Reinforcement Learning for Optimization and Control of Ultracold Quantum Gas Production

by

Nicholas Milson

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Physics
University of Alberta

Abstract

Machine-learning techniques are emerging as a valuable tool in experimental physics, and among them, reinforcement learning offers the potential to control high-dimensional, multistage processes in the presence of fluctuating environments.

In this experimental work, we apply reinforcement learning to the preparation of an ultracold quantum gas to realize a consistent and large number of atoms at microkelvin temperatures. This reinforcement learning agent determines an optimal set of thirty control parameters in a dynamically changing environment that is characterized by thirty sensed parameters. By comparing this method to that of training supervised–learning regression models, as well as to human-driven control schemes, we find that both machine learning approaches accurately predict the number of cooled atoms and both result in occasional superhuman control schemes. However, only the reinforcement learning method achieves consistent outcomes, even in the presence of a dynamic environment.

This thesis provides a comprehensive overview of the theoretical groundwork necessary for understanding the experimental sequence and machine learning techniques employed. Technical details of the cooling apparatus and machine learning agents are presented, along with the results of allowing the trained machine learning agents to autonomously control the cooling sequence.

Preface

The contributions to the work presented in this thesis are as follows.

The atom cooling apparatus was designed by Dr. Lindsay LeBlanc. It was constructed and subsequently modified over the years by many people, including Dr. Lindsay LeBlanc, Dr. Erhan Saglamyurek, Dr. Andrei Tretiakov, Dr. Arina Tashchilina, Dr. Logan Cooke, Greg Popowich, Taras Hrushevskyi, Benjamin Smith, Joseph Lindon, Manvir Gill, and myself.

The environmental parameter sensing hardware was built by Tian Ooi, Anna Czarnecka, James Maldaner, and myself. Environmental parameter post processing software and integration with the machine learning agents was done by Tian Ooi and myself.

Design, programming, and integration with the system of the machine learning methods were done by myself, with fruitful consultation and discussion with Dr. Arina Tashchilina, Zaheen Ahmad, and Abilmansur Zhumabekov. Data for the machine learning agents was collected by Dr. Arina Tashchilina, Tian Ooi, and myself.

Subsequent modifications and repairs to the system, based on conclusions from this thesis work, were done by Dr. Arina Tashchilina, Manvir Gill, and myself.

The results of the machine learning algorithms, serving both as predictive tools and for real-time control of atom cooling, are presented in the publication N Milson *et al* 2023 *Mach. Learn.: Sci. Technol.* 4 045057. Sections of figures and text from this article have been incorporated into this thesis, subject to varying degrees of modification for contextual relevance and coherence

Acknowledgements

I would like to express my gratitude to all of the members of my research group for their patience, support, and friendship.

I extend special thanks to my supervisor Dr. Lindsay LeBlanc, postdoctoral fellow Dr. Arina Tashchilina, and senior graduate students Dr. Logan Cooke and Joey Lindon. Their guidance has been invaluable in helping me navigate the challenges of graduate school and adapt to this new academic environment. I have learned a lot from these role models about how a scientist ought to operate, and I feel that with the conclusion of this Master's work and the beginning of my Ph.D. candidacy, my work as an experimental physicist is just getting started, and I am primed to embark on some new and productive scientific endeavours.

I would also like to thank Anindya Rastogi, Anna Czarnecka, Kusum Meena, and Tian Ooi who frequently worked alongside me in the lab, for always being willing to chat and share a laugh, which made work more fun than the hours in between.

Lastly, I would like to acknowledge my parents, Susan and Rob, for supporting my interests from day one, and my girlfriend Cali for her extreme patience as I strive towards competency as an independent scientist.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

As a general tool, machine learning (ML) offers remarkable advantages in far-reaching domains, ranging from large-language models to control-systems instrumentation. In the realm of scientific research, ML promises to improve the design, control, and optimization of experimental processes, particularly when these procedures are high-dimensional, when uncontrolled environmental factors affect outcomes, and when systematic optimization is implausible, leaving only intuition or trial and error [1]. The production of cold, dense ensembles of neutral atoms for Bose-Einstein condensates (BECs), for example, involves many detailed steps, each with several free parameters [2]. While experimentalists have risen to the challenge to create systems that consistently result in BECs, applications of ML to optimize magneto-optical traps [3, 4], evaporation curves [5–9], and simultaneous laser and evaporative cooling [10, 11] show the potential for creating greater reliability for these systems.

These impressive results in atom-cooling applications are examples of supervised ML, in which a model learns from labeled examples, aiming to predict or classify new instances of the same problem based on prior training. Outside of atom-cooling applications, supervised machine learning has been used successfully in many areas of physics, such as searches for exotic particles in high-energy physics [12], predictions of eigenvalues in photonic crystals [13], determination of transitions between many-body localized and thermalizing regimes in many-body quantum systems [14], and even reconstructions of the Cosmic Microwave Background [15]. So far, the supervised learning atom-cooling approaches have used Gaussian processes [16] or deep neural networks [17] to apply a ML regression model (sometimes known as a surrogate function), and then have used the trained model to find experimental control inputs that maximize the predicted output. These methods to find efficient cooling schemes demonstrated superior performance compared to human optimization and direct numerical optimization techniques, such as differential evolution [5]. However, in scenarios where optimal control parameters depend on changing environmental conditions, these approaches are inadequate. A direct maximization via Bayesian optimization [18] is unsuited, as the objective function varies with the environment. Furthermore, modeling the output metric as a

function of controllable and environmental factors and optimizing the model alone is likely insufficient, as previous investigations that only considered the controllable factors failed to yield consistent superhuman control parameter choices [3, 10]. Robustness of the found experimental control inputs is now also a consideration, as the usefulness of finding a single high-performing set is now highly dependent on how strongly the performance of this set changes with the dynamic environment.

Beyond the scope of supervised ML, machine learning also includes two more general approaches: unsupervised ML and reinforcement learning (RL). Unsupervised ML deals with finding patterns and structures in unlabeled data, through techniques such as clustering and dimensionality reduction. Some example use-cases for unsupervised learning in physics are the detection of both quantum and classical phase transitions [19–22], as well as generative models for complex physical systems [23, 24]. RL, on the other hand, represents a class of problems where an agent interacts with an environment, learning to make sequential decisions to maximize cumulative rewards. While still a young subfield in physics, the power of reinforcement learning has been demonstrated in the control of noisy and many-body quantum systems [25–30], control of nuclear fusion apparatuses [31], optimal design of molecules for different chemical applications [32], experimental control and navigation in stochastic turbulent environments such as micro-sphere carrying optical tweezers [33] and microorganism simulating artificial nano-swimmers [34]. Despite this recent wave of RL successes, live, autonomous optimization and control of atom-cooling apparatuses with RL remains largely unexplored, particularly in the context of high-dimensional control–parameter and environmental-parameter spaces.

In this thesis work, we optimize a rubidium-87 atom cooling experiment using ML approaches that consider the environment the apparatus is in. We focus our work on the crucial initial production stages of a BEC, including laser cooling and trapping, up to a high-field-gradient magnetic trap (MT) [35, 36]. Our high-dimensional ML schemes include thirty measured environmental parameters, sensed at specific times throughout the experimental sequence, in addition to thirty control parameters that are subject to ML optimization. We develop an RL controller that determines optimal control parameters based on the current environmental conditions, and find that the overall atom number achieved exceeds that of all other methods. We specifically compare this RL controller to a supervised regression model that maps the combined input space of control parameters and sensed environmental parameters to the number of atoms in the trap, and uses the model to find optimal control parameters that maximize atom number for a given environmental state. Our results show that RL offers unique advantages to experimental control in high-dimensional systems, especially for its ability to react to drifts in the environmental conditions that have influence over the outcomes

Chapter 2 introduces the theory behind the machine learning methods used in this thesis. Broadly, supervised learning, reinforcement learning, and function optimization are described.

Chapter 3 introduces the physics of the atom cooling techniques used in this thesis. The principles of magneto-optical trapping, motion induced orientation sub-Doppler cooling, and magnetic trapping are specifically described.

Chapter 4 presents a detailed description of the experimental system used in this thesis. The cooling apparatus is described, including the laser, vacuum, and magnetic field controlling systems. Furthermore, the aspects of the cooling system which the machine learning methods are given control of are described. The estimation of atom number, as well as sensing and processing of environmental factors are also described.

Chapter 5 details the design of the two artificially intelligent agents used in this thesis. Both agents are built to optimize the number of atoms following the cooling procedure in response to environmental changes, but the philosophy of the two agents are fundamentally different. The first is built upon the optimization of a supervised learning model, whereas the other is a reinforcement learning agent trained explicitly to learn a probabilistic policy. In this section, an algorithm to use the models to determine the relative importance of environmental factors is also described.

Chapter 6 presents the results of deploying both agents. The predictive capabilities are tested, as well as the capability to generate control parameter sets resulting in large atom number, in response to environmental changes. The live control performance is contrasted with other baselines to demonstrate the suitability of the two agents for use in such apparatuses.

Chapter 7 summarizes the work done in this thesis, and presents a retrospective on the aspects of this work that may be considered relative successes and failures. Lastly, realistic extensions to this work are given, both in general as well as specifically for our laboratory.

Chapter 2

Background Considerations: Machine Learning

This chapter seeks to introduce the theory behind the machine learning numerics used in this work, with physicists as an intended audience. This chapter is organized as follows: first, machine learning as a field is broadly described and categorized. The areas of machine learning applied in this thesis, namely supervised and reinforcement learning, are then described in more detail. For supervised learning, neural networks as universal function approximators are described, along with the mechanism of how they are trained. Within reinforcement learning, policy gradient methods are the main focus, specifically actor-critic methods. The mechanisms and intuitions behind training such actor-critics are described. Lastly, two methods to optimize the output of neural network models are described. Specifically, the Nelder–Mead method, and the method of Bayesian optimization with Gaussian processes.

2.1 What is Machine Learning?

Machine learning is the field concerned with the development of algorithms to learn from data automatically, i.e. without explicit programming. Instead of following rigid, rule-based instructions, machine learning models adapt and improve their performance by learning from experience.

The field of machine learning may be conceptualized as dividing into three subcategories: supervised machine learning, unsupervised machine learning, and reinforcement learning. Supervised machine learning is an approach where a model learns from labeled examples, aiming to predict or classify new instances based on prior training. Unsupervised machine learning, on the other hand, deals with finding patterns and structures in unlabeled data, through techniques such as clustering and dimensionality reduction. Lastly, reinforcement learning represents a class of problem where an agent interacts with an environment, learning to make sequential decisions to maximize cumulative rewards. The content of this thesis will make use

of supervised and reinforcement learning.

2.2 Supervised Learning

Supervised learning is the subfield concerned with building models to map input data to desired output labels by using a labeled dataset. This can be symbolically formulated in the following way. Given a dataset $D = (\mathbf{X}, \mathbf{Y})$, where $\mathbf{X}$ is a matrix of input variables and $\mathbf{Y}$ is a matrix of output variables, some model $\mathbf{f}(\mathbf{x};\boldsymbol{\theta})$ maps a given input vector $\mathbf{x}$ to output vector $\mathbf{y}$ via a set of adjustable parameters $\boldsymbol{\theta}$. A loss function $L(\mathbf{Y}, \mathbf{f}(\mathbf{X};\boldsymbol{\theta}))$, sometimes analogously called a cost function, is then defined to evaluate the quality of the model, so that the optimal model parameters can be found by minimizing said loss function, i.e.

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\{L(\mathbf{Y}, \mathbf{f}(\mathbf{X};\boldsymbol{\theta}))\} \,. \tag{2.1}$$

A default choice of loss function is the sum-of-squared residuals.

A key challenge in supervised learning is finding a balance between "training" a model that can make an accurate mapping and ensuring that it does not overfit, meaning it becomes too specialized in fitting the training data, resulting in poor generalization to out-of-sample data. This challenge is often addressed by splitting the dataset into two subsets: a training set and an evaluation set, where the evaluation set is withheld during training and used to evaluate the models performance on out-of-sample data.

2.2.1 Neural Networks

An artificial neural network, or simply a neural network, is a powerful model often used for $\mathbf{f}(\mathbf{X};\boldsymbol{\theta})$ [37]. In essence, neural networks consist of a series of interconnected layers, each comprising linear transformations and non-linear functions known as activation functions. A given layer i, being a linear transformation, consists of a matrix of slopes or "weights" $\mathbf{W}^{(i)} = (\mathbf{w}_1^{(i)}, \mathbf{w}_2^{(i)}, ..., \mathbf{w}_d^{(i)})$, and a vector of intercepts or "biases" $\mathbf{b}^{(i)} = (b_1^{(i)}, b_2^{(i)}, ..., b_d^{(i)})$, such that a layer accepts an input vector $\mathbf{x}^{(i)}$ and outputs

$$\mathbf{z}^{(i)} = \mathbf{W}^{(i)}\mathbf{x}^{(i)} + \mathbf{b}^{(i)} \,. \tag{2.2}$$

This output vector is then passed through the activation function, which may then be used as the input for subsequent layers. i.e.

$$\mathbf{x}^{(i+1)} = \sigma^{(i)}(\mathbf{z}^{(i)}) \,. \tag{2.3}$$

The layers situated between the initial input and the final output layers are referred to as "hidden" layers. When a neural network comprises multiple hidden layers, it is termed a "deep"

neural network.

The power of neural networks lies in their ability to approximate complex functions effectively. This ability is described by the universal approximation theorem, which asserts that a deep neural network with a sufficient number of learnable weights can approximate any continuous function with arbitrary accuracy. A complete graphical proof of the universal approximation theorem may be found in [38].

2.2.2 Calculation of Gradients

The training of neural networks is the iterative process of finding a set of optimal network parameters, i.e. weights and biases $\boldsymbol{\theta}^* = (\mathbf{W}, \mathbf{b})$, that satisfy equation 2.1. With a given loss function specified, gradient descent-based algorithms may be used to minimize said loss function. While different gradient descent-based algorithms may vary in sophistication, including elements of stochasticity and dynamical adjustment, the basic algorithm is

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla_{\boldsymbol{\theta}} L(\mathbf{Y}, \mathbf{f}(\mathbf{X}; \boldsymbol{\theta}))\,, \tag{2.4}$$

where the coefficient η_t is known as the learning rate at iteration t, which controls the step size taken in the direction of the gradient.

Directly estimating gradients of the loss function with respect to all parameters at each training iteration using brute force numerical methods would be computationally impractical. However, we can leverage the hierarchical architecture of neural networks by using the back-propagation algorithm [39]. For a neural network with layer number indexed by i, each layer containing a weight matrix with elements $w_{jk}^{(i)}$, and bias vector with elements $b_j^{(i)}$, we wish to determine $\frac{\partial L(\mathbf{Y},\mathbf{f}(\mathbf{X};\boldsymbol{\theta}))}{\partial w_{kj}^{(i)}}$ and $\frac{\partial L(\mathbf{Y},\mathbf{f}(\mathbf{X};\boldsymbol{\theta}))}{\partial b_j^{(i)}}$. See figure 2.1 for a graphical representation of how the weights and biases of the network are indexed. Using the chain rule, we can write the change in the cost function with respect to the output of a given layer's linear transformation as the derivatives of the subsequent layer:

$$\frac{\partial L(\mathbf{Y}, \mathbf{f}(\mathbf{X}; \boldsymbol{\theta}))}{\partial z_j^{(i)}} = \sum_k \frac{\partial L(\mathbf{Y}, \mathbf{f}(\mathbf{X}; \boldsymbol{\theta}))}{\partial z_k^{(i+1)}} \frac{\partial z_k^{(i+1)}}{\partial z_j^{(i)}} \tag{2.5}$$

$$= \sum_k \frac{\partial L(\mathbf{Y}, \mathbf{f}(\mathbf{X}; \boldsymbol{\theta}))}{\partial z_k^{(i+1)}} w_{jk}^{i+1} \frac{\partial \sigma^{(i)}}{\partial z_j^{(i)}}\,, \tag{2.6}$$

where the derivatives of activation functions $\sigma^{(i)}(\mathbf{z}^{(i)})$ are known analytically.

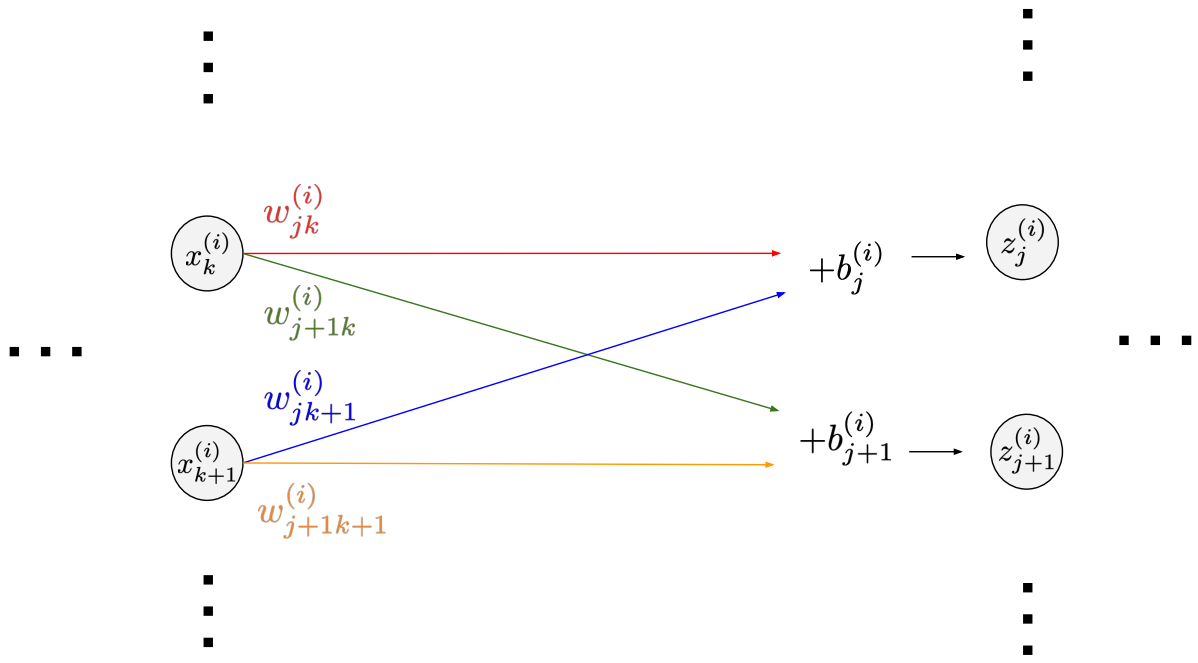

Figure 2.1: Structure of a neural network's linear transforming layer $\mathbf{z}^{(i)} = \mathbf{W}^{(i)}\mathbf{x}^{(i)} + \mathbf{b}^{(i)}$. Index i notates the layer number. Indices k and j respectively notate the input and output vector elements.

With equation 2.6, all one needs to notice is that the derivatives with respect to biases are

$$\frac{\partial L(\mathbf{Y}, \mathbf{f}(\mathbf{X};\boldsymbol{\theta}))}{\partial b_j^{(i)}} = \frac{\partial L(\mathbf{Y}, \mathbf{f}(\mathbf{X};\boldsymbol{\theta}))}{\partial z_j^{(i)}} \frac{\partial z_j^{(i)}}{\partial b_j^{(i)}} = \frac{\partial L(\mathbf{Y}, \mathbf{f}(\mathbf{X};\boldsymbol{\theta}))}{\partial z_j^{(i)}} , \tag{2.7}$$

and the derivatives with respect to weights are

$$\frac{\partial L(\mathbf{Y}, \mathbf{f}(\mathbf{X};\boldsymbol{\theta}))}{\partial w_{jk}^{(i)}} = \frac{\partial L(\mathbf{Y}, \mathbf{f}(\mathbf{X};\boldsymbol{\theta}))}{\partial z_j^{(i)}} \frac{\partial z_j^{(i)}}{\partial w_{jk}^{(i)}} = \frac{\partial L(\mathbf{Y}, \mathbf{f}(\mathbf{X};\boldsymbol{\theta}))}{\partial z_j^{(i)}} x_k^{(i-1)} . \tag{2.8}$$

Putting these three equations together, the backpropagation algorithm works by first feeding forward through the network, starting with $\mathbf{x}^{(1)}$ and calculating all subsequent $\mathbf{z}^{(i)}$ and $\mathbf{x}^{(i+1)}$. Also, the very last layer's derivative must be calculated. From here, propagate the layers' derivatives backwards using equation 2.6, therefore determining the desired derivatives via equations 2.7 and 2.8.

