NORTHWESTERN
UNIVERSITY

Computer Science Department

Technical Report
Number: NU-CS-2022-14

December, 2022

Application of Machine Learning for Spill Regulation System in Mu2e

Jing Jiang

Abstract

Mu2e is an upcoming experiment at Fermilab that observes the direct muon-to-electron conversion in order to look for new physics rules. To achieve rate uniformity of extracted proton spills in Fermilab's Delivery Ring, a Spill Regulation System is under design to regulate the rate variation of the extracted spills. This work presents the investigation of the application of machine learning to the Spill Regulation System, including supervised learning and reinforcement learning. Supervised learning model optimizes the signal via minimizing the difference between output and labeled data. Different from supervised learning, reinforcement learning is a branch of machine learning methods that involves the design a smart agent that can interact with the external environment, free of labeled data. The results demonstrate that the machine learning based methods is able to regulate the spill rate and can outperform the PID controllers in certain ways.

Keywords

Particle Accelerator, Experimental Physics, Machine Learning, Deep Learning, Supervised Learning, Reinforcement Learning

NORTHWESTERN UNIVERSITY

Application of Machine Learning for Spill Regulation System in Mu2e

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

MASTER OF SCIENCE

Field of Computer Science

By

Jing Jiang

EVANSTON, ILLINOIS

December 2022

ABSTRACT

Mu2e is an upcoming experiment at Fermilab that observes the direct muon-to-electron conversion in order to look for new physics rules. To achieve rate uniformity of extracted proton spills in Fermilab's Delivery Ring, a Spill Regulation System is under design to regulate the rate variation of the extracted spills. This work presents the investigation of the application of machine learning to the Spill Regulation System, including supervised learning and reinforcement learning. Supervised learning model optimizes the signal via minimizing the difference between output and labeled data. Different from supervised learning, reinforcement learning is a branch of machine learning methods that involves the design a smart agent that can interact with the external environment, free of labeled data. The results demonstrate that the machine learning based methods is able to regulate the spill rate and can outperform the PID controllers in certain ways.

ACKNOWLEDGMENTS

I would like to give my sincerest gratitude to my advisors and committee members, Professor Han Liu, and Professor Seda O. Memik. Their guidance and advice truly carried me through all stages of this thesis.

I would also like to express my warmest thanks to the group members for this project, Mattson Thieme, Aakaash Narayanan, and Vladimir Nagaslaev. The completion of this work would not have been possible without their help and support. Their suggestions are keys for the formation of my ideas to proceed forward on this project. They also taught me the best practices of conducting research. The values I learned working with them will surely transcend academia and continuously benefit my life.

Finally, it is my deepest pleasure to work with all the wonderful members in this collaboration between Northwestern University and Fermilab. They presented me the fields of Particle Accelerators that I have never imagined to have chances getting involved into.

LIST OF ABBREVIATIONS

AI: Artificial Intelligence.

DL: Deep Learning.

DQN: Deep Q-Learning.

DR: Delivery Ring.

GRU: Gated Recurrent Unit.

LSTM: Long Short-term Memory.

MDP: Markov Decistion Process.

ML: Machine Learning.

MSE: Mean Square Error.

NLP: Natural Language Processing.

Mu2e: Muon-to-Electron-Conversion Experiment.

PID: Proportional–Integral–Derivative Controller.

ReLU: Rectified Linear Unit.

RL: Reinforcement Learning.

SAC: Soft Actor Critic.

SRS: Spill Regulation System.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

In the world of particle physics, scientists from all around the world are interested in discovering new physics beyond the Standard Model, the current description of the building blocks of matter and how they interact. With new discoveries, it is possible to answer some of the most challenging questions about our universe, such as how the universe changed from being dominated by energy and radiation remnants to the one we are in today with visible matter. Particle scientists from Fermilab want to address these challenging questions beginning with conducting the Mu2e experiment [1] using their particle accelerator system.

