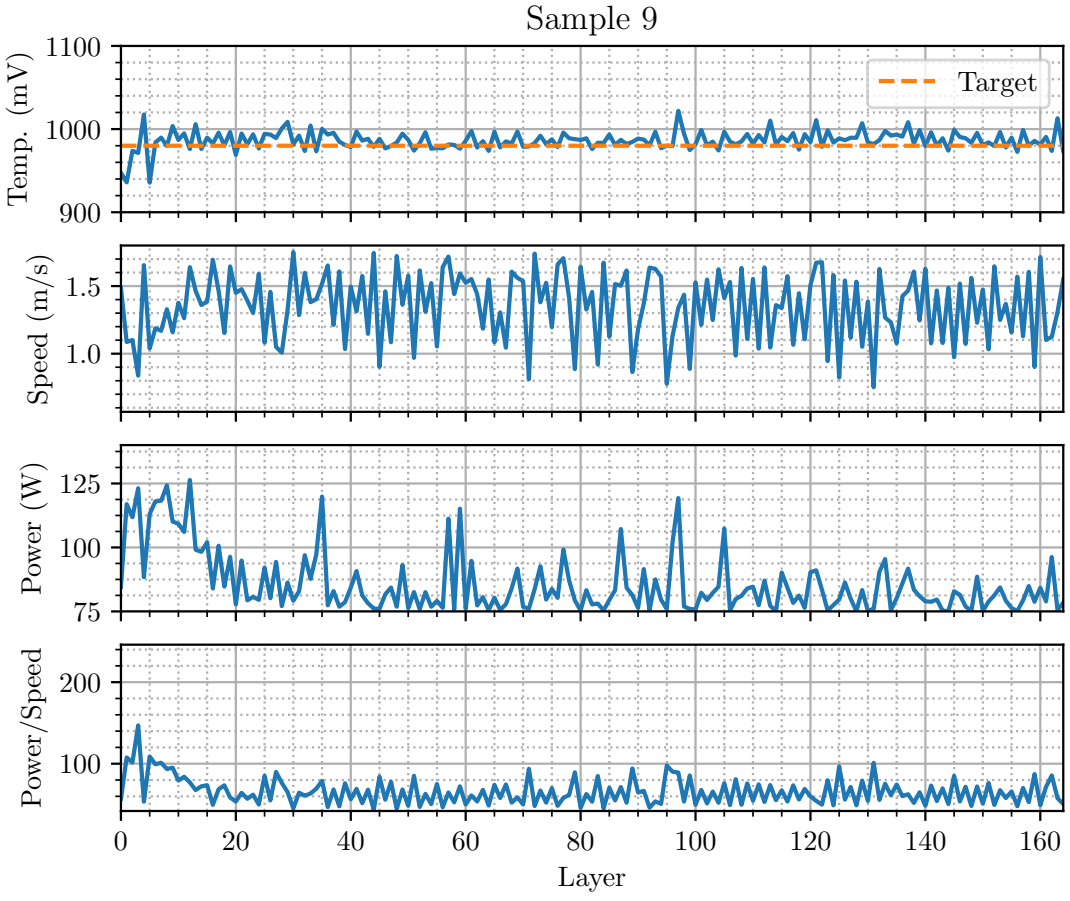

Sample 4
Temp. (mV)
1100
1000
900
Target
Speed (m/s)
1.5
1.0
Power (W)
125
100
75
Power/Speed
200
100
0
20
40
60
80
100
120
140
160
Layer