2.3 Reinforcement Learning

Reinforcement learning tackles the problem of sequential decision making within a Markov decision process (MDP) framework [40, 41]. An MDP may be described by five key compon-

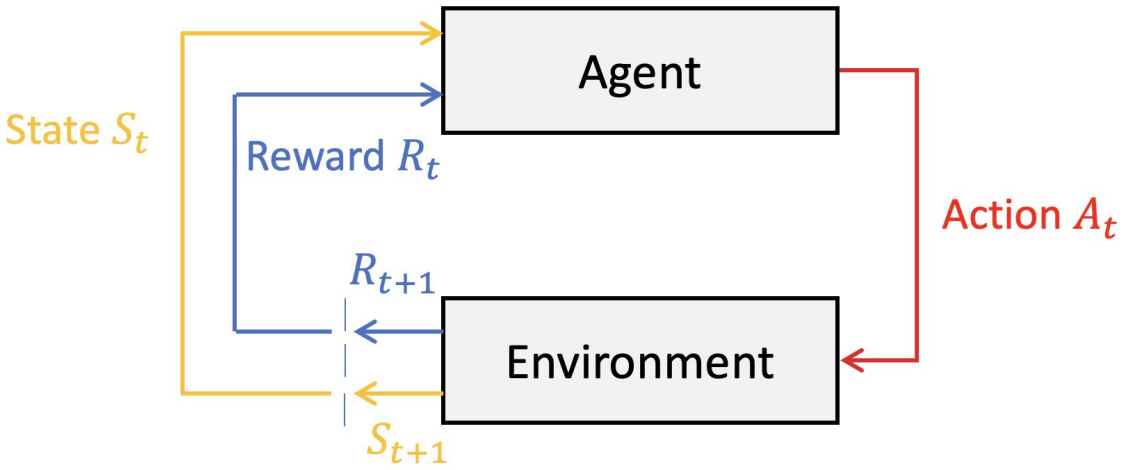

Figure 2.2: Diagrammatic representation of an RL agent operating in an MDP framework.

ents: an agent, an environment, a state, an action, and a reward. The agent is the artificially intelligent entity responsible for making decisions. The environment represents the external system or context in which the agent operates. The set of quantitative representations of the environment, denoted $\mathbf{S}$, are known as states, such that the current situation or configuration of the environment at a given time t is the state $S_t \in \mathbf{S}$. In response to a given state, the agent may choose an action $A_t \in \mathbf{A}$, where $\mathbf{A}$ is the set of all possible actions. After taking action A_t, the environment is represented by state $S_{t+1} \in \mathbf{S}$. For choosing action A_t in response to state S_t, the agent is given a reward $R_{t+1} \in \mathbf{R}$ where $\mathbf{R}$ is the set of possible rewards. The tuple comprising the current state, the chosen action, and the resulting state and reward make up what is known as an experience $e_t = (S_t, A_t, S_{t+1}, R_{t+1})$. This typical RL sequence is depicted in figure 2.2.

Generally speaking, in an MDP, an action A_t taken influences not just the resulting reward R_{t+1} but also the subsequent state S_{t+1}. A fully deterministic, pedagogical example is a chess playing agent taking an action to move a piece, therefore changing the state of the board. More generally, where the environmental state changes in ways both related and unrelated to the actions being taken, the transition to a given state and receiving a given reward has some conditional probability distribution $p(S_{t+1}, R_{t+1}|S_t, A_t)$. An intelligent agent will not just attempt to maximize the next reward, but the long term cumulative reward G_t. To avoid dealing with diverging series, RL agents often are made to maximize a discounted cumulative reward

$$G_t = \sum_j \gamma^j R_{t+1+j} \,, \tag{2.9}$$

where γ is a hyperparameter known as the discount factor.

Situations where the environment is full decoupled from the actions taken are known as contextual bandits [42]. In such cases

$$p(S_{t+1}, R_{t+1}|S_t, A_t) = p(S_{t+1}|S_t)p(R_{t+1}|S_t, A_t)\,. \tag{2.10}$$

Furthermore, agents need not worry about maximizing cumulative rewards, which is equivalent to stating $\gamma = 0$. An agent's experience tuple is now sufficiently described by $e_t = (S_t, A_t, R_{t+1})$.

2.3.1 Policies and Actor-Critics

An agent's action taking strategy may be formalized as a policy. There are many different algorithms and frameworks which inform how the policy is learned. One common family of approaches to RL is to directly optimize the policy. These approaches are known as policy gradient methods. Instead of estimating the value of taking specific actions in certain states (as in so-called value-based methods), policy gradient methods focus on learning a parameterized policy that maximizes the expected cumulative reward. Polices are often written as conditional probabilities $\pi(A_t|S_t, \boldsymbol{\theta})$, which should be understood to mean the probability that taking action A_t is optimal, given current state S_t and a policy parameterized by $\boldsymbol{\theta}$.

Policies can be defined in various ways, but as long as it is possible to calculate gradients with respect to their parameters, we can use gradient descent methods to optimize a performance metric denoted as $J(\boldsymbol{\theta})$. This flexibility allows us to use powerful function approximators, such as neural networks, to represent policies. Neural networks can naturally handle continuous action spaces and even learn stochastic policies, which involve making probabilistic decisions. This stochasticity is essential for balancing exploration and exploitation in reinforcement learning tasks.

In some policy gradient methods, the agent aims to not only learn a policy but also estimate the expected cumulative reward that the policy generates, known as the value function. This leads us to the concept of actor-critic methods. Actor-critic architectures involve two components: an actor, responsible for learning and improving the policy, and a critic, responsible for estimating the value function. The value function, denoted as $v_\pi(S_t)$, represents the expected cumulative reward when starting in state S_t and following policy π. It is formally defined by the equation:

$$v_\pi(S_t) = \mathbb{E}_\pi[G_t|S_t] \tag{2.11}$$

where $\mathbb{E}_\pi[\cdot]$ is the expected value following a policy π.

Learning the parameters of the policy involves iterative updates, typically using a gradient descent optimization approach, with the goal of maximizing the value function as a measure of performance. The policy gradient theorem provides a way to calculate the gradients of the performance metric with respect to the policy parameters [43]. This result simplifies the

update rule for the policy parameters, making it computationally feasible. A detailed proof of the policy gradient theorem may be found in [41], but the result conventionally states that

$$\nabla_\theta J(\boldsymbol{\theta}_t) = \mathbb{E}_\pi[G_t \nabla_\theta \ln(\pi(A_t|S_t, \boldsymbol{\theta}_t))] \,. \tag{2.12}$$

There are many algorithms that use this powerful result. For actor-critic methods, we may use our critic's value function to provide feedback on the expected cumulative reward G_t in equation 2.12, as the value function is being learned simultaneously with the policy anyway. The value function estimate, i.e. the critic, we may notate as $\tilde{v}_\pi(S_t, \boldsymbol{\phi}_t)$, where $\boldsymbol{\phi}$ are the critic learnable parameters. The critic is intuitively updated as

$$\boldsymbol{\phi}_{t+1} = \boldsymbol{\phi}_t + \nu_t \delta_t \nabla_\phi \tilde{v}_\pi(S_t, \boldsymbol{\phi}_t) \,, \tag{2.13}$$

where ν_t is the critic's learning rate, and

$$\delta_t = R_{t+1} + \gamma \tilde{v}_\pi(S_{t+1}, \boldsymbol{\phi}_t) - \tilde{v}_\pi(S_t, \boldsymbol{\phi}_t) \,. \tag{2.14}$$

δ_t is known as the temporal difference error, and it essentially measures the difference between the predicted value (based on the agent's current knowledge) and the actual received reward, so we therefore update the critic's weights depending on how off it's prediction was, and in what direction. The actor is similarly therefore updated as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta_t \delta_t \nabla_\theta \ln(\pi(A_t|S_t, \boldsymbol{\theta}_t)) \,, \tag{2.15}$$

changing the policy to make a given action more or less probable depending if outcomes were better or worse than expected.

2.4 Optimization Algorithms

When optimizing the weights and biases of neural networks, whether for supervised learning tasks or for reinforcement learning purposes, the go-to optimization method is gradient descent. More complex optimization techniques exist, but they often become computationally infeasible when dealing with the substantial size and complexity of neural networks used in modern problems. Gradient descent, with its simplicity and efficiency, remains realistically the most reliable choice for efficiently updating network parameters and achieving convergence in training [17].

Once a neural network is trained, another area of interest arises: finding the inputs that either maximize or minimize the network's outputs, as undertaken in this thesis work. For these types of problems, the dimension of optimization is significantly lower, typically on the order of 10, as opposed to the tens of thousands to millions often encountered during network

training. In such cases, more sophisticated and faster optimization methods come into play. In this work, we highlight two methods used to find optimal inputs for our neural network models: the Nelder–Mead method [44, 45] and Bayesian optimization [46, 47]

2.4.1 Nelder–Mead

The Nelder–Mead method, sometimes called the downhill simplex method, is a versatile and intuitive optimization technique. It is particularly well–suited for scenarios where the dimension of the optimization space is relatively small, making it an ideal choice for finding optimal inputs to neural network models in various applications.

The Nelder–Mead method operates by forming a simplex, which is a geometric shape consisting of $d + 1$ vertices, d being the input dimension of the function being optimized. It is a triangle in two dimensions, tetrahedron in three dimensions, and so on, within the optimization space. This simplex evolves iteratively by modifying its vertices in response to function evaluations. The key idea is to adapt the shape of the simplex based on the function values at its vertices, ultimately converging towards the optimal solution.

The algorithm starts by building the simplex with $d+1$ randomly chosen points $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{d+1})$. The function is evaulauted at each simplex point, and the resulting tuple $(f(\mathbf{x}_1), f(\mathbf{x}_2), ..., f(\mathbf{x}_{d+1}))$ is sorted from largest to smallest for maximization problems, or visa-versa for minimization. The worst point $\mathbf{x}_w$ is taken and "reflected" through the centroid $\mathbf{C}$ connecting the remaining better points, computed as

$$\mathbf{C} = \frac{1}{d} \sum_{i \neq w} \mathbf{x}_i \, . \tag{2.16}$$

The reflected point has coordinate

$$\mathbf{x}_r = \mathbf{C} + \alpha(\mathbf{C} - \mathbf{x}_w) \, , \tag{2.17}$$

where α is a hyperparameter of the optimization.

The function is then evaluated at the reflected point, and if it is better than at least one of function evaluations at the original vertices, but not better than every single one the original vertices, then the reflected point is accepted and becomes the newest vertex.

If the reflected point is better than all of the original vertices, then the algorithm knows it is going in the correct direction, and the reflected point is "expanded" further to coordinate

$$\mathbf{x}_e = \mathbf{C} + \gamma(\mathbf{C} - \mathbf{x}_w) \, , \tag{2.18}$$

where $\gamma > \alpha$. If this expanded reflection beats the original reflection, then the expanded coordinate $\mathbf{x}_e$ is taken as the new vertex, otherwise $\mathbf{x}_r$ is taken.

If though that the reflected point is not better than a single one of the original vertices,

then two "contracted" reflections are tried, given by coordinates

$$\mathbf{x}_{ci} = \mathbf{C} + \beta(\mathbf{C} - \mathbf{x}_w) \tag{2.19}$$

$$\mathbf{x}_{co} = \mathbf{C} + \beta(\mathbf{C} - \mathbf{x}_r)\,. \tag{2.20}$$

$\beta < \alpha$ is chosen such that $\mathbf{x}_{ci}$ lies inside the original simplex, while $\mathbf{x}_{co}$ lies outside it. If the function at either of these contracted points beats $f(\mathbf{x}_w)$, the best performing one is chosen as the new vertex. If both contracted points fail to beat our worst performer, then the simplex is "shrunk" inwards towards the best original vertex $\mathbf{x}_b$, so all points (aside from $\mathbf{x}_b$) become

$$\mathbf{x}'_j = \mathbf{x}_b - \delta(\mathbf{x}_j - \mathbf{x}_b) \tag{2.21}$$

where δ is another hyperparameter.

After one of the operations (reflect, expand, inside/outside contract, or shrink) is applied, the process repeats with the new simplex until a convergence criterion is sufficiently achieved.

One of the Nelder–Mead method's strengths is its simplicity and ability to handle non-smooth and non-convex objective functions. It is particularly useful for solving optimization problems where the objective function is not easily differentiable or lacks a closed-form expression. This method is well-suited for black-box optimization scenarios like finding the inputs to a neural network that maximize or minimize its output.

2.4.2 Bayesian Optimization

Bayesian optimization is a global optimization method within a class of methods known as surrogate methods. In this family of optimization techniques, surrogate models are constructed to approximate the true objective function, allowing for optimization in situations where direct evaluations of the objective function may be computationally expensive.

The distinctive feature of Bayesian optimization lies in its approach to constructing a probabilistic surrogate model, based in Bayesian statistics. This surrogate model is designed to capture and quantify the uncertainty associated with the objective function. Among the various surrogate models employed in Bayesian optimization, Gaussian process regressors [48] are prevalent, and we will focus on them here. Note however that other surrogates such as random forests [17] can be used within the Bayesian optimization framework.

The idea behind Bayesian optimization with Gaussian process modeling is to first assume that a set of evaluations of a function at n different input points, $f(\mathbf{x}_{1:n}) = [f(\mathbf{x}_1), f(\mathbf{x}_2), ..., f(\mathbf{x}_n)]$ were drawn from a multrivariate Gaussian distribution

$$f(\mathbf{x}_{1:n}) \sim \text{Normal}(\mu_0(\mathbf{x}_{1:n}), \Sigma_0(\mathbf{x}_{1:n}, \mathbf{x}_{1:n}))\,, \tag{2.22}$$

where $\mu_0(\mathbf{x}_{1:n})$ and $\Sigma_0(\mathbf{x}_{1:n}, \mathbf{x}_{1:n})$ are the mean and covariance functions defining the Gaus-

sian. This assumed distribution is known as the prior.

Given some new point of interest $\mathbf{x}$, if we would like to know the conditional probability that the unknown function takes a certain value at $\mathbf{x}$, given our previous n observations making up the prior, we may use Bayes' theorem. For a detailed computation of said conditional distribution using Bayes' theorem, one may consult [16], but in short

$$f(\mathbf{x})|f(\mathbf{x}_{1:n}) \sim \text{Normal}(\mu_n(\mathbf{x}), \sigma_n(\mathbf{x})) \tag{2.23}$$

where

$$\mu_n(\mathbf{x}) = \Sigma_0(\mathbf{x}, \mathbf{x_{1:n}})\Sigma_0^{-1}(\mathbf{x}_{1:n}, \mathbf{x}_{1:n})(f(\mathbf{x}_{1:n}) - \mu_0(\mathbf{x}_{1:n})) + \mu_0(\mathbf{x}) \tag{2.24}$$

and

$$\sigma_n(\mathbf{x}) = \Sigma_0(\mathbf{x}, \mathbf{x}) - \Sigma_0(\mathbf{x}, \mathbf{x}_{1:n})\Sigma_0^{-1}(\mathbf{x}_{1:n}, \mathbf{x}_{1:n})\Sigma_0(\mathbf{x}_{1:n}, \mathbf{x}) \,. \tag{2.25}$$

This distribution-defining mean and variance is thus directly computable with matrix inverting algorithms. The distribution is known as the posterior distribution, and is in essence a probability distribution over function values at the new point of interest $\mathbf{x}$ given the prior observations. Exactly which functions are likely to be sampled is controlled by the form of the covariance function. A standard choice of covariance function for Gaussian process modeling is the radial basis function

$$\Sigma_0(\mathbf{x}_i, \mathbf{x}_j) = \exp\{(||\mathbf{x}_j - \mathbf{x}_i||^2)\} \,, \tag{2.26}$$

where additional hyperparameters may be inserted as a pre-factor or into the argument of the exponential to modify the scale over which function is varying or the rate of frequencies of oscillation.

With the posterior distribution computed, one may be tempted to numerically find the input $\mathbf{x}^*$ that minimizes (or maximizes) $\mu_n(\mathbf{x})$, but such a strategy fails to account that different values of the learned surrogate $f(\mathbf{x})|f(\mathbf{x}_{1:n})$ are more certain than others, and therefore it may be helpful to sample regions of high uncertainty to update the posterior distribution. This trade-off between exploration (sampling in uncertain regions) and exploitation (sampling in regions where the objective function is expected to be optimal) is tackled by acquisition functions, which provide a criterion for selecting the next point to evaluate the objective function. Expected Improvement (EI) is a commonly used acquisition function. It quantifies the expected gain in objective function value compared to the current best observed value f_n^*, i.e.

$$\text{EI}(\mathbf{x}) = \mathbb{E}[\max((f(\mathbf{x}) - f_n^*), 0)] \tag{2.27}$$

In other words, it measures how much improvement one can expect by evaluating the objective function at a specific point. EI can be evaluated in closed form (see [49]), and unlike the objective function, it is generally inexpensive to compute. Therefore, it can be maximized quickly at

every iteration of Bayesian optimization, even with methods that require a significant number of function calls and higher-order gradient calculations, such as Newton's method.

The main limitation of Gaussian process Bayesian optimization is that the matrix inversions become costly with both larger input dimensions, and many sampled points. Typically Bayesian optimization is intended for input dimensions less than 20, and still the computational complexity of the Gaussian process regressor scales cubically with the number of points used to update the posterior [50, 51]. Furthermore, while Gaussian processes are robust to noise on the sampled function evaluations, the noise is assumed normally distributed, and therefore noise with some dynamic structure (referred to as heteroscedasticity) causes the underlying assumptions of a Gaussian process to fail.

Chapter 3

Background Considerations: Atom Cooling

This chapter seeks to introduce the pertinent theory behind the atom cooling techniques used in this work. In particular, the operating principles of a magneto-optical trap (MOT) are described. Following this, sub-Doppler cooling by motion induced orientation cooling is then described as a means to cool below the limits of a MOT. Lastly, pure magnetic trapping is described. In our laboratory's typical ultracold atom preparation sequence, the magnetic trap would be followed by subsequent cooling techniques such as evaporation by radio frequencies and optical dipole trapping, but our machine learning agents were trained to optimize up to and including a high-field gradient magnetic trap. This initial preparation sequence is crucial for successfully cooling ensembles to (BEC).

3.1 Magneto-Optical Trapping

Magneto optical trapping is an integral tool for atomic cooling procedures, often being the first stage in cooling sequences. MOTs can cool and confine atoms that are initially at temperatures higher than room temperature. Conceptially, MOTs may be broken down into two components, a velocity depended force via optical molasses, and a position depending force via magnetic field gradients.

3.1.1 Optical Molasses

The cooling mechanism of a MOT is optical molasses, alternatively known as Doppler cooling. Let us consider a cloud of two-level atoms, with a Maxwell-Boltzmann [52] distribution of velocities, placed inside of a laser beam red detuned from resonance. The beam is not significantly absorbed by the atoms at rest, but the atoms moving towards the beam Doppler shift their transition to be resonant with the beam. The excited atoms will spontaneously emit isoptropically, but are only being excited from one direction. Therefore they are experiencing a

net force, given by

$$\mathbf{F} = \hbar \mathbf{k} R \, . \tag{3.1}$$

R is known as the scattering rate, and is given by

$$R = \Gamma \rho_{22} \, , \tag{3.2}$$

where Γ is the natural line width, and ρ_{22} is the density matrix element representing the fraction of the population in the excited level. We may write the scattering force as a function of beam intensity I [52]

$$\mathbf{F} = \hbar \mathbf{k} \frac{\Gamma I}{2 I_{\text{sat}}} \frac{1}{1 + I/I_{\text{sat}} + 4(\delta/\Gamma)^2} \, , \tag{3.3}$$

where saturation intensity is $I_{\text{sat}} = \pi h c \Gamma / (3\lambda^3)$, and detuning $\delta = \omega - \omega_0 + kv$ is the difference between laser frequency ω and the atomic resonance frequency ω_0, plus the Doppler shift from velocity v.

If the atoms are in two beams counterpropogating each other, the total damping force is

$$\mathbf{F}_{\text{molasses}} = -\alpha \mathbf{v} \, , \tag{3.4}$$

which is linear in velocity and is therefore analogous to a frictional force. The coefficient α is given as

$$\alpha = 4\hbar k^2 \frac{I}{I_{\text{sat}}} \frac{-2\delta/\Gamma}{(1 + (2\delta/\Gamma)^2)^2} \, . \tag{3.5}$$

It's crucial to note that the coefficient α must be positive to induce the effect of a damping force in optical molasses. This requirement means a detuning $\delta < 0$, i.e. a red detuning.

While the spontaneous emission is isotropic and these momentum kicks are on average zero, these recoils are essentially a random walk of the velocity, and thus the final ensemble due to the last random kick is thermal. Furthermore, the effect of the random walk heating is cumulative, so to decelerate atoms along three spatial dimensions with a magneto-optical traps consisting of three pairs of counterpropogating beams, the lowest temperature molasses cooling can expect to achieve can be shown to be [53]

$$T_D = \frac{\hbar \Gamma}{2 k_B} \, . \tag{3.6}$$

3.1.2 Magnetic Field Gradient

The addition of a magnetic field gradient to an optical molasses set up, along with correct choices of beam polarization, adds a spatial trapping capability to the cooling apparatus. For a field magnitude that is minimum at the centre of the trap and larger in all directions, a pair of coils with currents circulating in opposite directions can produce such a quadrupole field, such as the anti-Helmholtz configuration used here. The field magnitude increases linearly in

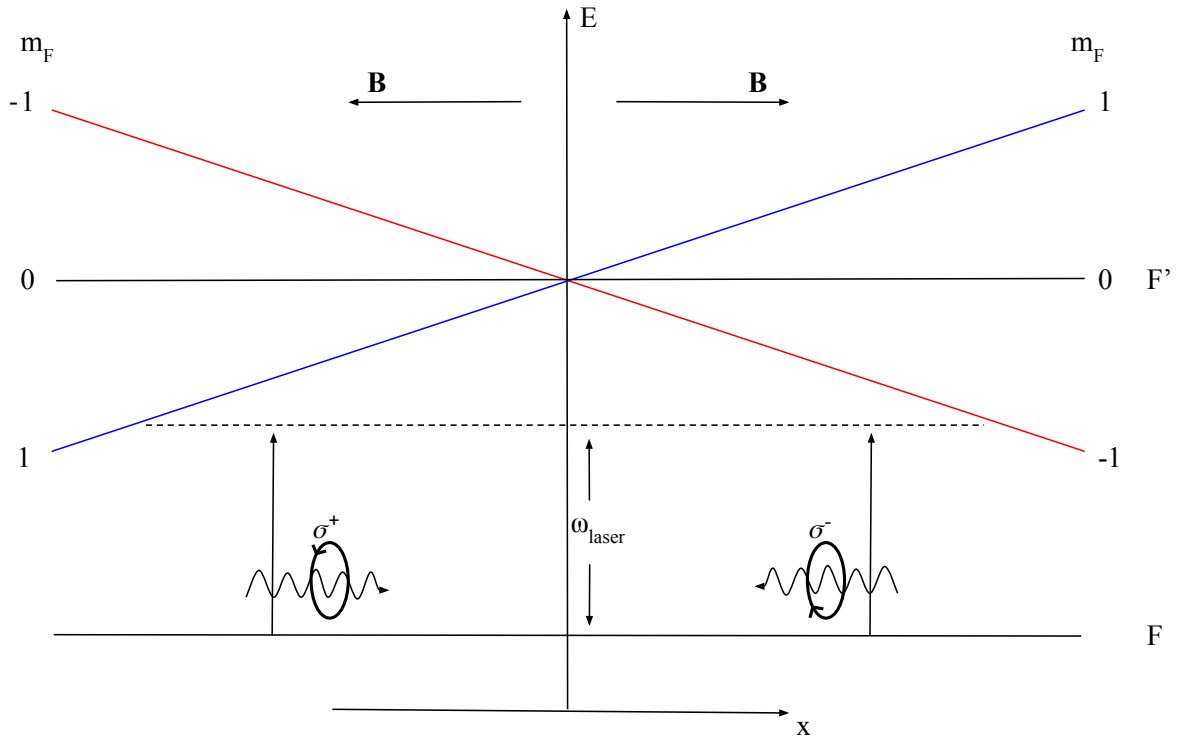

Figure 3.1: Adapted from [52]. A conceptual diagram of the mechanism behind magneto-optical trapping. The magnetic field Zeeman shifts moving atoms' transitions, addressable by circularly polarized light, into resonance.

every direction for small displacements about the centre point.