Mu2e experiment, as the name suggests, probes the muon-to-electron conversion. There are three discovery frontiers in particle physics: Cosmic, Energy and Intensity. Muon-to-electron conversion is key to new discoveries related to Intensity. Intensity Frontier searches will provide part of the context to interpret discoveries made on the other two frontiers and will narrow the number of plausible theories for new physics. Moreover, observing muon-to-electron conversion will remove a hurdle to understanding why particles in the same category, or family, decay from heavy to lighter, more stable mass states. Particle physicists have searched for this since the 1940s. Discovering this is central to understanding what physics lies beyond the Standard Model. Fermilab is currently constructing this experiment and the first series of experiments is expected to start in 2025.

In the Mu2e experiment setting, muon beams are generated by the process of proton extraction. Fermilab is developing a third-integer resonant slow extraction system for its Delivery Ring to deliver protons to the Mu2e experiment. During a slow extraction process, the beam on target is liable to experience small intensity variations due to many factors. The experiment expects a strict requirement in the quality of the spill. Because of this, a Spill Regulation System (SRS) is currently under design by Fermilab. The SRS mainly consists of three components - slow regulation, fast regulation, and harmonic content tracker. Traditionally, proportional–integral–derivative (PID) controllers are used to regulate the extracted spills during fast regulation processes. A PID controller continuously calculates an error value as the difference between a desired value and a variable to generate the correction signals for the regulation system.

Fermilab has initiated Real-time Edge AI For Distributed Systems (READS) project [2] [3] that embraces AI in the field of particle accelerator. Different approaches of AI techniques have been explored under the READS project [4] [5]. In collaboration with READS, we investigated the potential application of Machine Learning (ML) techniques in the fast regulation system in order to replace the PID controller. First, the paper discusses the application of supervised learning. Specifically, Recurrent Neural Networks (RNNs) are used to simulate the replacement of the PID controller. RNNs refers to a class of neural networks where connections between nodes can create a cycle, allowing output from preceding nodes to affect subsequent input to other nodes. The characteristics of RNNs make them suitable for tasks that involve sequential data, such as speech recognition. RNNs are chosen to apply to the model because of the temporal nature of the signals outputted by the regulation system. We presented an RNN system built with Gated Recurrent Units (GRUs) or Long short-term memory (LSTM) mechanisms, and discusses the simulated impact and limitation of machine response characteristics on the effectiveness of both PID and ML regulation of the spill.

Afterwards, Reinforcement Learning (RL) is studied for the application. RL is the field in machine learning that involves the design of smart agents that interact with the environment in order to maximize the cumulative reward. In recent years, it has shown potential in a lot of fields, such as robotics and artificial intelligence. RL is challenging in many ways, including the design for the environment, action shaping and reward shaping. We presented the design of the RL agent using the Soft Actor Critic (SAC) algorithm. The agent interacts with the physics environment in order to find the optimal policy to generate control signals at a given timestamp during the simulation. In a training loop, the agent observes the noised spill signal from the environment, selects the action to regulate the spill, and then updates itself in order to maximize the overall cumulative reward function. Results are analyzed and proof of concepts are then demonstrated by this paper.

1.1 Thesis Organization

The rest of this thesis is structured as follows: Chapter 2 discusses the fundamental experimental physics behind the Mu2e experiment. In a broad picture, this chapter captures the essentials for the resonant extraction process in the Mu2e experiment and then discusses the specific SRS design where machine learning is applied to optimize performance. Chapter 3 covers the specific machine learning techniques and results for the Mu2e experiment. This chapter first introduces the supervised learning method, and then jumps into the reinforcement learning method. This chapter will go through the background, architecture, and result analysis of the proposed methods. Finally, Chapter 4 summarizes this thesis, presenting a conclusion about strengths and the challenges of the proposed methods, along with a discussion about the future directions of this research.

CHAPTER 2
RESONANT EXTRACTION AND SPILL REGULATION SYSTEM

This chapter introduces the Resonant Extraction process and the Spill Regulation System (SRS) in Mu2e. The introduction for the Resonant Extraction contains the basic physics underlying the extraction process and the proposed structure of the third integer resonant extraction in the Mu2e experiment. The introduction for the SRS contains the three parts in its design, with an emphasis on the Fast Regulation system, where supervised learning and reinforcement learning models are applied.