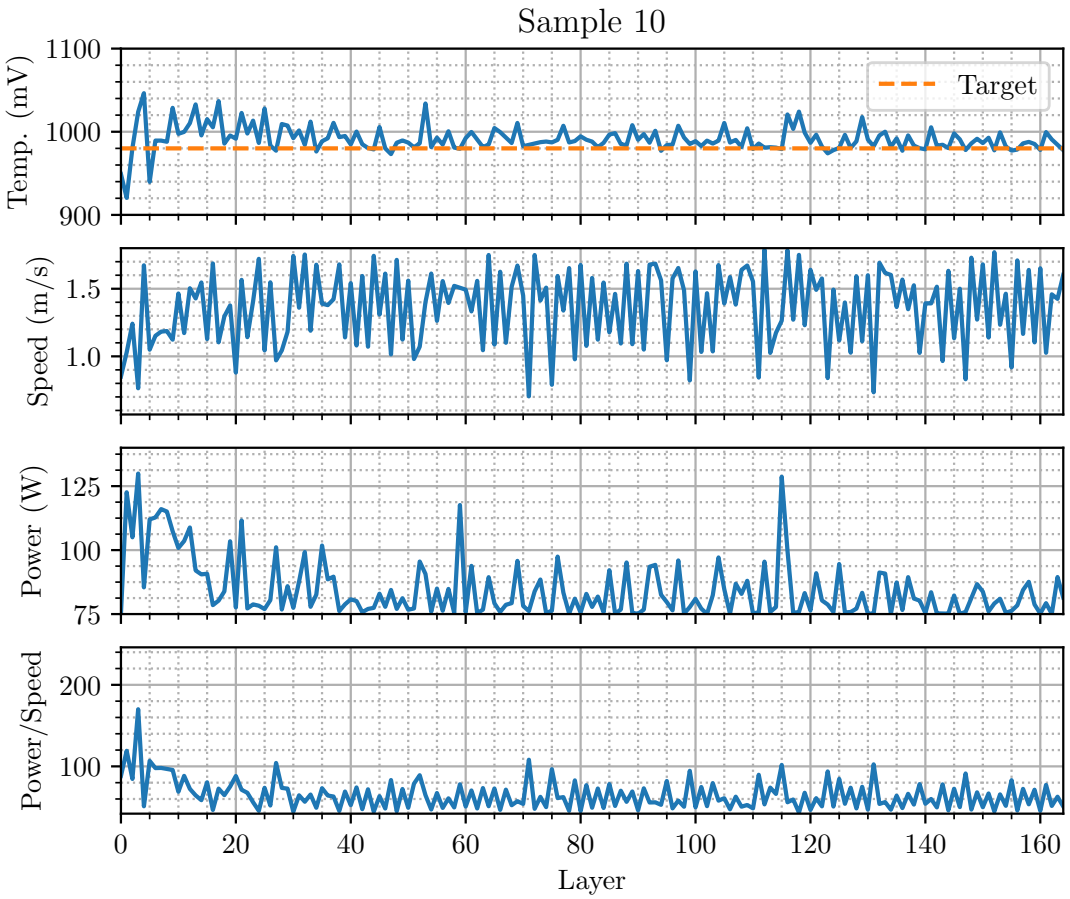

Sample 5
Temp. (mV)
1100
1000
900
Target
Speed (m/s)
1.5
1.0
Power (W)
125
100
75
Power/Speed
200
100
0
20
40
60
80
100
120
140
160
Layer


Sample 6
Temp. (mV)
1100
1000
900
Target
Speed (m/s)
1.5
1.0
Power (W)
125
100
75
Power/Speed
200
100
0
20
40
60
80
100
120
140
160
Layer


Sample 7
Temp. (mV)
1100
1000
900
Target
Speed (m/s)
1.5
1.0
Power (W)
125
100
75
Power/Speed
200
100
0
20
40
60
80
100
120
140
160
Layer


Sample 8
Temp. (mV)
1100
1000
900
Target
Speed (m/s)
1.5
1.0
Power (W)
125
100
75
Power/Speed
200
100
0
20
40
60
80
100
120
140
160
Layer


Sample 9
Temp. (mV)
1100
1000
900
Target
Speed (m/s)
1.5
1.0
Power (W)
125
100
75
Power/Speed
200
100
0
20
40
60
80
100
120
140
160
Layer


Sample 10
Temp. (mV)
1100
1000
900
Target
Speed (m/s)
1.5
1.0
Power (W)
125
100
75
Power/Speed
200
100
0
20
40
60
80
100
120
140
160
Layer

# REFERENCES


[1] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[2] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[3] I. Gibson, D. Rosen, and B. Stucker, *Additive Manufacturing Technologies: 3D Printing, Rapid Prototyping, and Direct Digital Manufacturing*. Springer, 2015.