The principle behind the addition of the field gradient, pictorially shown in figure 3.1, is to add a position dependant Zeeman effect. For an atom in a state with total angular momentum F in positions away from the centre, the states described by the angular momentum component along the field axis m_F split in energy. The further from the centre an atom is located, the larger the magnetic field, so the larger the Zeeman shift of the magnetic sub-levels. Restricting our thought to a gradient along a single axis centred at $x = 0$, an atom's m_F levels in the region $x > 0$ split such that those with negative m_F values will become lower in energy and thus a transition changing m_F by -1 moves closer to resonance for a beam red detuned from the manifold of states. Therefore a beam polarized σ^- (counter clockwise with respect to the direction of the spatial axis $\mathbf{e}_x$) will excite the $\Delta m_F = -1$ transition that has been shifted into resonance. For $x < 0$, the $\Delta m_F = +1$ transition is brought closer to resonance for a red detuned beam, and may thus be excited with a σ^+ transition.

The detuning in equation 3.3 is now not only modified by kv, but also by βx, where

$$\beta x \propto \frac{\partial B}{\partial x} x \,. \tag{3.7}$$

So the force on the atoms is now not only a friction analog, but also has a spatial restoring

force component

$$F_{MOT} = -\alpha v - \frac{\alpha\beta}{k} x \,, \tag{3.8}$$

making a magneto-optical trap analogous to a damped spring system.

3.2 Sub-Doppler Cooling

As mentioned in the section previous, there is a limit to which temperatures optical molasses can cool. For experiments requiring temperatures much below this limit, such as those involving Bose-Einstein condensates, techniques beyond magneto optical trapping and optical molasses are required. Laser cooling can go below this limit with the use of a spatial gradient in the polarization of counterpropogating laser beams.

There are two types of polarization gradient cooling schemes [53, 54]. One type uses beams with orthogonal linear polarizations, and is sometimes known as Sisyphus cooling. The other type uses circularly polarized beams of opposite chiralities, sometimes referred to as the σ^+–σ^- configuration, or alternatively as "motion induced orientation cooling". The σ^+–σ^- scheme is what is used in this thesis work, and will be the focus of this section.

Consider the two beams counterpropagating on the z-axis, with opposite circular polarizations

$$\mathbf{E}_1 = E_0 \exp(i(kz - \omega t))(\mathbf{e}_x + i\mathbf{e}_y) \tag{3.9}$$

$$\mathbf{E}_2 = E_0 \exp(i(-kz - \omega t))(\mathbf{e}_x - i\mathbf{e}_y) \tag{3.10}$$

The total field field is the summation of the two beams

$$\mathbf{E} = 2E_0 \sin(\omega t)(\cos(kz)\mathbf{e}_x - \sin(kz)\mathbf{e}_y) \,. \tag{3.11}$$

Notice the x and y components are in phase, however the relative amplitudes rotate with kz. This describes a linear polarization with a direction that rotates as the z-axis is traversed.

For the propose of instruction, let us consider a simple atomic transition for this cooling phenomenon: $F = 1 \rightarrow F' = 2$. Two key effects cause the atoms to cool while traversing this rotating linear polarization. The first is the simple observation that for atoms in the ground state $|F = 1, m_F = -1\rangle$, the Clebsch–Gordan coefficient coupling to $|F' = 2, m_F = -2\rangle$ (a σ^- transition) is unity, whereas the Clebsch–Gordan coefficient coupling to $|F' = 2, m_F = 0\rangle$ (a σ^+ transition) is $1/\sqrt{6}$. Likewise, for $|F = 1, m_F = +1\rangle$, the Clebsch–Gordan coefficient associated with a σ^+ transition is $\sqrt{6}$ times larger than that for σ^-. Note here that these states mentioned are eigenstates of the F_z operator (i.e. the direction of light propagation). See figure 3.2 for the relevant $F = 1 \rightarrow F' = 2$ Clebsh-Gordan coefficients .

The second effect is the eponymous motion-induced atomic orientation. In short, atoms moving along the field towards the σ^- beam are pumped preferentially into the $|F = 1, m_F = -1\rangle$

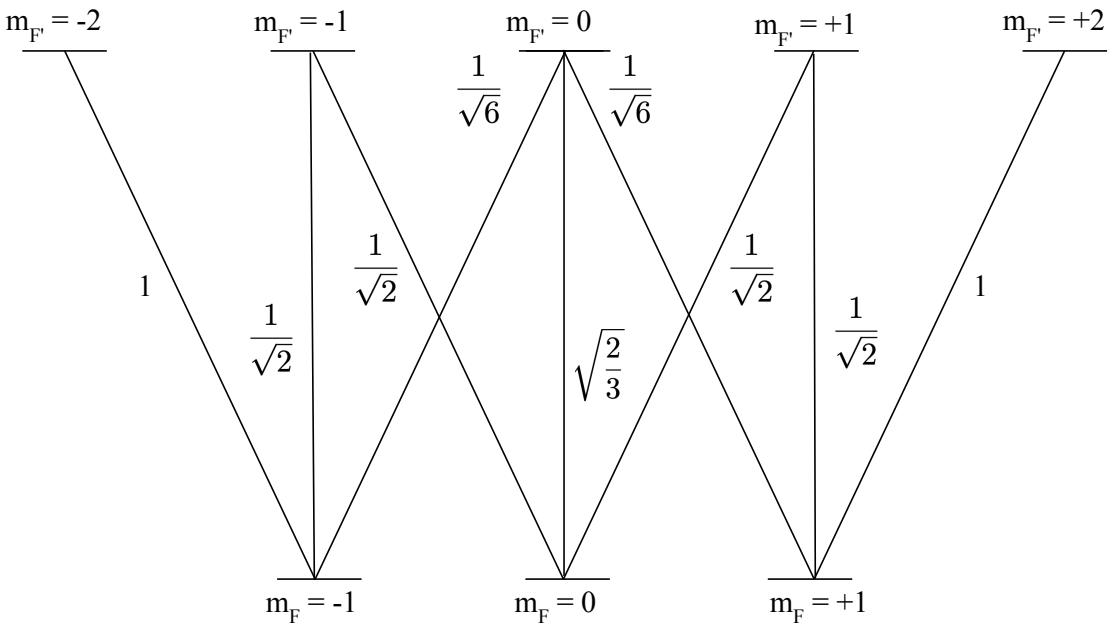

Figure 3.2: Atom energy levels with their respective Clebsch–Gordan coefficients, for transitions between $F = 1$ and $F' = 2$ manifolds.

state, and atoms moving towards the σ^- beam are pumped preferentially into the $|F = 1, m_F = +1\rangle$ state. The mechanism behind this effect is outlined well in appendix A of [54], but we can summarize it as follows. For an atom at rest, linear polarizations (let's say along $\mathbf{e}_y$) driving π transitions will concentrate atoms in the $|F = 1, m_F = 0\rangle_y$ state. Referring again to figure 3.2, we can see this is because the pumping rates are proportional to $(1/\sqrt{2})^2(1/\sqrt{2})^2 = 1/4$ and $(\sqrt{2/3})^2(1/\sqrt{6})^2 = 1/9$ for $|F = 1, m_F = \pm 1\rangle_y \rightarrow |F = 1, m_F = 0\rangle_y$ and $|F = 1, m_F = 0\rangle_y \rightarrow |F = 1, m_F = \pm 1\rangle_y$ respectively. Note here that states $|F, m_F\rangle_y$ are intended to represent eigenstates of the F_y operator. For an atom moving with velocity v, the quantization axis defining linear polarization it sees is rotating with $\phi = -kvt$. Say at time t, the quantization axis now lies along $\mathbf{e}_{\bar{y}}$. For weak pumping, the atoms' traversal of the field is non-adiabatic such that to describe the state in the eigenbasis of the new quantization axis's operator $\bar{F}_{\bar{y}}$, it is a superposition where the ratio of the components' amplitudes is proportional to the velocity of transversal v. Calculating the expectation value of F_z using these superposition states reveals a non zero $\langle F_z \rangle$, in other words, a motion induced orientation.

Since atoms moving towards the σ^- beam and away from the the σ^+ beam are oriented in the $|F = 1, m_F = -1\rangle$ state, they are more likely to absorb from the counter propagating beam, thus slowing the atoms down. Likewise, those moving towards σ^+ and away from σ^- are oriented in the $|F = 1, m_F = -1\rangle$ state but absorb more from the counterpropogating, therefore are slowed in the other direction. Applying such a scheme with three non-parallel beam pairs will provide a frictional force along all spatial dimensions. Note however that as the atoms cool and their average speed decreases, the cooling transition between the hyperfine manifolds will become Doppler shifted out of resonance, and thus a continual decrease of the

beams' red detuning is required to address a range of velocities.

3.3 Magnetic Trapping

Like with Doppler cooling, sub-Doppler cooling has a limit on how low it can make temperatures [54]. Cooling below the sub-Doppler limit requires techniques beyond absorption based cooling. These techniques, known as evaporative cooling, generally involve holding the atoms in a trap and removing the warm tail of the Maxwell-Boltzmann distribution [53]. While the final stage of BEC production often involves making an optical dipole trap [55] shallower so that the warmest atoms fall out, temperatures following sub-Doppler cooling are too high to load atoms in such small and shallow optical dipole traps. First, magnetic traps are used, from which radio frequencies drive the warmest atoms into untrappable states. In this thesis work demonstrating reinforcement learning control of atom cooling, the final step we control is a magnetic trap.

The central idea behind magnetic trapping uses atoms in a spin-polarized state that have lower energy for smaller magnetic field magnitudes. A magnetic field gradient will thus provide a potential well, where these low-field seeking spin-polarized atoms prefer to sit at the field minimum. Like with magneto-optical trapping, anti-Helmholtz coil pairs provide a quadrupole trap, whose magnetic field may be derived analytically using elliptical integrals [56, 57] . Near the origin, the field has magnitude given by

$$B \propto \sqrt{x^2 + y^2 + 4z^2}\,. \tag{3.12}$$

The force acting on an atom with magnetic dipole moment μ, along a one dimensional slice of the gradient is

$$F = -\frac{\partial(\boldsymbol{\mu} \cdot \mathbf{B})}{\partial z} \propto -g_F m_F \frac{\partial B}{\partial z}\,, \tag{3.13}$$

so whether or not the force attracts or repels atoms from the trap centre at $z = 0$ depends on the sign of the $g_F m_F$, the product of the magnetic quantum number and the total atomic angular momentum g-factor [52, 57]. For the hyperfine manifold $F = 2$, g_F is positive, so atoms pumped into $m_F > 0$ are low field seeking, and $m_F = +2$ constrains the atoms in as tight a trap as possible.

Chapter 4

Experimental Design

This chapter outlines the practical aspects of the experimental system used in this thesis. In Section 4.1, we break down the cooling sequence, detailing the apparatus and explaining how our agents interact with it to modify the cooling process. At the core of this work is the optimization of atom numbers resulting from the cooling sequence. Section 4.2 explains how, after each cycle, we estimate the atom count, providing crucial feedback for our agents to adjust their policies. For the agents to determine which controllable parameters of the cooling sequence to modify, we collect a representation of the environmental state. Section 4.3 explains the monitored parameters, the collection process, and how they are automatically processed for use in the agents' policies.

4.1 Atom Cooling Apparatus

We allow both the ML agents to control 30 parameters in our cooling sequence. The ML agents interface with the apparatus via an analog board and radiofrequency (RF) signal generator. For analog signals, a BNC2110 National Instrument Data Aqusition board is used, and for RF signals a Novatech 409B Direct Digital Synthesized Signal Generator is used. These 30 parameters are listed in Table 4.1, and identified by the letter "A" and a number. A schematic of the integration into the system of these agent controllable parameters, as well as the monitored environmental parameters (section 4.3), is shown in figure 4.1. In this section, we describe in detail the cooling sequence, and how the ML agents are given control of the various aspects of our atom cooling apparatus.

4.1.1 Cooling Sequence Overview

The ^{87}Rb atom-cooling process begins by capturing and cooling the atoms using two simultaneously applied magneto-optical traps (MOTs). This stage takes a total of 14 seconds. The main cooling transition we use in our MOTs is the D_2 $|5^2S_{1/2}, F = 2\rangle \rightarrow |5^2P_{3/2}, F' = 3\rangle$ trans-

Table 4.1: Agent controllable parameters

Name	Number	Stage	Implementation
2D coil 1	A1	MOT	current to 2D coil 1
2D coil 2	A2	MOT	current to 2D coil 2
2D coil 3	A3	MOT	current to 2D coil 3
2D coil 4	A4	MOT	current to 2D coil 4
3D x-bias	A5	MOT	current to x-axis bias coils
	A6	Sub-Doppler	
	A7	Optical pump	
	A8	Magnetic trap 1	
	A9	Magnetic trap 2	
3D y-bias	A10	MOT	current to y-axis bias coils
	A11	Sub-Doppler	
	A12	Optical pump	
	A13	Magnetic trap 1	
	A14	Magnetic trap 2	
	A15	Imaging	
3D z-bias	A16	MOT	current to z-axis bias coils
	A17	Sub-Doppler	
	A18	Optical pump	
	A19	Magnetic trap 1	
	A20	Magnetic trap 2	
2D repump	A21	MOT	laser power, via AOM
2D cooling	A22	MOT	laser power, via AOM
2D push beam	A23	MOT	laser power, via AOM
3D frequency	A24	MOT	laser frequency
3D repump	A25	MOT	laser power, via AOM
3D cooling	A26	MOT	laser power, via AOM
SD start freq.	A27	Sub-Doppler	laser frequency
SD end & OP freq.	A28	Sub-Doppler	laser frequency
OP power	A29	Optical pump	laser power, via AOM
Imaging freq.	A30	Imaging	laser frequency

ition. Figure 4.2 presents a diagram summarizing the transitions used in this work. There is a small yet finite probability of off-resonant excitation to the excited $F' = 2$ state, from which decay to the $F = 1$ ground state is a permitted channel. This would remove atoms from the cooling cycle if allowed to run long enough. We therefore address the additional transition $\left|5^2S_{1/2}, F = 1\right\rangle \rightarrow \left|5^2P_{3/2}, F' = 2\right\rangle$ to return atoms to the cooling cycle through optical pumping.

As atoms diffuse from a vaporized metallic source, the 2D MOT captures, cools and collimates the atoms along two of the three cardinal directions. These atoms form an atomic beam, which is accelerated using a push beam, which is blue-detuned above the $F = 2 \rightarrow F' = 3$ transition. The push beam frequency is tuned such that slow moving atoms that would not otherwise contribute to the atomic flux are accelerated. The atoms pass through a differ-

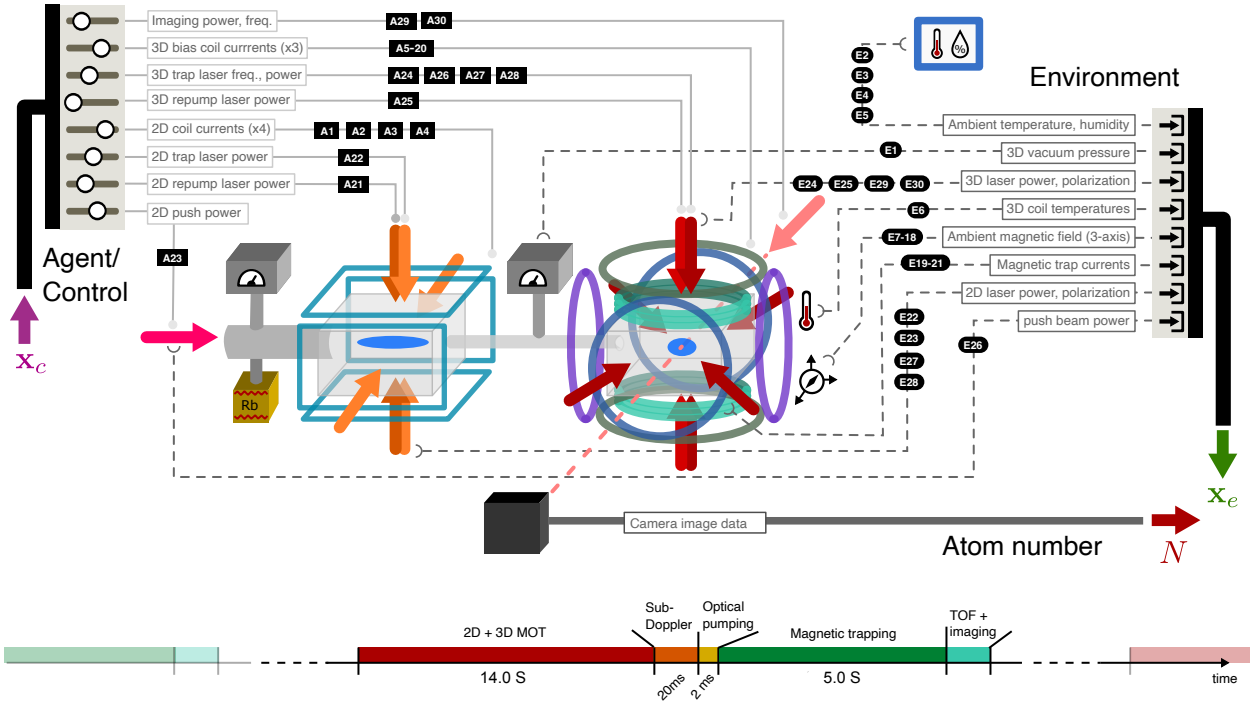

Figure 4.1: System schematic, adapted from our paper [58]. **Top:** A set $\mathbf{x}_c$ of agent-defined parameters (Ai) control the experimental apparatus, while a set $\mathbf{x}_e$ of environmental parameters (Ei) are measured during each experimental sequence. The apparatus consists of an oven-fed 2D-MOT (with agent-controlled coil currents and laser powers), which supplies a 3D-MOT (with agent-controlled laser powers and frequencies) via an agent-controlled push beam. At the 3D-MOT, several cooling steps precede the magnetic trapping of the atoms, agent-controlled bias magnetic field coils are used to optimize the environment throughout the sequence. The atom number N is the reward and determined using data obtained by the camera, with agent-controlled imaging laser power and frequency. Image credits to Dr. Lindsay LeBlanc. **Bottom:** The timing sequence for each experimental cycle, which is repeated after each destructive image of the atoms is recorded. The atoms are held in the high-field "compressed" MT for 4.5 seconds, a comparable time to the radiofrequency evaporation duration used in BEC generation.

ential pumping tube and enter the second, lower-pressure vacuum chamber, where the 3D-MOT is found. Here, the atoms are trapped and cooled in all three dimensions, resulting in the a microkelvin-temperature atomic cloud. The 2D-MOT uses a cylindrical quadrupole field, as well as two pairs of counter-propagating cooling laser beams red-detuned from the $F = 2 \rightarrow F' = 3$ transition. The 3D-MOT consists of a radially symmetric quadrupole trap, and three counterpropagating laser beams red-detuned from the $F = 2 \rightarrow F' = 3$ transition. Each MOT also includes a pair of the counterpropagating repump beams, closing the cooling loops with the $F = 1 \rightarrow F' = 2$ transition.