2.1 Resonant Extraction

Resonant extraction is the type of physics process in which a circulating beam's horizontal size is increased gradually, and then a slice of that beam is extracted turn by turn and sent to a desired location through a transfer beam line. The horizontal beam size is increased by purposefully driving the beam close to a resonance condition.

Typically, the extraction of the beam is achieved using an electrostatic septum that is placed at the extraction location which gives an instantaneous horizontal kick to the slice of beam that lies past the septum position in the horizontal plane. This horizontal deflection to the beam directs it to an extraction beam line whereby it is transported to its final location.

Figure 2.1 visualizes the explained process above [6]. It demonstrated a snapshot of the beam at the extraction location. The slice to the right of the vertical red line (the septum) shows the particles to be extracted.

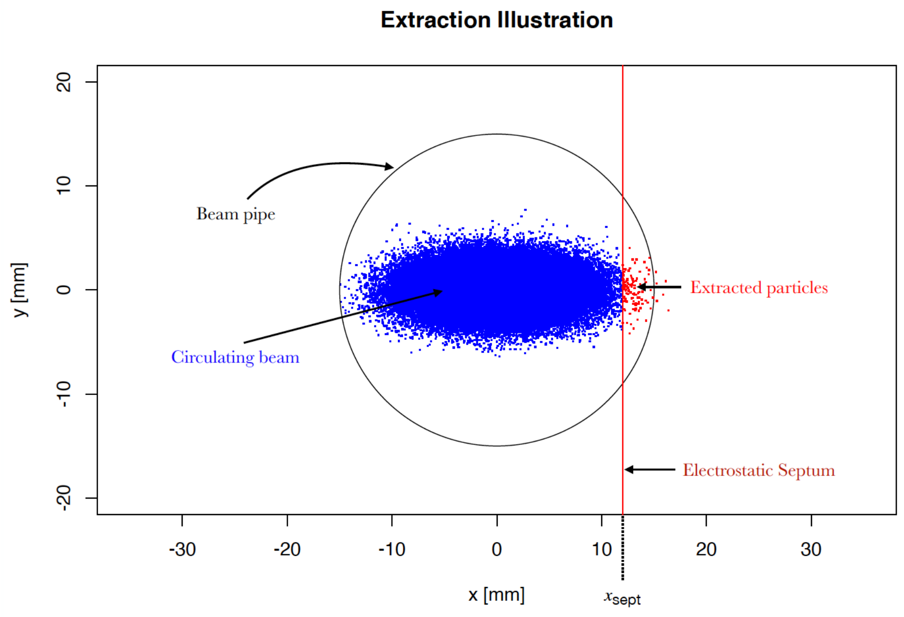

Figure 2.1: Resonant extraction slice.

2.1.1 Third Integer Resonant Extraction in Fermilab's Delivery Ring of Mu2e

The Mu2e experiment uses the non-linear third-integer extraction process [7] [8] [9], which involves the usage of sextupole elements to excite the resonance, and the tune ramping quadrupoles to drive the beam's horizontal tune closer to a resonant tune.

One spill of the process has a duration of 43 ms, and the number of turns over which the extraction occurs is approximately 25,430. This means the process will extracts about 4×10^7 protons per spill. The extraction is to be achieved by keeping the sextupole strength constant but delicately squeezing the horizontal tune from 9.650 to 9.666.

In Fermilab's Delivery Ring (DR), an electro-static septum system is placed at the extraction location. The septum provides a high electrostatic field to give an instantaneous kick in the horizontal direction to the portion of the circulating beam that lies past the septum position in order to