[4] Z. Y. Chua, I. H. Ahn, and S. K. Moon, "Process monitoring and inspection systems in metal additive manufacturing: Status and applications," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 4, no. 2, pp. 235–245, 2017.

[5] National Aeronautics and Space Administratio, "Standard for additively manufactured spaceflight hardware by laser powder bed fusion in metals," tech. rep., oct 2017.

[6] R. Bellman, "The theory of dynamic programming," tech. rep., RAND Corp Santa Monica CA, 1954.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[8] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[9] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.

[10] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

[11] L. C. Baird III and A. W. Moore, "Gradient descent for general reinforcement learning," in *Advances in neural information processing systems*, pp. 968–974, 1999.

[12] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[13] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[14] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.

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

[16] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, pp. 1889–1897, 2015.

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

[18] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

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

[20] A. Gruslys, M. G. Azar, M. G. Bellemare, and R. Munos, "The reactor: A sample-efficient actor-critic architecture," *arXiv preprint arXiv:1704.04651*, 2017.

[21] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," *arXiv preprint arXiv:1611.01224*, 2016.

[22] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, *et al.*, "Learning to navigate in complex environments," *arXiv preprint arXiv:1611.03673*, 2016.

[23] K. Kansky, T. Silver, D. A. Mély, M. Eldawy, M. Lázaro-Gredilla, X. Lou, N. Dorfman, S. Sidor, S. Phoenix, and D. George, "Schema networks: Zero-shot transfer with a generative causal model of intuitive physics," *arXiv preprint arXiv:1706.04317*, 2017.

[24] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum, "A compositional object-based approach to learning physical dynamics," *arXiv preprint arXiv:1612.00341*, 2016.

[25] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, *et al.*, "Interaction networks for learning about objects, relations and physics," in *Advances in neural information processing systems*, pp. 4502–4510, 2016.

[26] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *International Conference on Machine Learning*, pp. 2829–2838, 2016.

[27] G. Kalweit and J. Boedecker, "Uncertainty-driven imagination for continuous deep reinforcement learning," in *Conference on Robot Learning*, pp. 195–206, 2017.

[28] V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine, "Model-based value estimation for efficient model-free reinforcement learning," *arXiv preprint arXiv:1803.00101*, 2018.

[29] J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee, "Sample-efficient reinforcement learning with stochastic ensemble value expansion," *arXiv preprint arXiv:1807.01675*, 2018.

[30] B. C. Stadie, S. Levine, and P. Abbeel, "Incentivizing exploration in reinforcement learning with deep predictive models," *arXiv preprint arXiv:1507.00814*, 2015.

[31] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," *arXiv preprint arXiv:1809.01999*, 2018.

[32] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evolutionary computation*, vol. 11, no. 1, pp. 1–18, 2003.

[33] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," *arXiv preprint arXiv:1802.10592*, 2018.

[34] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft, "Learning and policy search in stochastic dynamical systems with bayesian neural networks," *arXiv preprint arXiv:1605.07127*, 2016.

[35] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, pp. 1–9, 2013.

[36] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, "Combining model-based and model-free updates for trajectory-centric reinforcement learning," *arXiv preprint arXiv:1703.03078*, 2017.

[37] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 512–519, IEEE, 2016.

[38] N. Wahlström, T. B. Schön, and M. P. Deisenroth, "From pixels to torques: Policy learning with deep dynamical models," *arXiv preprint arXiv:1502.02251*, 2015.

[39] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 2786–2793, IEEE, 2017.

[40] I. Lenz, R. A. Knepper, and A. Saxena, "Deepmpc: Learning deep latent features for model predictive control.," in *Robotics: Science and Systems*, 2015.