After the MOT stage, the atomic cloud undergoes 20 ms of polarization-gradient cooling, to cool the atoms below the Doppler limit. To achieve this, the quadrupole magnetic field is removed, and the cooling beam's frequency is linearly swept from a large red detuning to a small red detuning. Before activating the magnetic trap (MT), the atoms are optically pumped into the magnetically trappable Zeeman ground state $|F = 2, m_F = 2\rangle$ using right-circularly

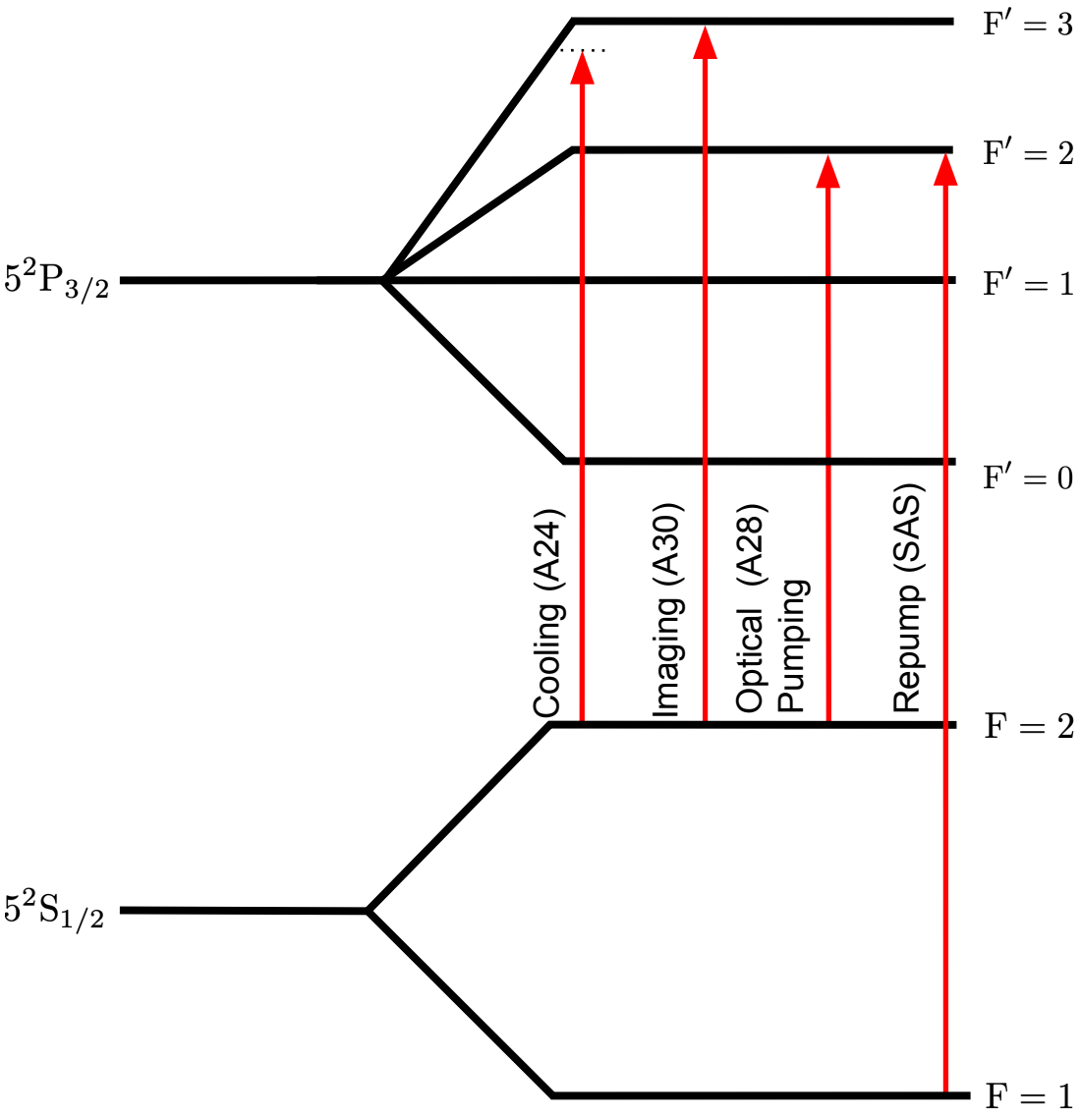

Figure 4.2: Relevant transitions between hyperfine energy levels on the ^{87}Rb D_2 line. The cooling, imaging, and optical pumping frequencies are controllable by our agents, whereas the repump frequency is fixed by locking to the SAS resolved spectral feature, and serves as the reference frequency for beat-note locking.

polarized light resonant with the $F = 2 \rightarrow F' = 2$ transition. Repump light is also employed to close the optical pumping loop.

The magnetic trap is comprised of a quadrupole magnetic field which is designed to confine atoms in the $|F = 2, m_F = 2\rangle$ state against the force of gravity, but is not strong enough to trap atoms in the $|F = 2, m_F = 1\rangle$ state against gravity, due to their smaller magnetic moment. This results in a spin-polarized ensemble. In preparation for subsequent evaporation cooling (which is not carried out in these experiments but is essential for BEC production), the magnetic trap is compressed to increase the atomic density and elastic collision rates. This compression is achieved by ramping the gradient of the magnetic field.

From here, production of BECs would normally proceed by forced evaporation via radio

frequency spin flips of the high-energy atoms. In this investigation, we instead choose this point to release the atoms from the trap and perform a destructive time-of-flight (TOF) image, from which atom number is estimated.

4.1.2 Laser System Control

All of the required laser frequencies used in this work are generated by two lasers, both with approximate central wavelengths of 780 nm to address transitions on the D_2 line. The first, which we refer to as the cooling laser, is the Toptica TAPro, a diode laser amplified by a tapered amplifier. The output of the cooling laser has 1.6 W of power. The second laser, which we call the repump laser, is a MOGLabs CEL external diode laser, with output power 95 mW.

The repump laser is locked directly to the $F = 1 \rightarrow F' = 2$ repump transition using the technique of Doppler-free saturated absorption spectroscopy (SAS) [52, 59]. A small percentage of the laser power is picked off to be sent through a cell of hot rubidium vapour. The relevant hyperfine line is resolved within the spectrum, so that the laser may be fixed to this frequency via proportional–integral–derivative (PID) controller [60]. Here, we allow no room for agent tunability, as the fixed repump laser is used as a reference for all other frequencies chosen by our agents. With the repump laser as the reference laser, and the cooling laser therefore as the secondary laser, in an optical fibre we combine a beam from each laser to generate a beat-note signal. The beat-note has a frequency which is the difference between the two lasers. With a reference signal provided as a control, the beat-note and reference signals are compared via phase-locked loop circuit (EVAL-ADF4007EBZ1), so we may use another PID controller to stabilize the frequency difference between the slave laser and its already-locked master. When we require our agent to have frequency tunability during the different stages of our cooling sequence, we allow it to set the reference signal frequency via our RF signal generator. Specifically, this is how the MOT frequency is tuned (A24), the starting frequency of our sub-Doppler ramp (A27), the ending frequency of the sub-Doppler ramp as well as the optical pumping frequency (A28), and our imaging frequency (A30).

The main outputs of both the cooling and repump lasers are split into multiple arms, so we may cool with both a 2D MOT followed by a 3D MOT, pump atoms into magnetically trappable states, and image the resulting ensembles. Each arm goes through series of a acousto optical modulators (AOMs) [61], as shown diagrammatically in figure 4.3. The AOM sound wave frequencies and amplitudes provide both frequency and power modulation to our beams. While we use only a constant frequency shift for each AOM (as our agents perform frequency tuning via laser lock-point), we give our agent control over the sound wave amplitudes and therefore the various beam powers. We allow the agent control of the 2D MOT repump (A21) and cooling (A22) powers, and the 3D MOT repump (A25) and cooling (A26) powers. Transfer from the 2D- to 3D-MOT can be optimized by our agents changing the acceleration of the atoms via the power of a blue-detuned push-beam (A23). The pumping of atoms into the magnetic-

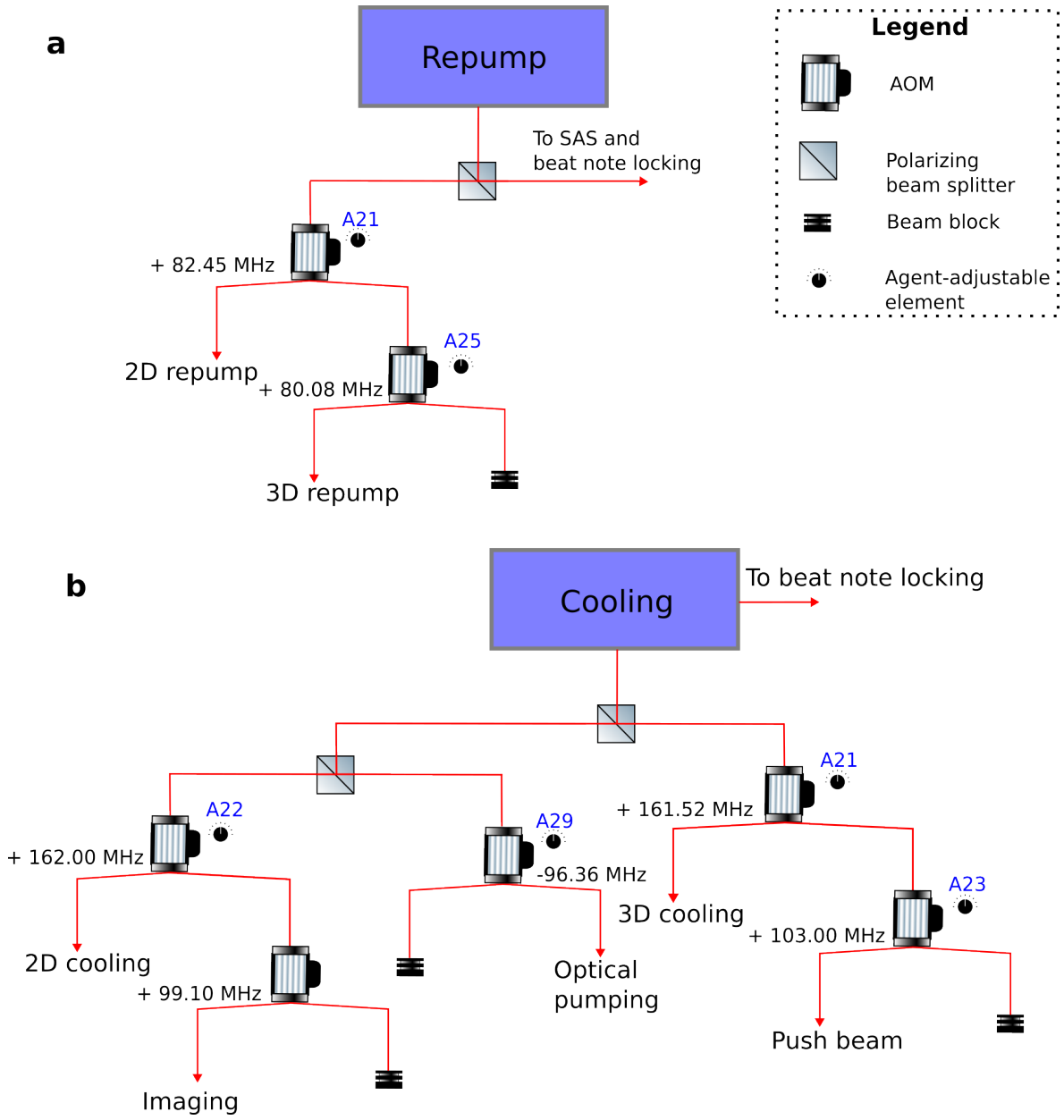

Figure 4.3: Beam control flowchart, for both **a** the repump and **b** the cooling lasers. The agents may control the AOM sound wave amplitude to modulate the power output sent into the various arms (see table 4.1). AOMs also provide a constant frequency shift, and the agents may perform frequency fine-tuning via laser lock-point. Light is picked off of the repump laser to be used for locking via SAS, as well as picked off of both to be used for beat-note locking.

ally trappable $m_F = +2$ state can too be optimized by the agent changing the power of the right-circularly polarized optical pumping beam (A29). Note that during optical pumping, the 3D repump beam is also present to close the pumping cycle, but we keep its power constant.

4.1.3 Magnetic Field Control

To provide the magnetic fields used for our cooling sequence, several sets of coils and power supplies are used.

The 2D MOT operates along two of the three cardinal directions. Four rectangular coils are used to create the necessary two dimensional quadrupole field to facilitate 2D magento-optical trapping. Here, the coils have 25 turns of 12 AWG wire each. To keep the coils from excessively heating, 16°C water circulates through them. Two Agilent 6651A DC power supplies are used for the four coils. To control exactly how much current passes through the individual coils, each coil has an associated MOSFET (IXFN180N25T) placed in series. Our agents therefore may control the current through each 2D MOT coil by setting the MOSFET gate voltages (A1-A4).

To create the fields used for both the 3D MOT and the MT, we use a single pair of coils in an anti-Helmholtz configuration, with a separation of 58 mm. The coils are made of Kapton insulated hollow copper wire, with high pressure water circulated through the cavity inside. Current is supplied by a single Agilent 6690A DC Power Supply, and controlled by a bank of 24 MOSFETS. As with the 2D MOT coils, we could give our agents dynamic control of the gate voltages and therefore the coil current. The issue with this is that we also would like to monitor coil currents as environmental parameters (section 4.3). To keep the environment fully decoupled from the chosen actions so that equation 2.10 holds, we keep the current choices for this coil pair static. During the 3D MOT, we run 28 A, giving a gradient of approximately 10.5 G/cm. Following optical pumping, we run an "uncompressed" MT at 65 A. This MT has a gradient of 27.2 G/cm, which is sufficient to confine atoms in the $|F = 2, m_F = 2\rangle$ state against gravity, while atoms in the $|F = 2, m_F = 1\rangle$ state are not trapped. If we were using our system to its full BEC generating capability, the MT would be followed by a subsequent evaporation cooling stage. To allow successful evaporative cooling, the magnetic trap is compressed. This compression is achieved by linearly ramping the current to 425 A over a duration of 0.1 seconds, making a magnetic field gradient of 176.4 G/cm. With the trap compressed to 425 A, the atomic ensemble is primed for evaporative cooling, but we choose here to be the last step the agent may optimize.

Where the agents are given a lot of influence in the magnetic field control is via three auxiliary coil pairs in Helmholtz configurations surrounding the 3D MOT cell, arranged in each of the 3 orthogonal directions. These coils provide a bias magnetic field that is critical in every step of the cooling sequence. During the 3D MOT, they shift the centre of the quadrupole field such that it matches the six beams' crossing point (A5, A10, A16). During sub-Doppler cooling, the presence of bias fields (A6, A11, A17) becomes crucial to cancel out external fields and facilitate the subsequent transfer of atoms into the magnetic trap. While optical pumping, the y-bias defines the quantization axis for the pumping (A12), and the other axes bias fields cancel out background fields (A12, A18). With the atoms in the MT, bias magnetic fields shift

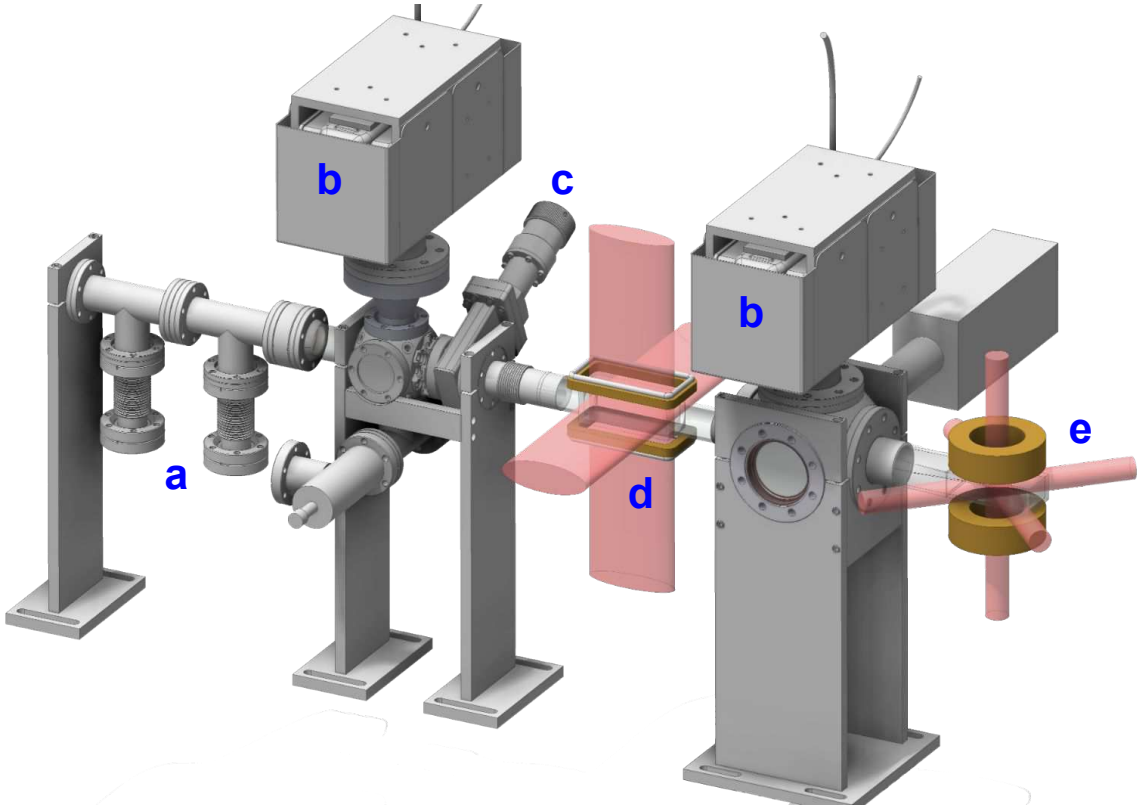

Figure 4.4: Schematic of our vacuum system. Key facets indicated as follows. a - rubidium oven. b - ion pumps. c - gate valve. d - 2D MOT. e - 3D MOT and MT. Image credits to Dr. Lindsay LeBlanc.

the zero point of the trap (A8, A13, A19). When compressing the trap, again the magnetic-gradient zero can be shifted (A9, A14, A20). Finally, when the atoms are dropped from the trap and illuminated with the imaging beam, the imaging quantization axis is defined by a magnetic field created by the y-bias coils (A15).

4.1.4 Vacuum System

The rubidium is kept under ultra-high vacuum for the entirety of the cooling sequence. The vacuum system may conceptually be divided into two sections, separated by gate valve (c in figure 4.4). The first section houses and vaporizes solid ^{87}Rb (a in figure 4.4), while the second section is where the cooling and trapping take place (d,e in figure 4.4).

The oven is a section of the vacuum system containing solid rubidium, wrapped with fibre-glass heat tape. The oven temperature is kept stable at a desired set point via PID controller and measurement by thermocouple. We do not allow the agents control of this set point. The oven takes time to heat/cool to desired set points, so the effect of actions would not be instantaneous. This would add much more complexity to this reinforcement learning problem, therefore we keep the oven temperatures fixed. The main section is held at 45°C, and the push beam window at 50°C to avoid rubidium accumulating and corroding the window seal.

Ultrahigh vacuum is maintained in the system by two ion pumps (Agilent VacIon 55 Starcell), one following the oven and one in between the 2D and 3D MOTs (b in figure 4.4). The agents are not given control of the set vacuum pressures as both pumps act to their full capabilities. Vacuum pressures as low as possible are desired to increases the efficiency of cooling and extend trap lifetimes. Furthermore, control of vacuum pressures would encounter the same issues with control of oven temperatures: pumping to desired pressures takes time and therefore the effect of action taking would not be instantaneous.

4.2 Estimating Atom Number From TOF Images

Our ML agents operate by attempting to maximize the number of atoms present at the end of our cooling sequence. Thus, accurate estimates of the number of atoms following each cycle is necessary.

In our sequence, the final step is compression of the MT, following which atoms are released from the trap and a destructive image is acquired. We allow the atoms a 10 ms TOF before they are illuminated with the $F = 2 \rightarrow F' = 3$ resonant imaging beam. While the agent chooses this resonant frequency via control parameter A30, estimates of the atom number are extracted in the following way.

Three images are recorded by a BFLY-U3-13S2M-CS CCD camera. The first image is timed to capture the atoms while they fall under gravity. The illuminating beam is partially absorbed by these atoms, casting a shadow on the camera. A second image is recorded after the atoms have fallen out of the field of view, with the absorbing beam on: this measures the nominal intensity of the absorbing beam. A third image is recorded next, with neither beam nor atoms, and thus captures the background illumination. By subtracting the third image from the first two, and using Beer's extinction law along with the well-known absorption characteristics of the rubidium atoms [35], we measure the number of atoms at the end of each experimental sequence.

$$N = \sum_{\text{num pixels}} \frac{A_{\text{pixel}}}{\sigma_{\text{scs}}} \ln\left(\frac{\text{CCD counts, no atoms}}{\text{CCD counts, with atoms}}\right), \tag{4.1}$$

where A_{pixel} is the area of single pixel, σ_{scs} is the resonant scattering cross-section for rubidium, and the sum is performed over every pixel in camera's field-of-view.

We show samples of experimental TOF images in figure 4.5. These images are processed composites of the three raw images, allowing extraction of atom number estimates. The fundamental uncertainty in the extracted atom number is low, no more that a few percent at most.