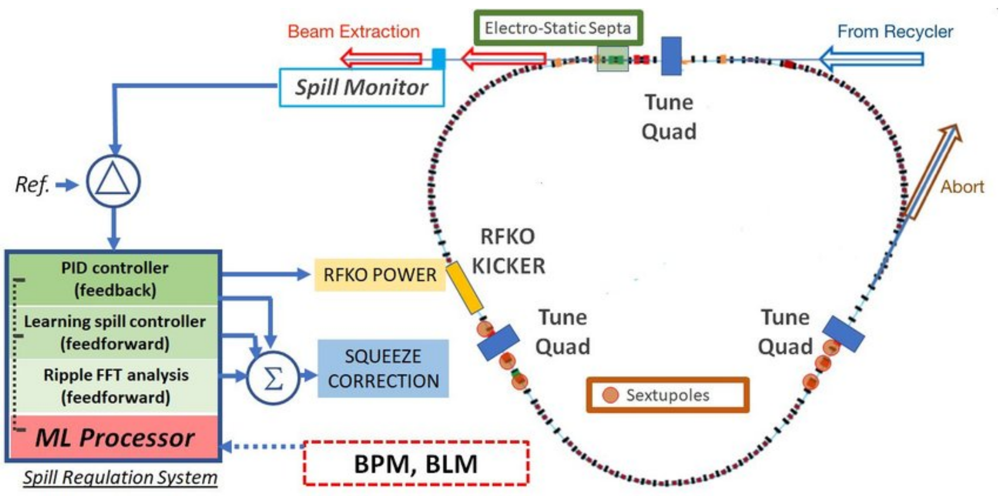

Figure 2.2: Details of Fermilab's Delivery Ring.

direct it to the extraction beam line. Figure 2.2 shows the schematics of the proposed structures in Fermilab's Delivery Ring.

2.2 Spill Regulation System

2.2.1 Spill Regulation System Requirements

The Mu2e experiment has strict requirements on the uniformity of the pulsed muon beam delivery to the stopping target made of aluminum. In Mu2e, the muons are produced by proton pulses hitting the muon production target. Because of this, the intensity of the muons produced from the target depends directly on the intensity of the protons extracted in a turn-by-turn manner through the resonant extraction process performed at the Delivery Ring.

Thus, the Mu2e experiment requires strict rate uniformity of the proton pulse beams. The most important reason for the requirement of rate uniformity is that drastic variations in the spill rate could adversely affect the signal of the Mu2e experiment. For example, the electrons from the cosmic ray could potentially mimic Mu2e electrons, influencing the performance of the Cosmic

Ray Veto detection system.

We primarily study the innovation in the regulation method of the spill rate. Appendix A presents the main beam parameters required for the resonant extraction at Fermilab's DR. For simulation, we assume that the initial number of particles is the same for every spill.

We define a factor that represents the quality of the spill, called the Spill Duty Factor (SDF), with the ideal spill rate normalized to 1.

$$SDF = \frac{1}{1 + \sigma_{spill}^2} \tag{2.1}$$

The SDF function is defined above, where σ_{spill}^2 is the standard deviation of the spill rate. An ideal spill would have a constant spill rate value of 1, therefore the ideal SDF is 1. In SRS, a good SDF value is defined to be of 0.6 or higher.

2.2.2 Spill Regulation System Components

The Spill Regulation System (SRS) [10] [11] primarily contains three components:

- Slow spill profile regulation (Slow regulation)
- Fast random ripple regulation (Fast regulation)
- Harmonic ripple content suppressor

The slow regulation controller tracks the slow changes in the spill profile, due to the characteristics of the proton beam that could slowly vary with time. This controller produces corrections to the three dedicated fast tune-ramping quadrupoles' current ramp, which are needed to achieve the uniformity of spill rate. This slow regulation is done adaptively over many spills.

The fast regulation response is the key mechanism that is used to correct for instantaneous ripples in the spill intensity. The experiment assumes that these fast noise ripples have a random nature or are a semi-random component of regular harmonic noise that the harmonic controller is not able to suppress. Traditionally, it is the part that PID controllers play an important role. And this is the part where we are conducting research to optimize on, aiming to use Machine Learning (ML) to replace the use of PID controllers and upgrade this process with more modern approaches.

Finally, the harmonic content suppressor is the component that cancels out the 60Hz harmonics that may rise from the magnet power supplies and then leak into the spill rate fluctuation.