[41] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Advances in neural information processing systems*, pp. 2746–2754, 2015.

[42] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *arXiv preprint arXiv:1805.12114*, 2018.

[43] M. Mani, S. Feng, B. Lane, A. Donmez, S. Moylan, and R. Fesperman, "Measurement science needs for real-time control of additive manufacturing powder bed fusion processes," 2015.

[44] T. Craeghs, S. Clijsters, E. Yasa, and J.-P. Kruth, "Online quality control of selective laser melting," in *Proceedings of the Solid Freeform Fabrication Symposium, Austin, TX*, pp. 212–226, 2011.

[45] K. Zeng, D. Pal, and B. Stucker, "A review of thermal analysis methods in laser sintering and selective laser melting," in *Proceedings of Solid Freeform Fabrication Symposium Austin, TX*, vol. 60, pp. 796–814, 2012.

[46] H. Meier and C. Haberland, "Experimental studies on selective laser melting of metallic parts," *Materialwissenschaft und Werkstofftechnik*, vol. 39, no. 9, pp. 665–670, 2008.

[47] K. Abd-Elghany and D. Bourell, "Property evaluation of 304l stainless steel fabricated by selective laser melting," *Rapid Prototyping Journal*, vol. 18, no. 5, pp. 420–428, 2012.

[48] A. Wegner and G. Witt, "Correlation of process parameters and part properties in laser sintering using response surface modeling," *Physics Procedia*, vol. 39, pp. 480–490, 2012.

[49] C. Knowles, T. Becker, and R. Tait, "Residual stress measurements and structural integrity implications for selective laser melted ti-6al-4v," *South African Journal of Industrial Engineering*, vol. 23, no. 3, pp. 119–129, 2012.

[50] T. Sercombe, N. Jones, R. Day, and A. Kop, "Heat treatment of ti-6al-7nb components produced by selective laser melting," *Rapid Prototyping Journal*, vol. 14, no. 5, pp. 300–304, 2008.

[51] S. Leuders, M. Thöne, A. Riemer, T. Niendorf, T. Tröster, H. Richard, and H. Maier, "On the mechanical behaviour of titanium alloy tial6v4 manufactured by selective laser melting: Fatigue resistance and crack growth performance," *International Journal of Fatigue*, vol. 48, pp. 300–307, 2013.

[52] E. Yasa, J. Deckers, T. Craeghs, M. Badrossamay, and J.-P. Kruth, "Investigation on occurrence of elevated edges in selective laser melting," in *International Solid Freeform Fabrication Symposium, Austin, TX, USA*, pp. 673–85, 2009.

[53] J. Delgado, J. Ciurana, and C. A. Rodríguez, "Influence of process parameters on part quality and mechanical properties for dmls and slm with iron-based materials," *The International Journal of Advanced Manufacturing Technology*, vol. 60, no. 5-8, pp. 601–610, 2012.

[54] R. Morgan, C. Sutcliffe, and W. O'neill, "Density analysis of direct metal laser re-melted 316l stainless steel cubic primitives," *Journal of materials science*, vol. 39, no. 4, pp. 1195–1205, 2004.

[55] E. Yasa, J. Deckers, and J.-P. Kruth, "The investigation of the influence of laser re-melting on density, surface quality and microstructure of selective laser melting parts," *Rapid Prototyping Journal*, vol. 17, no. 5, pp. 312–327, 2011.

[56] W. E. King, H. D. Barth, V. M. Castillo, G. F. Gallegos, J. W. Gibbs, D. E. Hahn, C. Kamath, and A. M. Rubenchik, "Observation of keyhole-mode laser melting in laser powder-bed fusion additive manufacturing," *Journal of Materials Processing Technology*, vol. 214, no. 12, pp. 2915–2925, 2014.