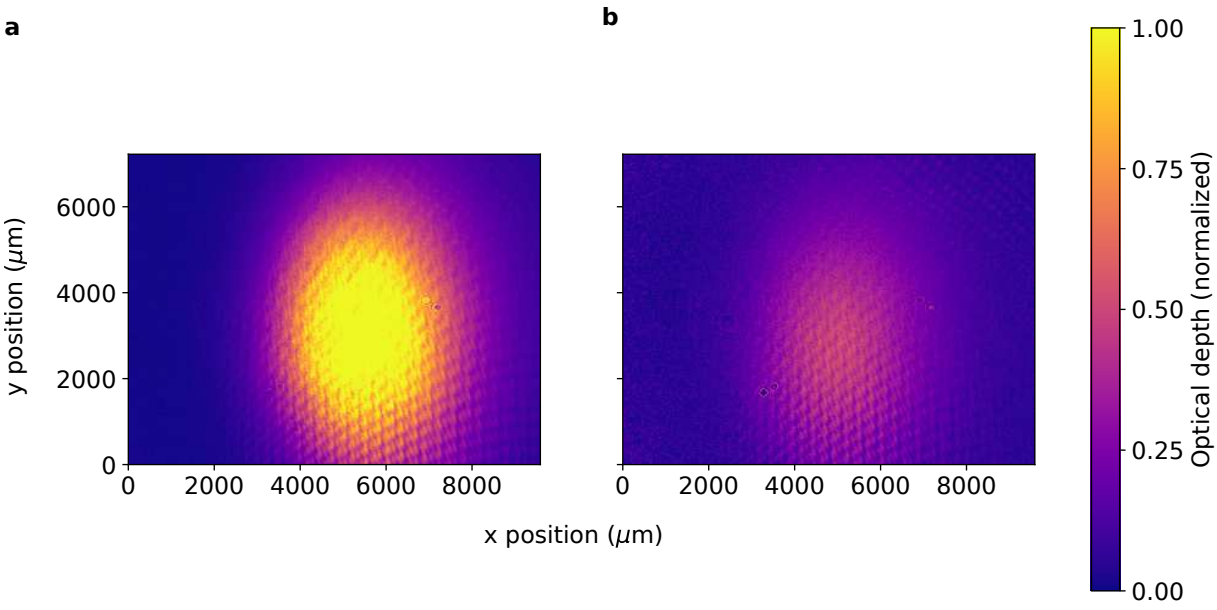

Figure 4.5: Processed TOF images, adapted from our paper [58]. Each image is within a 648 pixel by 488 pixel field of view, where the pixel size is 14.8 μm. The colour scale shows the optical depth of the atoms in cloud, normalized to the maximum. **a** Sample ensemble generated by a well performing control parameter set. We estimate 2.3×10^8 atoms ($\log N = 8.37$) are present in this image. **b** Sample ensemble generated by a poor performing control parameter set. We estimate 3.37×10^7 ($\log N = 7.53$) atoms are present in this image.

4.3 Environmental Sensing

In this work, we rely on real-time monitoring of 30 environmental parameters that we suspect play a vital role in our cooling process. These parameters collectively constitute the environmental state provided to our optimizing agents. For a quick summary of the monitored parameters, please refer to Table 4.2.

We monitor the majority of parameters at the end of each cooling cycle, which decouples the measured environmental parameters from the chosen control parameter values. Exceptions to this timing are our coil current parameters E19-E21. These parameters are measured during different stages of the cooling cycle (MOT for E19, MT for E20, E21) via the same current transducer used for our feedback stabilization (Danfysik Ultrastab 867-400). To prevent interferences between our agent's actions and the measured state, we do not allow our agent to control the maximum current or the current stability of our MOT and MT, and in this way, these parameters remain uncoupled from the control parameter actions of the agents. The signal from the current transducers is split and sent to both our feedback controller as well as to our primary data acquisition microcontroller, which samples the current 4 times per second. This collection frequency is fast enough to sample several points of the current active during the MOT, as well as during the high field "compressed" MT. A sample current signal for a single cycle is shown in figure 4.6. Post processing is performed to determine which parts the signal correspond to the MOT or the compressed magnetic trap. The maximum current

Table 4.2: Monitored environmental parameters

Name	Number	Implementation
Vacuum pressure	E1	Ion pump pressure gauge
Room temperature	E2	Thermistor
	E3	
Room humidity	E4	Capacitive humidity sensor
	E5	
Coil temperature	E6	Thermocouple
magnetic field x component	E7	Magnetometer
	E8	
	E9	
	E10	
magnetic field y component	E11	Magnetometer
	E12	
	E13	
	E14	
magnetic field z component	E15	Magnetometer
	E16	
	E17	
	E18	
Maximum MOT current	E19	current transducer
Maximum MT current	E20	
MT current stability	E21	
2D repump beam power	E22	Photodiode
2D cooling beam power	E23	
3D repump beam power	E24	
3D cooling beam power	E25	
push beam power	E26	
2D repump beam polarization	E27	Photodiode pair &
2D cooling beam polarization	E28	polarizing beamsplitter
3D repump beam polarization	E29	
3D cooling beam polarization	E30	

during the MOT (E19) and MT (E20), as well as the standard deviation of the MT current (E21) are thus recorded for each cycle to be sent to the agents.

Environmental parameter E1 tracks the vacuum pressure of the oven chamber. Maintaining ultrahigh vacuum is critical, as higher vacuum pressures may result in decreased trap lifetimes and inefficient cooling procedures. Referring back to figure 4.4, our vacuum system can be thought as being segregated by a gate valve into two main components: the oven which houses solid ^{87}Rb used as a source, and the atom cooling chambers. Recall that each side of the gate valve has an ion pump constantly pumping on the system. Measurements of the pressure in each side of the system are estimated by the current generated by the pump. While the pressure is so near vacuum on the cooling side of the valve (of the order 1×10^{-11} Torr) that the ion current is too low to be read, we use the ion current by the oven chamber's ion pump

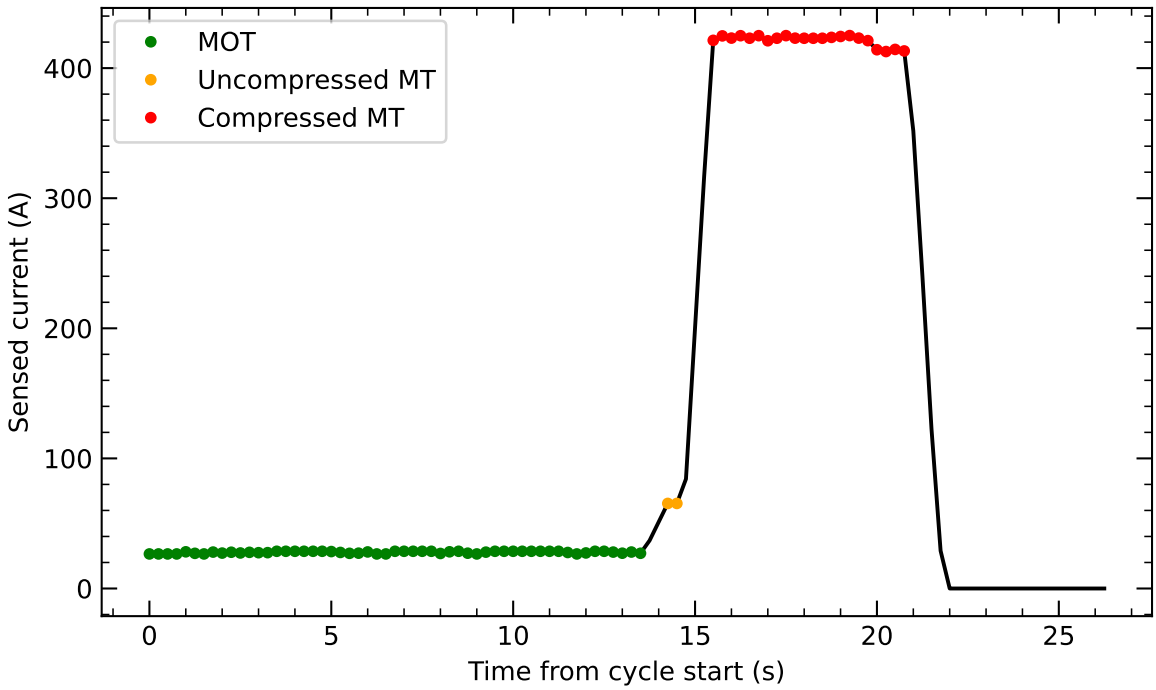

Figure 4.6: Sample measured current signal for a single cycle. Points are sampled while the MOT (green), uncompressed MT (orange), and compressed MT (red) are active. The points from the MOT and compressed MT are then automatically processed to determine E19, E20, and E21 for the given cycle. The black line connects adjacent sensed points, as a guide to the eye.

as a proxy. We read this pressure on 2 minute intervals and send it to our data acquisition computer directly via serial communication.

Two ambient-condition probes (Adafruit AM2302) are placed in our laboratory, each measuring room temperature and humidity once per cooling cycle. One is placed next to our power supplies (E2 and E4), and connects directly to the the primary data acquisition microcontroller. The other probe is placed next to the 3D-MOT (E3 and E5), and communicates with the primary data acquisition microcontroller via radio transceiver module (nRF24L01 - 2.4 GHz). Temperature and humidity both may have effects on the cooling procedure by changing the air index of refraction, slightly misaligning optical elements, and performance of the diode lasers. Furthermore, a K-type thermocouple measures the temperature of the main magnetic coil used both in the 3D-MOT and MT. The thermocouple is secured via Kapton tape to the 10 AWG cable connecting the two coils. Changes in the coil temperature would change the resistance of the circuit, and therefore could effect the performance of the magnetic traps.

Using a magnetoresistive vector magnetometer (Honeywell HMC1053) placed next to the 3D-MOT, we take multiple readings at the end of each cycle, sent directly to our data acquisition computer via serial communication. We measure the field from a fixed current when just the x-axis, y-axis, and z-axis bias coils are active, and when just the MOT coils are active. respectively. We assign parameter numbers for these field readings in the following way. Along the $x-$, $y-$, and $z-$axes respectively, we measure the magnetic field when the x-axis bias coils

are active, alone, at a fixed current (E7, E11, E15), when the y-axis bias coils are active alone at a fixed current (E8, E12, E16), when the z-axis bias coils are active alone at a fixed current (E9, E12, E17), and when the 3D-MOT/MT coils are active at a fixed current (E10, E14, E18). We use such a configuration to detect unwanted external fields along all three Cartesian axes, as well as to detect potential issues with our power supplies. This separates completely independent background fields from unintended fluctuations from our coil pairs.

From our Evanescent Optics custom beam splitting optical fibres, there are additional paths which divert between 0.5% and 0.8% of the inputted beam power. We use this small amount of diverted power of the 2D- and 3D-MOT's repump and cooling light for monitoring of the beam powers (E22-E25) and polarizations (E27-30). To obtain the polarizations of all four beams, we need eight measurements, one for each polarization component. We achieve this with four photodetectors (Thorlabs DET36A2), two polarizing beams splitters, and a staggered pulse sequence at the end of each cycle. The four detectors are connected to our secondary micro-controller, which we use just for monitoring of the laser powers and polarizations. Figure 4.7a presents a schematic of the photodetector-beam splitter configuration used. Each photo-detector pair has diverted repump and cooling light of the same beam size. The first pulse is the repump light, split into its vertical and horizontal polarization components and measured by the detectors. The second pulse is the cooling light, which is also split into its constituent polarization components and measured. The repump beams are weaker than their respective cooling beams, and thus post processing to find which sections of the signal are from which pulse is possible. Figure 4.7b shows a sample reading from one detector at the end of a given cycle. The average beam intensity for each detector $I_{1,2}$, during each pulse, is thus computed by integrating over the post processed signal, and the powers P (which is trivially proportional to the measured intensity via the detector size which we omit) and polarization angles θ are respectively calculated as [62]

$$P = I_1 + I_2 \tag{4.2}$$

and

$$\theta = \frac{1}{2}\frac{I_1 - I_2}{I_1 + I_2} \tag{4.3}$$

Drops in power or rotations of polarization would lead to inefficient cooling during both the MOT and sub-Doppler stages (refer back to chapter 3 to see the importance of intensity and polarization) .

The push-beam power (E26) is sensed via photodetector after passing through the entire vacuum chamber and exiting a window near the 3D-MOT. The signal is sent to the primary microcontroller. Post processing for pushbeam just requires integrating each end of cycle pulse to find the average value. An underpowered push beam would fail to transfer the atoms optimally from the 2D-MOT to 3D-MOT, whereas if overpowered it would destroy the MOT's ability to effectively capture atoms.

Acquisition of the sensed environmental parameters is done with two simple micro-controllers

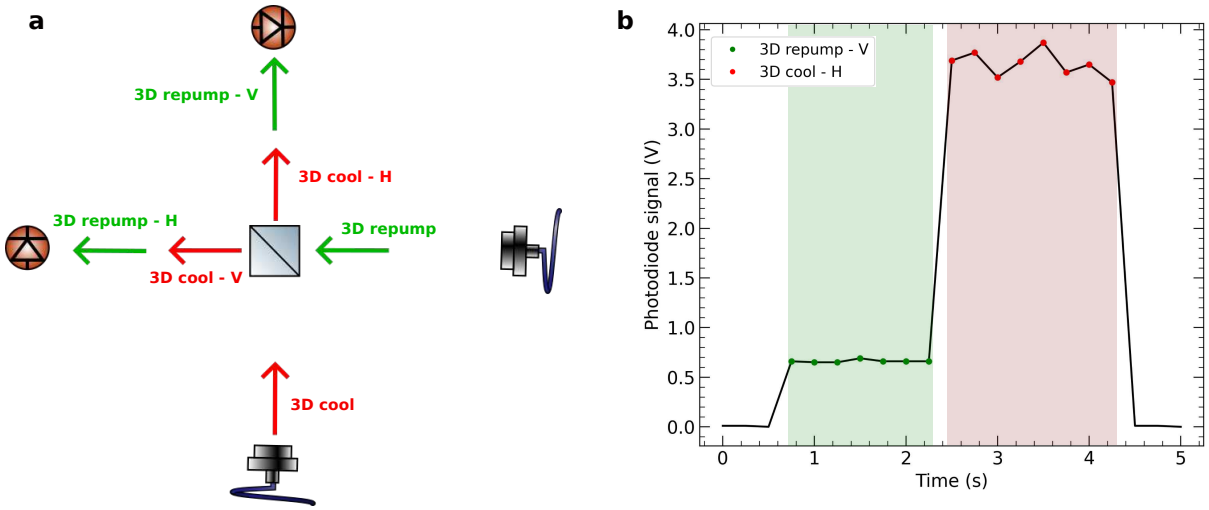

Figure 4.7: a: Laser monitoring detector scheme. Small percentages of the beam powers used in the cooling sequence are picked off via beam splitting fibre. To measure powers and polarizations, fibres are assembled in pairs of two, incident on a polarizing beam splitter. Two photodiodes follow each beam splitter output, to record the horizontal and vertical polarization components of the beams. At the end of each cooling cycle, the beams are pulsed in a staggered sequence so that polarization components of each beam may be resolved. **b**: Sample signal from one of the two photodiodes. The timed and staggered sequence of the pulsing makes it possible to resolve the two beams. With both detector signals, power (equation 4.2) and polarization (equation 4.3) are calculated automatically every cycle, for 2D and 3D cooling and repump beams. The green shaded region and points indicates timing of the repump laser pulse. The red shaded region and points indicates timing of the cooling laser pulse. The black line connects adjacent detected points as a guide to the eye.

(Arduino Mega 2560), along with our magnetometer which interfaces directly to our data acquisition computer. The micro-controllers and magnetometer interface with a central computer via USB serial communication.

Chapter 5

Agent Design

In this chapter, we outline our two different artificially intelligent agents. Fundamentally, the question posed to the two agents we design is the same: given the current state of the environment $\mathbf{x}_e$, what is the ideal agent action $\mathbf{x}_c$ (i.e., the settings of some controllable parameters) that maximizes the number of atoms imaged at the end of the cooling procedure? A direct maximization with any method including Bayesian optimization is unsuited, as the objective function being optimized varies with environmental fluctuations. Instead, we devise two agents: one based on a supervised learning regression model, and one based on reinforcement learning.

The implementation of all of the machine learning algorithms detailed in this chapter were written from scratch in Python using Google's neural network software library "Tensorflow". Including the reinforcement learning agents, no third party implementations were used. This deliberate choice ensures the maximum reproducibility of this work, free from dependencies on external software, as well as giving us full control over the implementation details.

5.1 Supervised–Regression–Based Agent

The first agent, which we refer to as our supervised–regression–based agent, is an extension on the method used in most of the previous work on ML control of atom cooling experiments. This class of ML is the one used in several previous studies investigating ultracold atom preparation [3–11]. In such experiments, a desired output quantity is defined such as ensemble temperature or number of atoms, and some a regression model is trained to map a set of controllable parameters to the output quantity. The trained model is then optimized to find the controllable parameter input set which maximizes the output. The universal–function–approximating models commonly used for cold atom optimization are Gaussian processes and artificial neural networks. Here, we use a deep, feed-forward, densely connected artificial neural network to extend this style of regression–based agent to control the control–parameters cycle-to-cycle in reaction to environmental changes.

The input layer of the regression–based agent's neural network accepts a vector $\mathbf{x} = (\mathbf{x}_c, \mathbf{x}_e)$ that concatenates the control parameters and the environmental parameters, while the output layer predicts $\log(N)$. The internal architecture of the network consists of four hidden layers, each with 128 neurons. This specific architecture was found to give the best predictive capabilities on out-of-sample validation data. We choose the non-linearity following each hidden layer to be the Gaussian-Error Linear-Unit (GELU) activation functions [63]. GELU activations have the form

$$\sigma_{GELU}(\mathbf{z}) = \frac{\mathbf{z}}{2}\left(1 + \text{erf}\left(\mathbf{z}/\sqrt{2}\right)\right) \tag{5.1}$$

where $\text{erf}(x)$ is the Gauss error function. We chose GELU activations for the supervised–regression–based agent for several reasons: GELU is theoretically motivated because it differentiates easily due to its smoothness, and does not saturate anywhere, so gradient descent does not get "stuck". Furthermore, it was also used successfully in previous work using neural networks to model and optimize cold atom experiments [3], albeit without environmental factors.

To avoid our network weights initialized too large or small (such that training is immediately unstable), our network weights and biases are initialized using the Glorot-normal method, which is common protocol for neural network design [64, 65]. Glorot-normal initialization sets the initial weights by drawing them from a normal distribution $\text{Normal}(0, \tau_i^2)$. The crucial part of Glorot initialization is choosing the variance of the initial weights. The variance for a given layer is set based on the number of input nodes and output nodes of that layer. Specifically, the i–th layer's variance is

$$\tau_i^2 = \frac{2}{d_i + d_{i+1}}\,, \tag{5.2}$$

where d_i is the number of inputs and d_{i+1} is the number of outputs. The factor of 2 in the numerator is because in backpropagation, there are two sources of gradients: one during the forward pass and another during the backward pass, and we assume that these two sources are equally significant. When one computes the gradients during backpropagation, one considers both the gradient of the loss with respect to the input (forward-pass gradients) and the gradient of the input with respect to the weights (backward-pass gradients).

The weights and biases are adjusted with a gradient descent algorithm as written in equation 2.4. The procedure we choose for dynamically adjusting the learning rate η_t is the Adam algorithm, which is also standard protocol for modern neural network design [66]. Adam works by adjusting the learning rate based on quantities known as the first and second moments of the gradients. The first moment is a moving average of the gradients, and is updated at each iteration of gradient descent. At iteration $t+1$, the first moment of the gradient, $\mathbf{m}_{t+1}$,

is calculated as

$$\mathbf{m}_{t+1} = \beta_1 \mathbf{m}_t + (1 - \beta_1)\nabla_{\boldsymbol{\theta}} L(\mathbf{Y}, \mathbf{f}(\mathbf{X}; \boldsymbol{\theta})) \tag{5.3}$$

The second moment is a moving average of gradients squared, and it essentially tracks how much the gradients vary. At iteration $t + 1$, the second moment of the gradient, $\mathbf{v}_{t+1}$, is calculated as

$$\mathbf{v}_{t+1} = \beta_2 \mathbf{v}_t + (1 - \beta_2)[\nabla_{\boldsymbol{\theta}} L(\mathbf{Y}, \mathbf{f}(\mathbf{X}; \boldsymbol{\theta}))]^2 \tag{5.4}$$

where $\beta_{1,2}$ are the hyperparameters controlling the moving average decay rates. To correct bias of the raw first and second moments towards their initialized values of zero, bias corrected moments $\hat{\mathbf{m}}_t$ and $\hat{\mathbf{v}}_t$ are calculated as

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t} \tag{5.5}$$

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t} \tag{5.6}$$

where $\beta_{1,2}^t$ means $\beta_{1,2}$ to the power of timestep t. Due to the fact that the moving averages are being updated from a zero initial value, any small correction has a substantial effect. In practice, this can cause a bias towards lower values, making the optimization less stable. These corrected moments amplify the moment estimates at initial times. With these corrected moments, the network parameters are updated as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\eta_0 \hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}\,, \tag{5.7}$$

where η_0 is the initial learning rate hyper-parameter and ϵ is a small constant added to prevent division by zero. Note that the square root and subsequent division of vectors denotes element-wise operation. For clarity, the network parameters denoted $\boldsymbol{\theta}$ consists of weights and biases. Each layer, being a linear transformation, has a weight matrix $\mathbf{W}_i$ and bias vector $\mathbf{b}_i$, such that learning all of the parameters $\boldsymbol{\theta}$ means learning all the elements w_{ijk} of a rank-3 weight tensor, and all elements b_{ij} of a rank-2 bias tensor.

In Adam, the first corrected moment plays the role of momentum for the gradient descent, as it captures the historical trend of how the gradients have been changing, giving some inertia to the parameter updates. The second moment keeps track of how much the gradients have been varying or fluctuating, essentially modulating the parameter updates based on local curvature of parameter space. We find best predictive performance with decay factors $\beta_1 = 0.9$ and $\beta_2 = 0.995$, stability factor $\epsilon = 1 \times 10^{-7}$, and initial learning rate $\eta_0 = 0.00005$.