CHAPTER 3
MACHINE LEARNING IN FAST REGULATION

This chapter introduces the application of Machine Learning techniques to Fast Regulation in SRS. First, the concept of neural networks is described, and the simulation setup is introduced. Then this chapter will demonstrate specific Machine Learning (ML) models, including supervised learning and reinforcement learning, together with the underlying algorithms. Specifically, the RNN algorithm will be introduced in the supervised learning section, and the SAC algorithm will be covered for reinforcement learning. Then this chapter will analyze the results of the models.

3.1 Machine Learning Overview

Machine Learning (ML) is the field that is devoted to building up predictive methods from learning and analyzing certain types of data. ML algorithms are applied in many tasks, such as recommendation systems, computer vision, and artificial intelligence. Given the nature that the fast regulation system in Mu2e's SRS involves the process of generating a control signal, it is natural to apply ML methods.

There are a variety of approaches in Machine Learning. Specifically, we choose Deep Learning (DL) [12] [13] models on this task. The basic building blocks in DL are the neural networks. In a typical neural network, layers of nodes are arranged in different ways to represent the specific task. In each training iteration, the weights and biases of the nodes are updated in order to minimize the loss function. After a certain number of training iterations, the model is tested for fitting the task by making predictions given new data. In this work, we have trained Recurrent Neural Network (RNN) models and Reinforcement Learning (RL) models, and eventually let them control the resonant extraction process so it can regulate the extraction quality.

3.2 Physics Simulator

For the fast regulation process in SRS, we have built a physics simulator to simulate the resonant extraction and generate training data for the usage of ML models [14]. We use Python to code the environment and model, and we use the PyTorch framework to train, validate, and test the ML models.

For the simulation, we assume a perfect extraction without any instantaneous noise in the setup. As discussed before in Chapter 2, we separate slow extraction from the fast regulation, so that their behaviors can be studied independently from each other. We also make the assumption that the slow regulation system can provide an ideal quad current ramp.

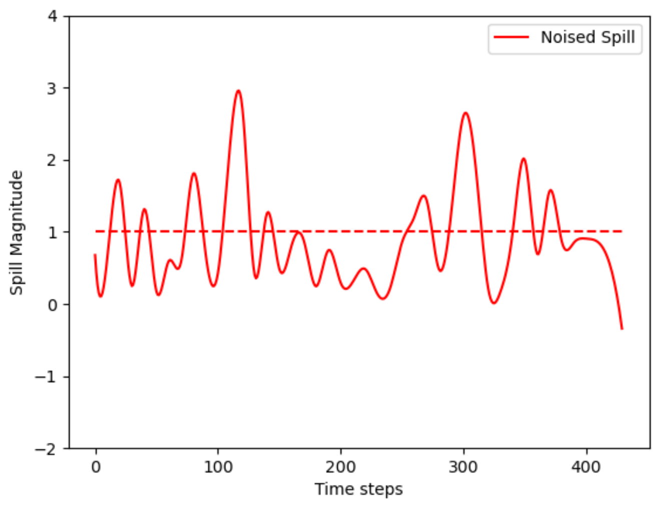

Figure 3.1: An example of noised spill.

We model the noised spill using a log-normal distribution, since we expect random fluctuations in the spill rate from the fast variation ripples. And for every spill, a new noise is generated with a mean value of 0. In Chapter 2, we define SDF to be a measurement for the quality of the spill. The standard deviation for the noised spill is chosen to deliberately make the SDF of the full spill to have a value around 0.5.

The physics simulator generates random log-normal points at 1 KHz rate. And it interpolates the generated points to provide a smooth noise data for over 430 time steps, as we have 43 ms for 1 spill and we assign 10 time steps for 1 ms. Then the expectation value of the spill rate is normalized to 1, and the log-normal generated noised spill is added to 1 for every spill. An example of the generated noised spill is shown in Figure 3.1.