[57] D. Gu, Y.-C. Hagedorn, W. Meiners, G. Meng, R. J. S. Batista, K. Wissenbach, and R. Poprawe, "Densification behavior, microstructure evolution, and wear performance of selective laser melting processed commercially pure titanium," *Acta Materialia*, vol. 60, no. 9, pp. 3849–3860, 2012.

[58] N. T. Aboulkhair, N. M. Everitt, I. Ashcroft, and C. Tuck, "Reducing porosity in alsi10mg parts processed by selective laser melting," *Additive Manufacturing*, vol. 1, pp. 77–86, 2014.

[59] I. Yadroitsev, P. Krakhmalev, and I. Yadroitsava, "Hierarchical design principles of selective laser melting for high quality metallic objects," *Additive Manufacturing*, vol. 7, pp. 45–56, 2015.

[60] H. E. Helmer, C. Körner, and R. F. Singer, "Additive manufacturing of nickel-based superalloy inconel 718 by selective electron beam melting: Processing window and microstructure," *Journal of Materials Research*, vol. 29, no. 17, pp. 1987–1996, 2014.

[61] H. Gong, K. Rafi, H. Gu, T. Starr, and B. Stucker, "Analysis of defect generation in ti–6al–4v parts made using powder bed fusion additive manufacturing processes," *Additive Manufacturing*, vol. 1, pp. 87–98, 2014.

[62] S. Storch, D. Nellessen, G. Schaefer, and R. Reiter, "Selective laser sintering: qualifying analysis of metal based powder systems for automotive applications," *Rapid Prototyping Journal*, vol. 9, no. 4, pp. 240–251, 2003.

[63] J.-P. Kruth, J. Duflou, P. Mercelis, J. Van Vaerenbergh, T. Craeghs, and J. De Keuster, "On-line monitoring and process control in selective laser melting and laser cutting," in *Proceedings of the 5th Lane Conference, Laser Assisted Net Shape Engineering*, vol. 1, pp. 25–28, 2007.

[64] T. Craeghs, S. Clijsters, J.-P. Kruth, F. Bechmann, and M.-C. Ebert, "Detection of process failures in layerwise laser melting with optical process monitoring," *Physics Procedia*, vol. 39, pp. 753–759, 2012.

[65] S. Berumen, F. Bechmann, S. Lindner, J.-P. Kruth, and T. Craeghs, "Quality control of laser-and powder bed-based additive manufacturing (am) technologies," *Physics procedia*, vol. 5, pp. 617–622, 2010.

[66] R. B. Dinwiddie, R. R. Dehoff, P. D. Lloyd, L. E. Lowe, and J. B. Ulrich, "Thermographic in-situ process monitoring of the electron-beam melting technology used in additive manufacturing," in *Thermosense: Thermal Infrared Applications XXXV*, vol. 8705, p. 87050K, International Society for Optics and Photonics, 2013.

[67] S. Price, K. Cooper, and K. Chou, "Evaluations of temperature measurements by near-infrared thermography in powder-based electron-beam additive manufacturing," in *Proceedings of the Solid Freeform Fabrication Symposium*, pp. 761–773, University of Texas, Austin, TX, 2012.

[68] T. Craeghs, F. Bechmann, S. Berumen, and J.-P. Kruth, "Feedback control of layerwise laser melting using optical sensors," *Physics Procedia*, vol. 5, pp. 505–514, 2010.

[69] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, pp. 9–48, Springer, 2012.

[70] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[71] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in neural information processing systems*, pp. 950–957, 1992.

[72] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[73] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

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

[75] G. B. Orr and K.-R. Müller, *Neural networks: tricks of the trade*. Springer, 2003.

[76] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 2951–2959, Curran Associates, Inc., 2012.

[77] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "Boa: The bayesian optimization algorithm," in *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 1*, pp. 525–532, Morgan Kaufmann Publishers Inc., 1999.

[78] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[79] G. Bi, C. Sun, and A. Gasser, "Study on influential factors for process monitoring and control in laser aided additive manufacturing," *Journal of Materials Processing Technology*, vol. 213, no. 3, pp. 463–468, 2013.