A possible pitfall in this agent's design is catastrophic untracked changes, such as the spontaneous unlocking or mode-hopping of one of our lasers. As a contingency to make the training more robust against these rare outlier points, performance evaluation is carried out

using the Huber loss function [67]:

$$L(\mathbf{Y}, \mathbf{f}(\mathbf{X};\boldsymbol{\theta})) = \frac{1}{N}\sum_{i}^{N} \begin{cases} \frac{1}{2}|\mathbf{f}(\mathbf{x}_i;\boldsymbol{\theta}) - \mathbf{y}_i|^2 & \text{, if } |\mathbf{y}_i - \mathbf{f}(\mathbf{x}_i;\boldsymbol{\theta})| < \Delta \\ \Delta(|\mathbf{y} - \mathbf{f}| - \Delta/2) & \text{, otherwise.} \end{cases} \tag{5.8}$$

Huber is essentially a basic mean squared error loss function for errors less than some hyperparameter Δ, and it is linear for errors larger than Δ, while still remaining differentiable. This makes the agent underreact to very large errors, such that the network's learning is less sensitive to large outliers.

Neural networks, such as this supervised learning regression model, are complex models with many fittable parameters. When there is noise present in data, neural networks can overfit the noise, similar to how a high-order polynomial could perfectly fit data generated by adding noise on top of a low-order polynomial [68]. To prevent overfitting in our model, training is terminated when the model's loss, validated on previously unseen data, no longer improves with subsequent epochs. To be sure the solution is not just stuck in local minimum which could be overcome with the stochasticity of our gradient descent method, we allow the optimizer 50 epochs to find its way out before terminating and going back to the best performing parameter set. This is a common technique, sometimes refereed to as early-stopping [17, 69]. A typical training curve is shown in figure 5.1. Notice that the validation loss eventually stops improving while the training loss continues to become smaller, showcasing the benefit of our early stopping implementation.

It is important to note that, unlike with the RL agent (see subsequent section), we do not, in the final model, include additional elements in the environmental parameter vector to handle partial observability, such as information about the model's previous performance. Trials that included these elements resulted in inferior control–parameter actions within our architecture.

The core principle of this regression–based agent is to select the control–parameter set that maximizes the atom number, given a specific environmental state, by partially optimizing the model while holding the environmental parameters constant in the current configuration (Figure 5.2). The control parameter set that maximizes the atom number, given an environmental parameter set, is

$$\mathbf{x}_c^* = \underset{\mathbf{x}_c}{\operatorname{argmax}}\{F(\mathbf{x}_c, \mathbf{x}_e)\}, \tag{5.9}$$

where $\mathbf{x}_e$ denotes the current environmental state, and $F(\mathbf{x}_c, \mathbf{x}_e)$ represents the network model approximating the atom number as a function of control and environmental parameters.

The selection of control–parameter values in this scheme, as is common in similar experiments, relies on finding the set that maximizes the network's output. While neural networks may not in general possess explicit analytical invertibility, approximate optimization methods can be used, such as the gradient–free Nelder–Mead [44] method or probabilistic Bayesian op-

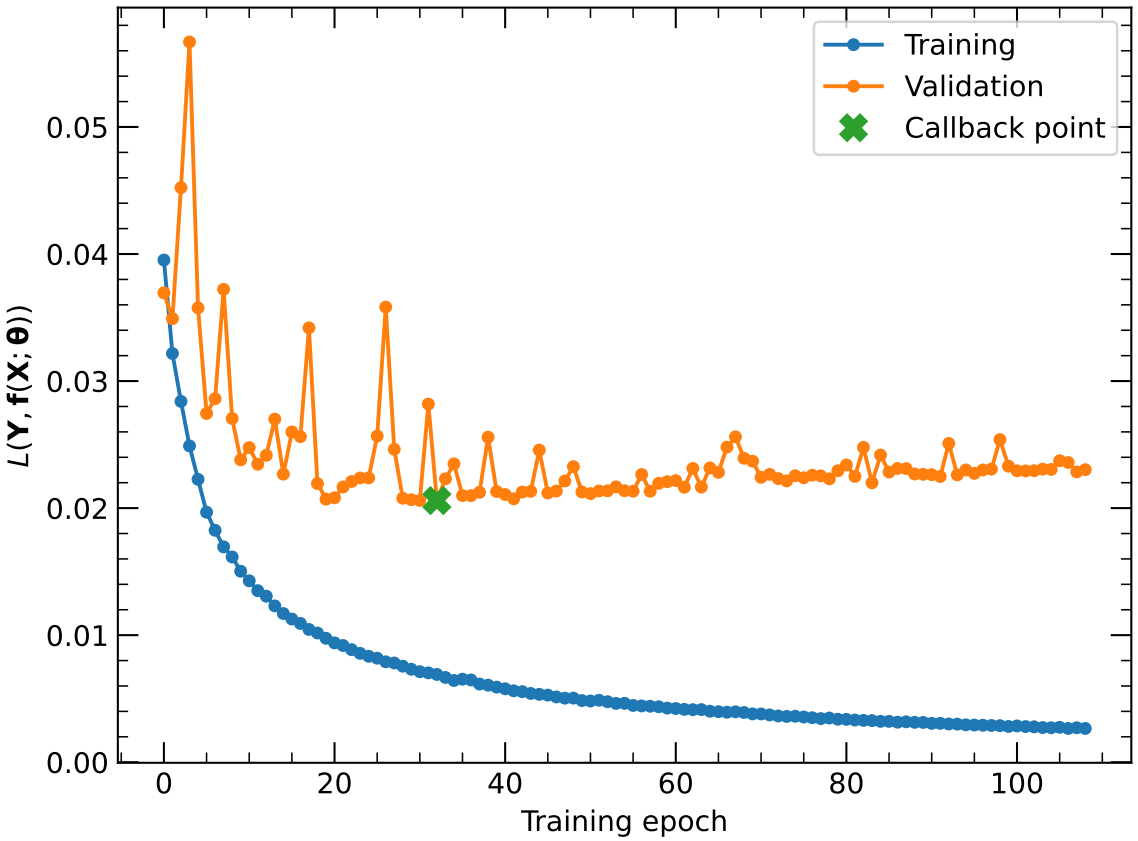

Figure 5.1: Sample training curve of the supervised learning model. Huber loss as a function of epoch (epoch refers to a single pass through the entire training dataset during the training of a neural network), for both training data (blue), as well as validation data (orange). The training and validation data are partitioned in ratio 85:15, at random. The epoch with the lowest validation loss is indicated (green cross), i.e. the weights to which the model is restored.

timization. In this experiment, we employ the Nelder–Mead algorithm, as it yielded the best results given our time constraint that control parameters need to be generated before changes to the environment occur, i.e. optimization ought to take a matter of seconds. Although we explored Bayesian optimization with Gaussian processes, and it returned similar optimizations to the Nelder–Mead when tested offline, the computational feasibility was limited due to the significant training time required for the Gaussian-process regressor: the computational complexity of the Gaussian-process regressor scales cubically with the number of function calls [47, 50], making it impractical for optimizing the neural network within a reasonable timeframe, i.e., before significant changes in the environmental state occur. We found that each Bayesian optimization would take over 15 minutes. On the contrary, the Nelder–Mead optimizations each took under 5 seconds.

To train the network, data was initially collected by sampling the control parameter space randomly. Once 1000 such points were collected, we proceeded to construct the training set using an iterative feedback loop. In this loop, the agent reads the current environmental state, adjusts the control parameters accordingly, creates an atomic ensemble, and measures the

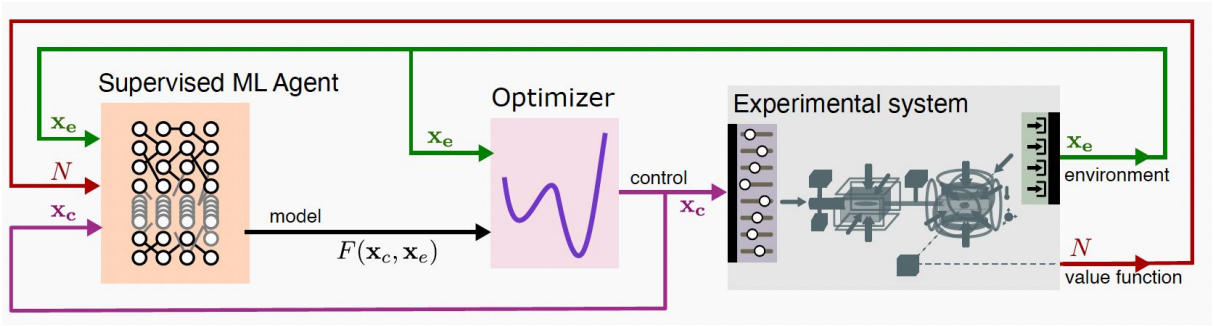

Figure 5.2: Schematic diagram of regression–based agent's live control loop, adapted from our paper [58]. The neural network regressor uses its bank of training data to makes a map between the concatenated control and environment parameter vectors, and the numbers of atoms imaged. The network is then partially optimized to find the control parameters that maximize the outputted number of atoms, given the current environmental state. These maximizing control parameters are implemented into our system, and the resulting cloud of cooled atoms is imaged. With this additional experience tuple consisting of the new atom number and the control and environmental parameters that generated it, the network weights and biases are retrained and the loop may repeat.

number of atoms. This experience tuple is added to the training data bank. By doing so, the network has an expanding training set to improve its model fitting. The feedback loop continues until a training set of 10 005 points is established. The size of this training data bank is large compared to similar experiments [3, 10], which is necessary because our model incorporates environmental parameters that our agent cannot directly control, meaning that many iterations are required to obtain a representative sampling of the combined control and environment space.

5.2 Reinforcement Learning Agent

We refer to the second agent as our reinforcement learning agent. Our system was specifically designed such that the observed outcomes are decoupled from previous control parameter actions, technically making it akin to a contextual bandit problem [41]. Solving contextual bandits is a whole area of research separate from the more complex area of RL for Markov decision processes [42]. Despite the reduced complexity resulting from the decoupling, we opt for a more powerful RL approach as opposed to more traditional contextual bandit solving algorithms. We make this choice for several reasons. The proximate reason being the continuously varying environment. The vast majority of work in the space of contextual bandits are in on the case where the "contexts", i.e. the environmental states, are discrete. We could have discretized our environment state space such that sensed values within a predetermined bin were recorded as a discrete value. However, it is difficult to know a priori how small or large to make such bins, so as to not wash out significant changes in the state. The secondary reasons for choosing the generically more powerful RL method are the high dimensionality of the action space, our lack of knowledge about the objective function's true form, the need

for quick generation of control parameter actions, and the potential for a partially observable environment leading to heteroscedasticity [70].

In contrast to models like our supervised regression–based agent, which was trained to predict the desired output metric based on inputs including the control parameters, the philosophy of RL is to explicitly train an agent to learn an optimal policy, from which control parameter actions may be drawn. Here, we build our agent as an actor-critic neural network. Actor-critic networks are well-suited here, as they generalize naturally to continuous action spaces and learn stochastic policies which allow our optimizing agent some degree of exploration. We show a schematic of the control loop in figure 5.3.

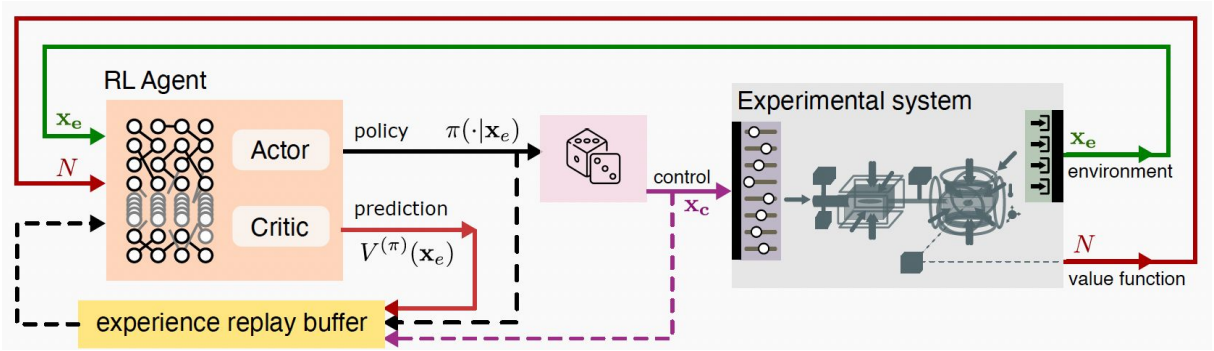

Figure 5.3: Schematic diagram of the actor-critic agent's live control loop, adapted from our paper [58]. The agent leverages the sensed current environment as well as information about its own previous performance to output a probabilistic policy, from which control parameter actions are drawn. The control parameters are implemented into our system, and the resulting cloud of cooled atoms is imaged. The weights and biases of the actor-critic network are then adjusted depending on the number of atoms in the cloud. This loop may then repeat.

Our actor-critic is built as a single neural network. In the literature, schemes exist where the actor and critic are individual neural networks, as well as schemes where a single neural network has output nodes for both the actor's policy and the critic's value function prediction [71]. We chose a single network to both save memory and processing power by reusing parts of the network for both tasks, and so the actor and critic can share learned features of the problem.

The sole input to our network is the environmental state vector, $\mathbf{x}_e$. The network output has two heads: the first head, the actor, seeks to output an optimal policy that chooses subsequent control parameter values, $\mathbf{x}_c$. Considering the 30-dimensional control parameter space in this problem, the actor is learning a 30-dimensional independent normal distribution

$$\pi(\mathbf{x}_c|\mathbf{x}_e) = \text{Normal}(\boldsymbol{\mu}(\mathbf{x}_e), \boldsymbol{\sigma}(\mathbf{x}_e)) \tag{5.10}$$

The actor head consists of 60 output nodes, half of which are learning-control–parameter means $\boldsymbol{\mu}$, while the other half are the associated variances $\boldsymbol{\sigma}^2$. To avoid outputed values incompatible with our system's hardware (for example requesting a bias field that our power supplies physically could not generate) the mean approximating nodes are bounded using

hyperbolic-tangent activation functions, which limits the potential outputs within our allowed range while maintaining differentiability of the network through back-propagation. Similarly, we bound the variance-approximating nodes to be always larger than zero. To accomplish this while still maintaining differentiability, these nodes are followed by softplus activations

$$\text{softplus}(x) = \log(\exp(x) + 1)\,. \tag{5.11}$$

The other head, the critic, consists of a single node. The critic learns to predict the value function $\tilde{v}_\pi(\mathbf{x}_e)$, that is it seeks to predict the resulting atom number while following the actor's policy, given the input environmental state.

The internal architecture of our actor-critic agent consists of four hidden layers, each comprising 128 neurons. Similarly to the supervised regression–based agent, the network is initialized using the Glorot-Normal method (equation 5.2), and network weight and bias updates are computed using the Adam algorithm (equation 5.7) with hyper-parameters ($\beta_1 = 0.9$, $\beta_2 = 0.995$, $\epsilon = 1 \times 10^{-7}$, $\eta_0 = 0.0001$). Unlike the supervised regression–based agent, there was a problem encountered during training: the issue of exploding gradients [72]. Exploding gradients are a problem with neural networks during training, when the gradients of the loss function with respect to the model's parameters become extremely large. This leads to numerical instability and makes it difficult for the network to learn effectively. The GELU activation function allowed the gradients to blow up and the training to fail (see figure 5.4 for a sample training curve where the gradients explode and learning become unstable). To address the issue of exploding gradients during training, we employ Scaled Exponential Linear Unit (SELU) activation functions for each hidden layer [73]. Critically, SELU activations have a self-normalizing property, which keeps the gradients from exploding. SELU activations encourage each neuron's output to maintain a mean value close to zero and a variance close to one. These activations have the form

$$\text{SELU}(x) = \lambda \begin{cases} x, & x > 0 \\ \alpha(\exp(x) - 1), & x \leq 0 \end{cases} \tag{5.12}$$

where α and λ are scaling hyperparameters. Through experimentation with various activation functions, we found that SELU effectively mitigated the gradient instability and allowed training to converge to a well–performing policy.

The supervised regression–based agent uses early stopping, which is a very natural mechanism for preventing overfitting. For reinforcement learning, where the goal is not as simple as fitting models to data, other techniques are needed. For our actor-critic network, we apply an L_1-norm penalty term as regularization, which helps prevent overfitting [17] and has added benefit of promoting sparsity in the network weights, meaning it has the potential to implicitly reduce the dimensionality of the problem [74]. An L_1-norm penalty, sometimes referred to as LASSO (least absolute shrinkage and selection operator) is the addition of a term to the loss

function proportional to the norm of each layer's weight matrix $||\mathbf{W}_i||$. We add an L_1 penalty for each layer of our network, such that our network learns by minimizing the modified loss function

$$\tilde{L}(\mathbf{Y}, \mathbf{f}(\mathbf{X};\boldsymbol{\theta});\boldsymbol{\kappa}) = L(\mathbf{Y}, \mathbf{f}(\mathbf{X};\boldsymbol{\theta})) + \sum_{i}^{N} \kappa_i ||\mathbf{W}_i|| \tag{5.13}$$

where κ_i determines how harsh the penalty is for the i-th layer. We follow a standard approach and allow all the layers' $\kappa_i = \kappa = 0.01$. Notice the penalty is only added to the weights, not the biases. Overfitting occurs when a model becomes overly sensitive to small variations in the input data, resulting in complex, oscillating functions that try to precisely match the target values. This phenomenon demands a high degree of curvature in the model's learned function. The bias parameters, on the other hand, do not significantly influence the curvature of the model. They primarily affect the model's translation, shifting it up or down. Therefore, regularizing bias terms often provides minimal benefit, as the focus of regularization is to control the complexity and variance of the model.

The network's critic node is optimized by iteratively adjusting the network's weights and biases according to equation 2.13. These updates, which follow the gradient of the critic's value function in a direction chosen by the sign of the temporal difference error, may be reformulated by minimizing a squared-error loss function. This makes our agent network compatible with Tensorflow's gradient calculating capabilities [75]. For our case, in which subsequent environmental parameters are decoupled from the previously chosen control parameters, the critic loss function looks like

$$L_c = [\log(N) - \tilde{v}_\pi(\mathbf{x}_e;\boldsymbol{\theta})]^2, \tag{5.14}$$

where N is the measured atom number. Here we have assigned $\log(N)$ as the quantity to be maximized, i.e. reward $R_{t+1} = \log(N)$. $\tilde{v}_\pi(\mathbf{x}_e;\boldsymbol{\theta})$ is the critic's value-function prediction, which depends on learnable network parameters $\boldsymbol{\theta}$. The critic's value-function indicates the expected number of atoms (on a logarithmic scale) that the network predicts when control–parameter actions are chosen according to policy π, given the current environmental state $\mathbf{x}_e$.

The actor nodes are optimized according to equation 2.15, derived from the policy-gradient theorem. As with the critic, this update rule may be reformulated as a minimization for compatibility with the gradient calculating software. This reformulation is less trivial than the critic loss becoming a squared error, but it may intuitively be understood all the same. The actor's policy changes by trying to make a given control parameter set more or less probable depending if outcomes were better or worse than expected. The actor loss looks like

$$L_a = -\log[\pi(\mathbf{x}_c|\mathbf{x}_e;\boldsymbol{\theta})] \cdot [\log(N) - \tilde{v}_\pi(\mathbf{x}_e;\boldsymbol{\theta})]\,, \tag{5.15}$$

where $\pi(\mathbf{x}_c|\mathbf{x}_e;\boldsymbol{\theta})$ is the probability that control parameter action $\mathbf{x}_c$ was chosen from the learnable conditional probabilistic policy $\pi(\cdot|\mathbf{x}_e;\boldsymbol{\theta})$. When the critic overestimates the number of atoms after a given control parameter action, such that zero-discount temporal difference error $\log(N) - \tilde{v}_\pi(\mathbf{x}_e;\boldsymbol{\theta})$ is negative, to make the loss as negative as possible, the probability needs to be made unlikely. Likewise, if the critic underestimates the number of atoms such that the temporal difference error is positive, minimizing the loss function requires making the probability that that control parameter action is drawn to be highly probable. This tandem iterative process of making the critic predict as accurately as possible, while adjusting the actor's probabilistic policy to make actions that the critic underestimated likely (and overestimated unlikely) is exactly how our actor-critic agent converges to a high performing policy.

The total non-L_1-penalized loss function is thus a weighted sum of the two constituent loss functions

$$L = \rho_a L_a + \rho_c L_c \,, \tag{5.16}$$

where the weighting coefficients $\rho_{a,c}$ are hyper-parameters that can be thought of as different initial learning rates for the actor and critic. We found the balance between ρ_a and ρ_c quite brittle. Too large of a ρ_c/ρ_a leads to an overly conservative actor, whereas if the ratio is made too small the actor will take over-reactive actions. We found best results when over-weighting the critic relative to the actor 100 fold, i.e. $\rho_c/\rho_a = 100$.

Despite the comprehensive sensor array we deploy, the complex nature of atom cooling leads us to hypothesize that the recorded environmental parameters may not fully capture the complete environmental state. This limitation, known as the issue of partial observability, means that the agent has access to only a subset of the pertinent information required for informed decision-making. Building upon prior research on autonomous curling robots [76], we enhance the actor-critic network by incorporating temporal information from the agent's past performances. Specifically, we augment the environmental state with variables such as the previous atom number, the previous value function, and the probability of selecting the previous action based on the previous policy. Consequently, the agent can leverage not only the current environmental measurements but also its recent performance history when making decisions. For example, with these additional parameters the agent can know that the previous critic prediction was a vast over-estimate, caused by some un-tracked environmental parameter, and it should attempt to adjust its policy and value function estimation as a response.