3.3 Supervised Learning with Recurrent Neural Networks

With the aim to replace PID controllers with ML models, we have selected Recurrent Neural Networks (RNN) [15] as the starting point. RNNs are designed to handle training data that contains time-series information, or sequential data. If the training data is labeled in such a fashion that the input at time step t is sensitive to the input given in the previous time-step $t-1$, a basic feed forward neural network would perform poorly because it will fail to capture the relation of the data through time. RNNs are designed to deal with this as it allows output from some nodes to affect subsequent input to the same nodes. For this reason, RNNs are popular in tasks such as speech recognition and translation in Natural Language Processing (NLP) as these tasks exhibit temporal dynamic behaviors.

In our case, RNNs are a natural choice to be considered to replace a PID controller because the physics data for the resonant extraction process is also temporal and thus sequential in nature. Supposing that there is a rising spike in the spill rate as the spill proceeds, we would want the ML model to also consider this spike when generating the regulation signal even if it happened for example 30 time steps ago.

The specific mechanism for RNNs to be better at dealing with temporal data is that they have memory capabilities. While making a decision, an RNN model takes into consideration the current input and also what it has learned from the inputs it received previously. Output from the previous steps is fed as input to the current step, thus creating a feedback loop.

However, there are several issues related to this mechanism. When training RNN models, sometimes the gradients can become too small or too large, resulting in situations so called "vanishing gradients" or "exploding gradients". These problems are due to the fact that RNNs backpropagate gradients through layers and also through time, causing the previous nodes to be considered too many times unnecessarily.

In order to overcome the vanishing and exploding gradient problem in RNNs, we can apply different gating mechanisms, such as Long Short-term Memory (LSTM) [16], or Gated Recurrent Unit (GRU) [17]. In this work, GRU is chosen as the model to proceed with since it is more lightweight.

3.3.1 Gated Recurrent Unit

The overview of a GRU is shown in Figure 3.2. The GRU consists of two types of gates. The update gate helps the model to determine how much of the past information needs to be passed along to the future. And the reset gate decides how much of the past information to forget.

Our implementation contains a GRU with two layers, with hidden state sizes of 128 and Rectified Linear Unit (ReLU) activations [18], The model will ingest at each point in the spill a window of the past 40 observations. To allow the model to begin regulation at the first step in the spill, we prepend the spill profile with a vector of length 40. Each element in this prepended vector is assigned the value of the first element in the random noise profile. Then, we walk forward in the usual manner to predict 430 quadrupole corrections. At the end of each spill, the loss is calculated

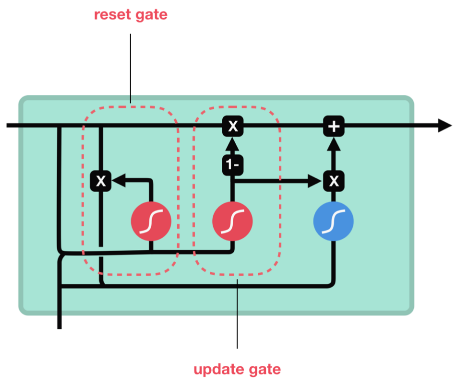

Figure 3.2: Gated recurrent unit.

using Mean Square Error (MSE) and the model parameters are updated.

3.3.2 Recurrent Neural Networks Results

We demonstrate here that the ML model not only performs comparablely to PID controller's regulation but also can outperform the PID controller, after sufficient training iterations.

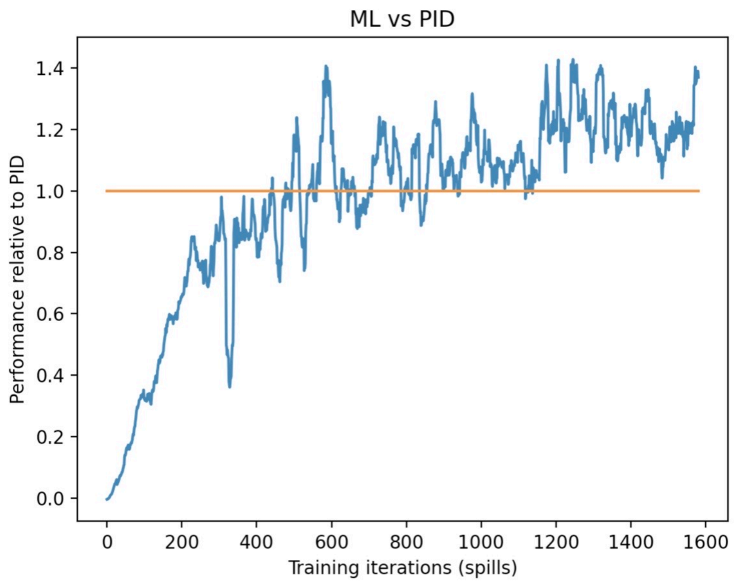

Figure 3.3: Supervised learning regulation vs PID regulation.