The vanilla policy gradient method used to train actor-critic models is generally an online algorithm, where the network weights and biases are updated after every new experience tuple collected. Here, an experience tuple consists of the environmental parameters, control parameters, and the atom number: $\{\mathbf{x}_e, \mathbf{x}_c, \log(N)\}$. Re-calculating the gradient based on the entire bank of past experiences is computationally expensive, and furthermore loses the benefits

of randomly sampling experiences to introduce some stochasticity into the gradient descent. Conversely, calculating the gradient using only the newest experience results in extremely high variance in the updates. As a consequence, the model's parameters can oscillate wildly from one iteration to the next, making convergence very noisy and erratic. To ensure some degree of gradient-calculation stability, we maintain a small experience-replay buffer [77] for performing batch stochastic gradient descent. Due to the addition of historical performance parameters to combat partial-observability, we refrain from randomly sampling the batches for gradient calculation from the experience-replay buffer, so that only the most temporally recent experiences are used in the calculations. Each new experience tuple is appended to the beginning of the buffer, while the oldest point is discarded. Additionally, to use the previously collected experiences from the supervised regression–based agent detailed below, we provide the actor-critic agent with a head start. Before deploying the actor-critic and allowing it to train itself, we train the it on this bank of previously collected points as if they were being collected live, allowing the agent to benefit from this initial training data.

5.3 Parameter Importance Algorithm

Aside from using a regression–based agent for live control of atom cooling procedures, the agents' predictive capability could makes them valuable for experimental design purposes. By leveraging the trained model, we can gain insights into the influence of different measured environmental parameters on the resulting number of atoms. This analysis helps experimental designers identify the critical components of the apparatus that require improvement and optimization. Furthermore, lack of predictive ability can imply that certain key parameters were not being tracked.

To estimate parameter importance, we use a feature permutation algorithm [78]. The algorithm for estimating environmental parameter importance for our regression model is as follows:

1. Train the model using an 85% randomly selected subset of the data bank, incorporating all environmental parameters.

2. Evaluate the Huber loss for the predictions on the remaining 15% of the data, which serves as the validation set.

3. Select an external parameter and randomly shuffle its values while keeping all other parameters unchanged. Re-evaluate the Huber loss on the same validation set to quantify the increase in the loss value, indicating the importance of the parameter.

4. Repeat this process for each parameter column in the input array.

We repeat the algorithm 50 times, selecting different validation sets at random in each iteration.

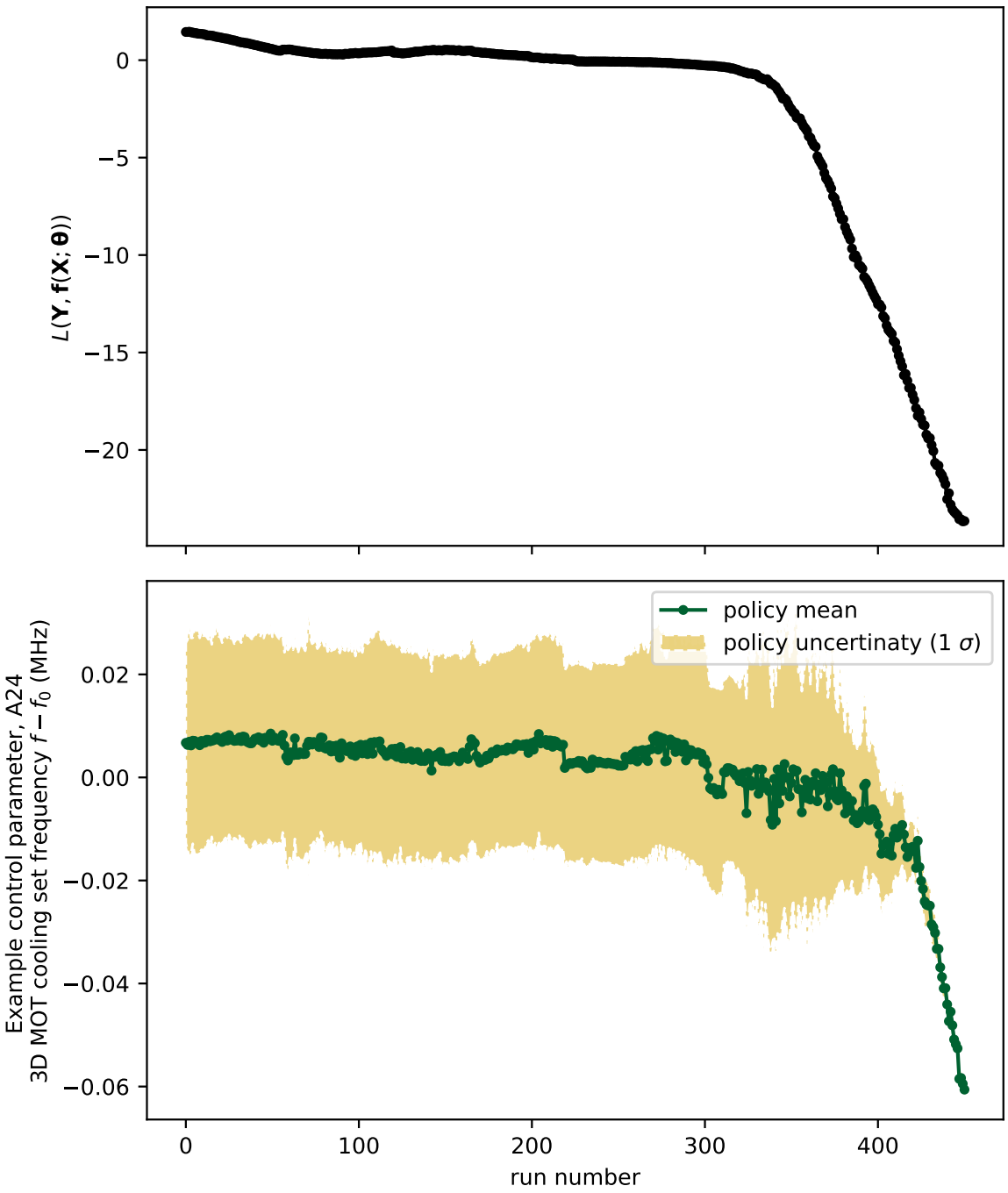

Figure 5.4: Sample offline training of the RL agent using GELU activations instead of SELU. **Top**: Training loss (equation 5.16) as new experiences are collected and the agent's network weights are adjusted via batch gradient descent. Note for roughly the first 300 training epochs, the loss appears relatively stable before rapidly diverging. This behavior occurs when the initial weights of the neural network are such that the gradients remain manageable for the first few epochs, allowing for some progress in training, before suddenly blow up due to exploding gradients. **Bottom**: Sample dimension of the policy, as new experiences are collected and training progresses. This dimension corresponds to the MOT laser cooling frequency (control parameter A24 in table 4.1) policy, where we display the mean value (green), and one standard deviation (gold). Note that when the gradient calculations become unstable, the MOT laser frequency mean becomes further detuned off resonance, and the variance collapses to zero.

Chapter 6

Results

In this chapter, we present the results of deploying our agents live. We show the predictive ability of both agents, estimations of environmental parameter importance based on the predictive capability, and we compare the agents' ability to vary the controllable parameters in response to environmental changes. The live control performance of each agent is furthermore compared to other baselines to gauge the viability of using such agents.

6.1 Supervised–regression–based Agent Performance

Chronologically, we trained the supervised–regression–based agent before the RL agent. We initially trained this agent on 7000 experiences split 85:15 into training:validation subsets. This milestone was chosen as validation-data predictive ability begins to plateau around this number of experiences. The first 1000 experiences were collected by randomly sampling the control space while reading the environmental states. The next 6000 experiences fill out the training set by numerically finding the control parameters maximizing the network output, given the current environmental state. To allow fair comparisons between the agents, the regression–based agent's performance is ultimately re-evaluated using the full bank of experiences, including those collected by the RL agent's exploration, thus resulting in a total data bank of 10 005 experiences.

To assess the predictive capability of adding environmental parameters to extend the class of supervised learning based agents as we have done in this work, we evaluated the performance in predicting atom numbers for given control parameter sets $\{\mathbf{x}_c\}$ when including or excluding the respective environmental parameter sets $\{\mathbf{x}_e\}$. To accomplish this we retrained the model 500 times, each time randomly divided into training and validation sets using an 85:15 split, with different random partitions of the data for each retraining.

On average, we observed a significant reduction in our loss metric when incorporating environmental parameters into the model (Figure 6.1a). This provides confirmation both that the parameters we chose to measure do indeed influence the resulting number of atoms cooled,

as well as that the architecture of the network underpinning this model is able to capture at least some of the complex relationships between environment, control parameters, and atom number.

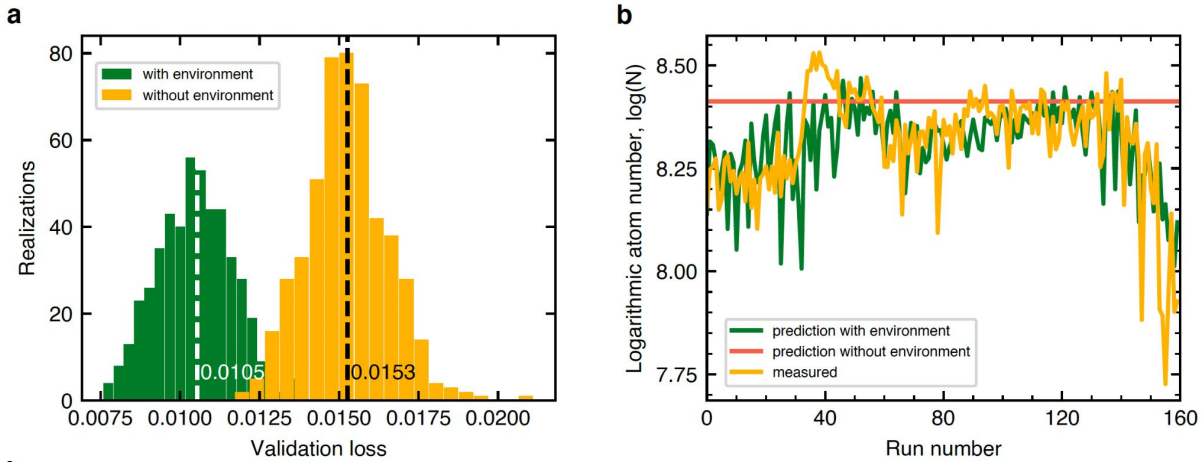

Figure 6.1: Supervised learning model predictive results, adapted from our paper [58]. **a** Histograms of validation data loss, for models including (green) and excluding (gold) environmental parameters. The model is retrained 500 times with a different randomly selected validation set each time. **b** Model prediction (gold) of drifting atom numbers (green). Control parameters are fixed, such that the drift is driven solely by environmental changes. The model's prediction (red) is also shown when excluding the environment.

We further evaluated the predictive power of the model on a separate set of testing points, where the control parameters were kept constant and atom-number fluctuations were driven by the environment only. Our model often successfully predicted (varying) atom numbers over periods as long as 2 hours, consisting of 160 iterations. Figure 6.1b shows a particularly striking example of the model accurately predicting the atom number in response to solely the environmental fluctuations.

After training the regression–based agent and verifying its predictive capacity, we evaluated its performance in generating control–parameter actions in response to different environmental conditions. Ultimately, the effectiveness of the agent should be assessed on two primary metrics: achieving a high overall atom number and maintaining consistent and stable performance over time. To evaluate its live control capabilities, we grant the agent full control of the control parameters for a continuous period of two hours. For comparison, we also assess the performance by fixing the control parameters at the best configuration determined by human experts in our laboratory. This best "human expert" derived parameter set is found by simply scanning each control parameter one at a time in an iterative pattern until further improvement can not be achieved.

Figure 6.3c illustrates the results obtained from the agent's live control and the human expert's best set. While the agent occasionally achieves control parameter sets that surpass human performance, there remains a significant variance in the resulting atom numbers. Over our 160 cycle showcase, the resulting $\log(N)$ generated by the agent has a variance of 0.14,

compared to fixing the control parameters at the human expert's best set resulting in a $\log(N)$ variance of 0.01. In figure 6.3d, we show the agent's chosen MOT laser cooling frequency as an example of the large cycle-to-cycle variability of the control parameters. Note that this high variability in the resulting control parameter set quality is not unexpected. It is consistent with observations from similar experiments [3, 10] inverting supervised learning models to find control parameter sets. In these works, the environment is not considered, so all they needed was a single high performing set which they could henceforth use every cycle. The fact that most optimizations of their model result in poorly performing control parameter sets is of little consequence. This variability however limits such supervised regression–based agent's suitability for live adaptive control and emphasizes the necessity of RL techniques to address this challenge.

Although regression–based agents may be inadequate for live control of atom cooling procedures, their predictive capability makes them valuable for experimental design purposes. By leveraging this trained model, we can gain insights into the influence of different measured environmental parameters on the resulting number of atoms. This analysis helps experimental designers identify the critical components of the apparatus that require improvement and optimization. To estimate parameter importance, we use a feature permutation algorithm (see section 5.3), and we present this metric for all parameters in figure 6.2.

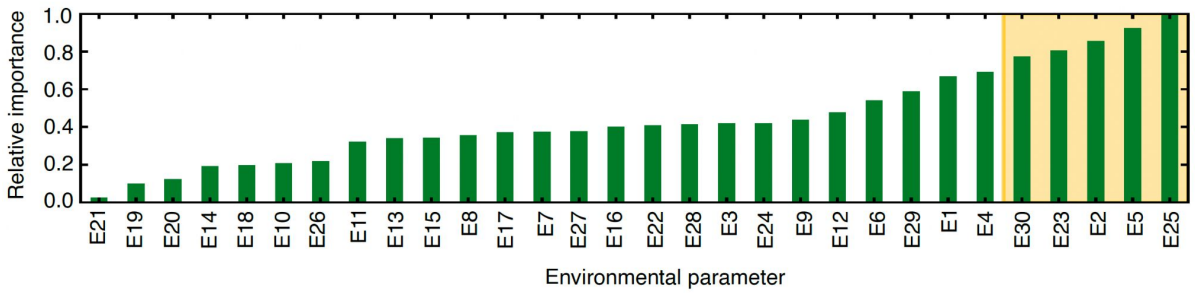

Figure 6.2: Importance of environmental parameters E*i* (see table 4.2 for parameter names), adapted from our paper [58]. The five most important parameters (highlighted gold) include the room temperature (E5) and humidity (E2), as well as the 3D-MOT cooling power (E25) and polarization (E30), and the 2D-MOT cooling power (E23).

6.2 Reinforcement-Learning-Based Agent Performance

The RL-based agent is initially trained on an offline-collected dataset and subsequently refined through live training using the learned probabilistic policy. The training set ultimately comprises 10 005 experiences collected over a period of approximately three months.

To evaluate the performance of the agent, we assessed its actions by implementing the control–parameter actions deterministically, i.e., solely from the policy's mean. Aside from allowing the regression–based agent to react live to the environment (figure 6.3c), for a com-

prehensive comparison, figure 6.3a shows the agent's actions with two alternative baselines: control parameters fixed at a human-optimized set, and fixed at the best set obtained from our regression–based agent. Although we see the environment effecting our system's performance, one may suggest instead of going through the effort of having an agent live reacting to the changing environment, we could optimize a supervised learning model several times until high performance is achieved, and then just apply this control parameter set from that point onward, effectively treating the environment as background noise. This is why we compared the RL agent's live performance to a fixed control parameter set from human manual optimization, as well as a fixed control parameter set from optimizing a supervised learning model. For this we ran three independent experiments, and these three different methods are applied in a sequence one after another. We interchange in between methods in order for the system to be subjected to approximately the same environment for all three approaches. We assume that control parameters from the previous experimental run don't effect the current experiment, so three approaches act independently. The RL agent can consistently, run-to-run, outperform fixing the parameter set at human and supervised learning model derived control parameter sets. This is what is demonstrated over 160 runs in figure 6.3a. We assessed the agent's performance over a period of 160 experimental runs as this period is long enough to assess the capabilities of the agent, while being small relative to the size of the training set, so that any change to agent's network weights will have a negligible effect on performance.

We found that the RL-based agent generates control parameter actions that outperform human optimization as well as holding $\mathbf{x}_c$ constant at a single high performing set found by the regression–based approach. Not only did it outperform these two baselines, it did so consistently and autonomously cycle-to-cycle. As an example of the control parameters used to generate this comparison, figure 6.3b displays the MOT laser cooling frequency dimension of the multivariate policy. On the other hand, the RL agent was not able to fully compensate the drift present in the atom number. While it was able to find a high performing $\mathbf{x}_c$ set in spite of the fluctuating environment and atom number drift, it lacked the capability to fully dampen the effect of the environment.

To further assess the effectiveness of our actor-critic network, we examined the predictive capabilities of the critic. Figure 6.3a additionally presents a comparison between the atom numbers obtained by following the actor's policy and the critic's estimations. The critic's ability to accurately predict the atom numbers generated by the actor's policy serves as extra validation of the RL scheme. The effectiveness of the critic's predictions is crucial in this scheme for several reasons. The critic provides feedback to the actor by estimating the value of different environmental states. These estimates act as a guide for the actor to discern which actions are more likely to lead to higher rewards. If the critic's predictions are inaccurate, the actor will receive misleading information, hindering the learning process and the agent's ability to make optimal decisions. Additionally, a well-performing critic contributes to the stability of the learning process. When the critic provides reliable assessments of state values,

it helps smooth out the learning trajectory, preventing erratic updates to the policy. This stability is crucial for convergence to an optimal policy and ensures that the actor's learning process is grounded in meaningful evaluations of its actions. Lastly, the predictive ability of our critic acts as confirmation that the architecture of the neural network is adequate to capture the complex relationships between the environment, control parameters, and resulting atom number.

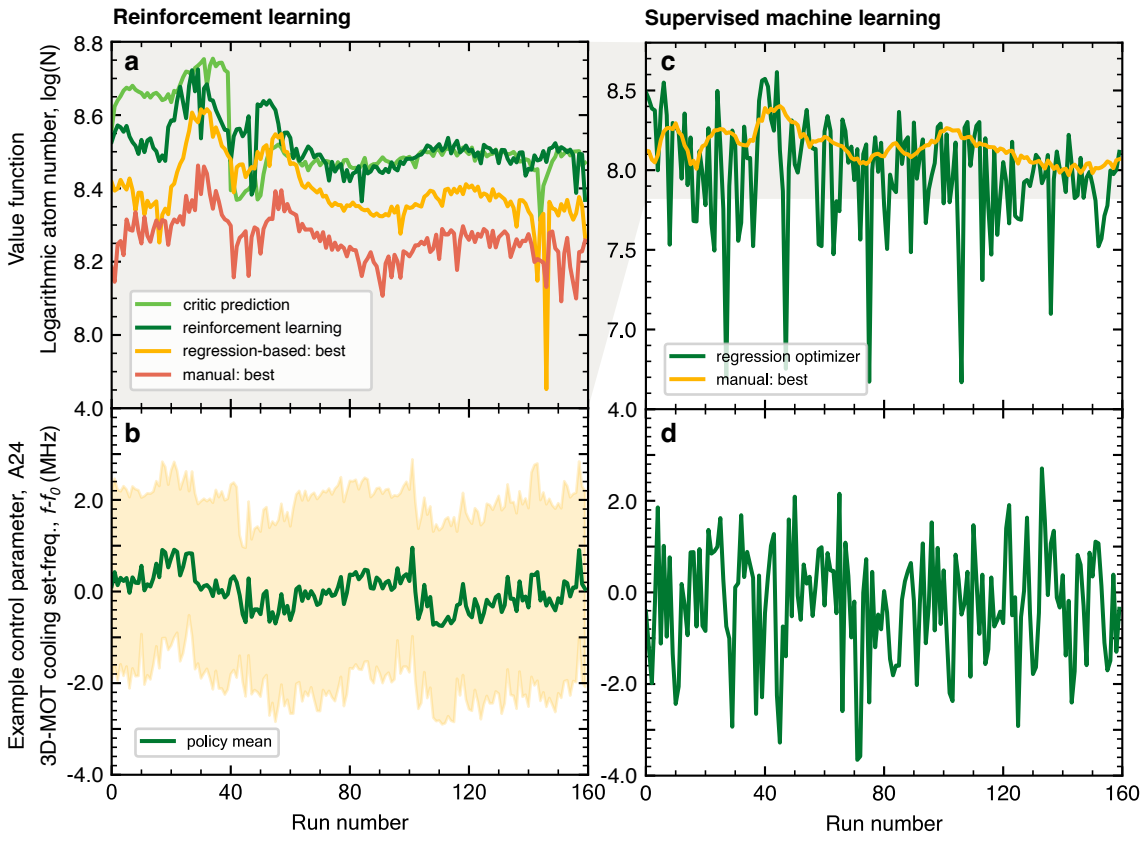

Figure 6.3: Machine learning performance, adapted from our paper [58]. **a** In terms of the target atom-number value function, we compare the reinforcement-learning actor-critic's live control of the control parameters (top, dark green) vs. fixing the control parameters at the best human-optimized set (red) and the best ever set from our regression–based agent (gold). Each run of these three approaches are performed one after another so that the system is subjected to approximately the same environment for all three approaches, i.e., the RL agent takes an action, followed by the fixed regression-agent derived action, followed by the human optimized action. Control parameters from the previous experimental run do not effect the current experiment, so the three approaches act independently. Note here that a run for a given approach constitutes applying the cooling sequence, followed by extracting the resulting number of atoms from the TOF image. This pattern then repeats 160 times. The critic's prediction of the agent's performance is also shown (light green). **b** MOT laser cooling frequency (control parameter A24) policy, showcasing the mean value (green), and one standard deviation (gold). **c** Live performance of the supervised ML agent (green), noting that scale is expanded compared to the RL results, in light of the large fluctuations in the performance. **d** Optimized MOT laser cooling frequency (control parameter A24) policy determined by the supervised ML agent.

Chapter 7

Discussion

The impetus for this work was a genuine need to stabilize our Bose-Einstein condensate generating system. While this same unstable system was used successfully in various recent works, including a demonstration of full control of an ultracold atomic qutrit [79], as well as Floquet engineering non-Abelian geometric phase for the application of holonomic quantum computing [80], the fluctuation of atom number was a huge hindrance and potentially even a critical limitation. At its most benign, it limited the use of the system to only a few hours per day, as the generated ensembles could become so small that it was no longer possible to image the transfer of fractions of the total ensemble population between different quantum mechanical states. Furthermore, in the Floquet-engineering for holonomic quantum computing work, it was found that such a scheme is unrobust to dynamic changes in resonance likely driven by external magnetic fields. Therefore, a self-correcting and environmentally stable system would not only drastically increase the hours available for scientific use but also potentially enhance the sophistication of the system's scientific capabilities.

The idea of live, autonomous optimization and control of atom-cooling apparatuses was largely unexplored, particularly in the context of high-dimensional control–parameter and environmental-parameter spaces. In this way, the work presented in this thesis was a success. We have taken existing machine learning methods used for atom cooling and extended them to be live-reactive to environmental changes, as well as introducing a reinforcement learning approach, which at the time of starting this work was entirely novel for cold atom preparation. We have shown both methods to at least occasionally generate control parameter sets that outperform sets derived by human optimization, and most notably, the reinforcement learning approach was able to consistently cycle-to-cycle outperform all of our other baselines, including human optimized.

When considering the problem of fluctuating atom number that this work was intended to solve, it was less successful. The supervised learning-based agent had much too high of a variance in quality to be suitable for live control, and although the reinforcement learning agent could achieve consistent high performance, the resulting atom number did not have

significantly less drift than the other baselines. The exact reason for this is hard to know and is potentially multifaceted. The fact that the agents, particularly the RL agent, was able to learn a high performing (albeit not significantly drift-resistant) policy gives confidence that the architecture of the agent was not the issue. What is most likely seems to be that there is some critical but untracked environmental parameters, or some critical control parameters that the agent does not have control of, or a combination of both. In fact, this does appear to be the case. At the time of writing this thesis, it has been discovered that the laser beam providing the optical pumping transition was such an untracked source of drift. Specifically, the polarization of the optical pumping beam was not always right-circularly polarized, such that inefficient or even destructive optical pumping would sometimes occur. This appears to be due to a combination of the beam before the fiber having a spurious elliptical polarization, as well as the polarization rotating in the fiber due to light poorly aligned along the fiber's two orthogonal polarization-maintaining axes. The agent had neither a mechanism of sensing this polarization issue, nor a mechanism of correcting it. It was after the partial failure of our agent to compensate for drift given the environmental and control parameters, that led us to know that other untracked and uncontrolled parameters were critical. While having the least scientific novelty, it is perhaps in this way that this work has impacted the apparatus the most.

Given the apparent success of the RL agent relative to the supervised regression–based agent, there is a very natural question of why it performs better.

Overall, while our findings suggest the applicability of RL actor-critic models in controlling the production of ultracold ensembles, we envision this as just the beginning of this a burgeoning area of research, as well as a tool in our laboratory. We were likely limited by a combination of not sensing all the relevant environmental parameters, not allowing control of all the relevant control parameters, and a general dearth of training data. This work has primed us to remove the parameters we found not to be important, and include parameters that were since found to be important. Furthermore, there are some potential ways to increase the speed of training point collection. Allowing a non-destructive imaging technique like fluorescence imaging as opposed TOF absorption imaging would allow the data collection to run non-stop, irrespective of which experiment is being performed using the apparatus. With many points being collected offline, algorithms designed explicitly to be off-policy, such as Soft Actor-Critic [41, 81], could be used to great effect. Additional computing power could allow more-expensive global optimization of neural network regression models, such as Bayesian optimization with Gaussian processes. This, coupled with significantly more training points, could enable regression–based approaches to achieve similar, or even superior, performance compared to RL. Such a high-performing supervised learning model, forming a trustworthy mapping between the combined space of control and environmental parameters to resulting atom number, would have the additional benefit of allowing experimentalists to see how control parameter sets predicted to be high-performing react to perturbations in the environmental parameters, thus giving an estimate of control parameter robustness.

There are many more directions that this work could take as well. Other forms of RL such as Q-learning [41, 82, 83] or Monte-Carlo policy gradients [41, 84] may also hold potential for producing interesting results when applied to the control and optimization of ultracold atom experiments. Discretization of the environment space to enable the application of standard contextual bandit-solving algorithms like ϵ-greedy or gradient bandits is a potential avenue for investigation [41]. Moreover, the incorporation of more complex reward schemes, including ensemble temperature, cooling duty-cycle duration, and cloud shape, among others, could further enhance the performance of such RL approaches. If stability is valued above all else, a reward scheme that maximizes when the resulting atom number is equal to a target atom number could be employed. Additionally, considering phase space density along with atom number and giving the agent control over the details of the evaporation trajectory, our RL protocol could naturally be extended to control the production of BEC. Integrating these ML approaches with advanced environmental-state monitoring, such as magnetometry via the Faraday effect [85] or even using a convolutional neural network to extract non-obvious environmental factors reflected in the TOF images [21, 86], and implementing supplementary hardware configurations like additional coils for magnetic-field compensation [87], RL control could emerge as a crucial tool in the toolkit of atomic physics research and in other similarly high-dimensional experimental systems

References

1 M. Krenn, M. Erhard and A. Zeilinger:
'Computer-inspired quantum experiments',
Nature Reviews Physics **2**, 649–661 (2020).

2 R. Heck, O. Vuculescu, J. J. Sørensen, J. Zoller, M. G. Andreasen, M. G. Bason, P. Ejlertsen, O. Elíasson, P. Haikka, J. S. Laustsen et al.:
'Remote optimization of an ultracold atoms experiment by experts and citizen scientists',
Proceedings of the National Academy of Sciences **115**, E11231–E11237 (2018).

3 A. D. Tranter, H. J. Slatyer, M. R. Hush, A. C. Leung, J. L. Everett, K. V. Paul, P. Vernaz-Gris, P. K. Lam, B. C. Buchler and G. T. Campbell:
'Multiparameter optimisation of a magneto-optical trap using deep learning',
Nature communications **9**, 4360 (2018).

4 S. Xu, P. Kaebert, M. Stepanova, T. Poll, M. Siercke and S. Ospelkaus:
'Maximizing the capture velocity of molecular magneto-optical traps with bayesian optimization',
New Journal of Physics **23**, 063062 (2021).

5 P. B. Wigley, P. J. Everitt, A. van den Hengel, J. W. Bastian, M. A. Sooriyabandara, G. D. McDonald, K. S. Hardman, C. D. Quinlivan, P. Manju, C. C. Kuhn et al.:
'Fast machine-learning online optimization of ultra-cold-atom experiments',
Scientific reports **6**, 25890 (2016).

6 I. Nakamura, A. Kanemura, T. Nakaso, R. Yamamoto and T. Fukuhara:
'Non-standard trajectories found by machine learning for evaporative cooling of 87 rb atoms',
Optics express **27**, 20435–20443 (2019).

7 Y. Wu, Z. Meng, K. Wen, C. Mi, J. Zhang and H. Zhai:
'Active learning approach to optimization of experimental control',
Chinese Physics Letters **37**, 103201 (2020).

8 E. Davletov, V. Tsyganok, V. Khlebnikov, D. Pershin, D. Shaykin and A. Akimov:
'Machine learning for achieving bose-einstein condensation of thulium atoms',
Physical Review A **102**, 011302 (2020).

9 J. Ma, R. Fang, C. Han, X. Jiang, Y. Qiu, Z. Ma, J. Wu, C. Zhan, M. Li, B. Lu et al.:
'Bayesian optimization of bose-einstein condensation via evaporative cooling model',
arXiv preprint arXiv:2303.05358 (2023).

10 A. J. Barker, H. Style, K. Luksch, S. Sunami, D. Garrick, F. Hill, C. J. Foot and E. Bentine:
'Applying machine learning optimization methods to the production of a quantum gas',
Machine Learning: Science and Technology **1**, 015007 (2020).

11 Z. Vendeiro, J. Ramette, A. Rudelis, M. Chong, J. Sinclair, L. Stewart, A. Urvoy and V. Vuletić:
'Machine-learning-accelerated bose-einstein condensation',
Physical Review Research **4**, 043216 (2022).

12 P. Baldi, P. Sadowski and D. Whiteson:
'Searching for exotic particles in high-energy physics with deep learning',
Nature communications **5**, 4308 (2014).

13 D. Finol, Y. Lu, V. Mahadevan and A. Srivastava:
'Deep convolutional neural networks for eigenvalue problems in mechanics',
International Journal for Numerical Methods in Engineering **118**, 258–275 (2019).

14 F. Schindler, N. Regnault and T. Neupert:
'Probing many-body localization with neural networks',
Physical Review B **95**, 245134 (2017).

15 J. Caldeira, W. K. Wu, B. Nord, C. Avestruz, S. Trivedi and K. T. Story:
'Deepcmb: lensing reconstruction of the cosmic microwave background with deep neural networks',
Astronomy and Computing **28**, 100307 (2019).

16 M. Seeger:
'Gaussian processes for machine learning',
International journal of neural systems **14**, 69–106 (2004).

17 P. Mehta, M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher and D. J. Schwab:
'A high-bias, low-variance introduction to machine learning for physicists',
Physics reports **810**, 1–124 (2019).

18 J. Snoek, H. Larochelle and R. P. Adams:
'Practical bayesian optimization of machine learning algorithms',
Advances in neural information processing systems **25** (2012).

19 S. J. Wetzel:
'Unsupervised learning of phase transitions: from principal component analysis to variational autoencoders',
Physical Review E **96**, 022140 (2017).

20 S. J. Wetzel and M. Scherzer:
'Machine learning of explicit order parameters: from the ising model to su (2) lattice gauge theory',
Physical Review B **96**, 184410 (2017).

21 N. Käming, A. Dawid, K. Kottmann, M. Lewenstein, K. Sengstock, A. Dauphin and C. Weitenberg:
'Unsupervised machine learning of topological phase transitions from experimental data',
Machine Learning: Science and Technology **2**, 035037 (2021).

22 C. Wang and H. Zhai:
'Machine learning of frustrated classical spin models. i. principal component analysis',
Physical Review B **96**, 144432 (2017).

23 A. Rocchetto, E. Grant, S. Strelchuk, G. Carleo and S. Severini:
'Learning hard quantum distributions with variational autoencoders',
npj Quantum Information **4**, 28 (2018).

24 Z.-Y. Han, J. Wang, H. Fan, L. Wang and P. Zhang:
'Unsupervised generative modeling using matrix product states',
Physical Review X **8**, 031012 (2018).

25 Y. Ding, X. Chen, R. Magdalena-Benedito and J. D. Martín-Guerrero:
'Closed-loop control of a noisy qubit with reinforcement learning',
Machine Learning: Science and Technology **4**, 025020 (2023).

26 M. Bukov, A. G. Day, D. Sels, P. Weinberg, A. Polkovnikov and P. Mehta:
'Reinforcement learning in different phases of quantum control',
Physical Review X **8**, 031086 (2018).

27 T. Haug, R. Dumke, L.-C. Kwek, C. Miniatura and L. Amico:
'Machine-learning engineering of quantum currents',
Physical Review Research **3**, 013034 (2021).

28 F. A. Cárdenas-López, L. Lamata, J. C. Retamal and E. Solano:
'Multiqubit and multilevel quantum reinforcement learning with quantum technologies',
PloS one **13**, e0200455 (2018).

29 L. Lamata:
'Basic protocols in quantum reinforcement learning with superconducting circuits',
Scientific reports **7**, 1609 (2017).

30 A. Seif, K. A. Landsman, N. M. Linke, C. Figgatt, C. Monroe and M. Hafezi:
'Machine learning assisted readout of trapped-ion qubits',
Journal of Physics B: Atomic, Molecular and Optical Physics **51**, 174006 (2018).

31 J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas et al.:
'Magnetic control of tokamak plasmas through deep reinforcement learning',
Nature **602**, 414–419 (2022).

32 Z. Zhou, S. Kearnes, L. Li, R. N. Zare and P. Riley:
'Optimization of molecules via deep reinforcement learning',
Scientific reports **9**, 10752 (2019).

33 M. Praeger, Y. Xie, J. A. Grant-Jacob, R. W. Eason and B. Mills:
'Playing optical tweezers with deep reinforcement learning: in virtual, physical and augmented environments',
Machine Learning: Science and Technology **2**, 035024 (2021).

34 S. Colabrese, K. Gustavsson, A. Celani and L. Biferale:
'Flow navigation by smart microswimmers via reinforcement learning',
Physical review letters **118**, 158004 (2017).

35 W. Ketterle, D. Durfee and D. Stamper-Kurn:
'Making, probing and understanding Bose-Einstein condensates',
in *Bose-einstein condensation in atomic gases* (IOS Press, 1999), pp. 67–176.

36 Y.-J. Lin, A. R. Perry, R. L. Compton, I. B. Spielman and J. V. Porto:
'Rapid production of ^{87}Rb Bose-Einstein condensates in a combined magnetic and optical potential',
Phys. Rev. A **79**, 063631 (2009).

37 C. M. Bishop:
Neural networks for pattern recognition,
(Oxford university press, 1995).

38 M. A. Nielsen:
Neural networks and deep learning,
Vol. 25 (Determination press San Francisco, CA, USA, 2015).

39 D. E. Rumelhart and D. Zipser:
'Feature discovery by competitive learning',
Cognitive science **9**, 75–112 (1985).

40 R. A. Howard:
Dynamic programming and markov processes.
(John Wiley, 1960).

41 R. S. Sutton and A. G. Barto:
Reinforcement learning: an introduction,
(MIT press, 2018).

42 C.-C. Wang, S. R. Kulkarni and H. V. Poor:
'Bandit problems with side observations',
IEEE Transactions on Automatic Control **50**, 338–355 (2005).

43 R. S. Sutton, D. McAllester, S. Singh and Y. Mansour:
'Policy gradient methods for reinforcement learning with function approximation',
Advances in neural information processing systems **12** (1999).

44 J. A. Nelder and R. Mead:
'A simplex method for function minimization',
The computer journal **7**, 308–313 (1965).

45 W. H. Press:
Numerical recipes 3rd edition: the art of scientific computing,
(Cambridge university press, 2007).

46 J. Mockus and J. Mockus:
The bayesian approach to local optimization,
(Springer, 1989).

47 R. Garnett:
Bayesian optimization,
(Cambridge University Press, 2023).

48 M. I. Jordan:
Learning in graphical models,
(MIT press, 1999).

49 D. R. Jones, M. Schonlau and W. J. Welch:
'Efficient global optimization of expensive black-box functions',
Journal of Global optimization **13**, 455–492 (1998).

50 G. Lan, J. M. Tomczak, D. M. Roijers and A. Eiben:
'Time efficiency in optimization with a bayesian-evolutionary algorithm',
Swarm and Evolutionary Computation **69**, 100970 (2022).

51 P. I. Frazier:
'A tutorial on bayesian optimization',
arXiv preprint arXiv:1807.02811 (2018).

52 C. J. Foot:
Atomic physics,
Vol. 7 (OUP Oxford, 2004).

53 H. J. Metcalf and P. Van der Straten:
Laser cooling and trapping,
(Springer Science & Business Media, 1999).

54 J. Dalibard and C. Cohen-Tannoudji:
'Laser cooling below the doppler limit by polarization gradients: simple theoretical models',
JOSA B **6**, 2023–2045 (1989).

55 R. Grimm, M. Weidemüller and Y. B. Ovchinnikov:
'Optical dipole traps for neutral atoms',
in *Advances in atomic, molecular, and optical physics*, Vol. 42 (Elsevier, 2000), pp. 95–170.

56 J. D. Jackson:
Classical electrodynamics,
(American Association of Physics Teachers, 1999).

57 J. Pérez-Ríos and A. Sanz:
'How does a magnetic trap work?',
American Journal of Physics **81**, 836–843 (2013).

58 N. Milson, A. Tashchilina, T. Ooi, A. Czarnecka, Z. F. Ahmad and L. J. LeBlanc:
'High-dimensional reinforcement learning for optimization and control of ultracold quantum gases',
Machine Learning: Science and Technology (2023).

59 D. W. Preston:
'Doppler-free saturated absorption: laser spectroscopy',
American Journal of Physics **64**, 1432–1436 (1996).

60 K. H. Ang, G. Chong and Y. Li:
'Pid control system analysis, design, and technology',
IEEE transactions on control systems technology **13**, 559–576 (2005).

61 R. W. Boyd:
Nonlinear optics, third edition,
3rd (Academic Press, Inc., USA, 2008).

62 W. Gawlik and S. Pustelny:
Nonlinear magneto-optical rotation magnetometers,
(Springer, 2017), pp. 425–450.

63 D. Hendrycks and K. Gimpel:
'Gaussian error linear units (gelus)',
arXiv preprint arXiv:1606.08415 (2016).

64 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 (JMLR Workshop and Conference Proceedings, 2010), pp. 249–256.

65 L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie and L. Farhan:
'Review of deep learning: concepts, cnn architectures, challenges, applications, future directions',
Journal of big Data **8**, 1–74 (2021).

66 D. P. Kingma and J. Ba:
'Adam: a method for stochastic optimization',
arXiv preprint arXiv:1412.6980 (2014).

67 P. J. Huber:
'Robust estimation of a location parameter',
in *Breakthroughs in statistics: methodology and distribution* (Springer, 1992), pp. 492–518.

68 E. Meijering:
'A chronology of interpolation: from ancient astronomy to modern signal and image processing',
Proceedings of the IEEE **90**, 319–342 (2002).

69 F. Girosi, M. Jones and T. Poggio:
'Regularization theory and neural networks architectures',
Neural computation **7**, 219–269 (1995).

70 Q. V. Le, A. J. Smola and S. Canu:
'Heteroscedastic gaussian process regression',
in Proceedings of the 22nd international conference on machine learning (2005), pp. 489–496.

71 K. W. Cobbe, J. Hilton, O. Klimov and J. Schulman:
'Phasic policy gradient',
in International conference on machine learning (PMLR, 2021), pp. 2020–2027.

72 B. Hanin:
'Which neural net architectures give rise to exploding and vanishing gradients?',
Advances in neural information processing systems **31** (2018).

73 G. Klambauer, T. Unterthiner, A. Mayr and S. Hochreiter:
'Self-normalizing neural networks',
Advances in neural information processing systems **30** (2017).

74 E. J. Candes, M. B. Wakin and S. P. Boyd:
'Enhancing sparsity by reweighted l1 minimization',
Journal of Fourier analysis and applications **14**, 877–905 (2008).

75 Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu and Xiaoqiang Zheng:
TensorFlow: large-scale machine learning on heterogeneous systems,
Software available from tensorflow.org, 2015.

76 D.-O. Won, K.-R. Müller and S.-W. Lee:
'An adaptive deep reinforcement learning framework enables curling robots with human-like performance in real-world conditions',
Science Robotics **5**, eabb9764 (2020).

77 L.-J. Lin:
'Self-improving reactive agents based on reinforcement learning, planning and teaching',
Machine learning **8**, 293–321 (1992).

78 A. Fisher, C. Rudin and F. Dominici:
'All models are wrong, but many are useful: learning a variable's importance by studying an entire class of prediction models simultaneously.',
J. Mach. Learn. Res. **20**, 1–81 (2019).

79 J. Lindon, A. Tashchilina, L. W. Cooke and L. J. LeBlanc:
'Complete unitary qutrit control in ultracold atoms',
Physical Review Applied **19**, 034089 (2023).

80 L. W. Cooke, A. Tashchilina, M. Protter, J. Lindon, T. Ooi, F. Marsiglio, J. Maciejko and L. J. LeBlanc:
'Demonstration of floquet engineered non-abelian geometric phase for holonomic quantum computing',
arXiv preprint arXiv:2307.12957 (2023).

81 T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel et al.:
'Soft actor-critic algorithms and applications',
arXiv preprint arXiv:1812.05905 (2018).

82 C. J. Watkins and P. Dayan:
'Q-learning',
Machine learning **8**, 279–292 (1992).

83 S. Gu, T. Lillicrap, I. Sutskever and S. Levine:
'Continuous deep q-learning with model-based acceleration',
in International conference on machine learning (PMLR, 2016), pp. 2829–2838.

84 A. Lazaric, M. Restelli and A. Bonarini:
'Reinforcement learning in continuous action spaces through sequential monte carlo methods',
Advances in neural information processing systems **20** (2007).

85 D. Budker, D. Kimball, S. Rochester, V. Yashchuk and M. Zolotorev:
'Sensitive magnetometry based on nonlinear magneto-optical rotation',
Physical Review A **62**, 043403 (2000).

86 E. Zhao, T. H. Mak, C. He, Z. Ren, K. K. Pak, Y.-J. Liu and G.-B. Jo:
'Observing a topological phase transition with deep neural networks from experimental images of ultracold atoms',
Optics Express **30**, 37786–37794 (2022).

87 J. Li, K. Lim, S. Das, T. Zanon-Willette, C.-H. Feng, P. Robert, A. Bertoldi, P. Bouyer, C. C. Kwong, S.-Y. Lan et al.:
'Bi-color atomic beam slower and magnetic field compensation for ultracold gases',
AVS Quantum Science **4** (2022).