In our experiments, we use a window size of 40 time steps. And the ML model at time step t_n will learn and generate predictions of the current to be supplied to the quadrupole on the time step at t_{n+1}.

In Figure 3.3, we compare the performance of the PID controller with the supervised learning controller by plotting the ratio of the improvement in SDF by the ML agent to that of the PID controller. The performance of the PID controller is corresponded with the horizontal line at value 1. And we see in the figure that the ML agent starts outperforming the PID controller after around 500 training iterations.

3.4 Reinforcement Learning with Soft Actor Critic

We've presented that the RNN based supervised learning method can outperform PID controllers. However, since RNN models rely on supervised data to train, it is not practical to apply it in the real world for Mu2e at Fermilab. For this reason, we turned to Reinforcement Learning (RL), which is free from training with supervised data, and can be applied potentially in the real world. As our ultimate goal is to control a physical device, RL represents a compelling opportunity.

Reinforcement Learning [19] is the field of ML that in which a smart agent is designed to perform a certain task in a predefined environment. The agent's goal is learn an optimal policy that maximizes the cumulative reward function it observes during interactions in a turn by turn manner. Because of this, it is commonly applied to gaming Artificial Intelligence (AI) since there are usually scoreboards in those environment setups.

Unlike in supervised learning, there is no supervised data in reinforcement learning, as the RL agent generate its own data when interacting with the environment. The learning process is centered around reward function, which we ascribe to the actions taken by the agent together with the observed state from the environment. Thus the training data is heavily dependent on the action shaping of the agent and the reward shaping of the environment. In addition, the RL methods we apply in our experiment is technically Deep RL, involving the usage of neural networks when designing the RL architecture.

3.4.1 Markov Decision Process

RL can be modeled as a Markov Decision Process (MDP) which involves a state space S that gives the agent the state of the environment, an action space A that consists of actions taken by the agent, a reward $R(s_t, s_{t+1})$ value that is proportional to the goodness of action a_t to take the state from the state s_t at present time t to the next state s_{t+1}, and a transition function $P_{a_t}(s_t, s_{t+1})$ that describes the probability of transitioning from state s_t to state s_{t+1} given an action a_t.

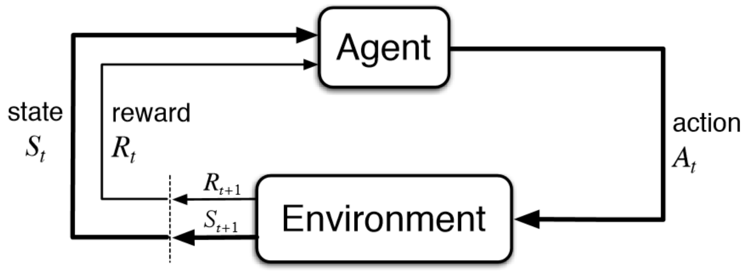

Figure 3.4: Interaction between agent and environment in RL.

MDPs are the foundations of Reinforcement Learning as it defines the trajectories of the interactions between the agent and the environment. Figure 3.4 shows the canonical feedback loop of the agent and the environment.

3.4.2 Value Functions and Bellman Equation

A value function estimates how good it is for the RL agent to be in a given state or how good it is to perform an action in a given state. There are two types of value functions: the state value function, or V-value, and the state-value pair function, or Q-value. We will use Q-value for our RL agent.

The Bellman equation is the central piece in RL algorithms. In summary, Bellman equation decomposes the value function into two parts, the immediate reward and the discounted future values. The Bellman equation greatly simplifies the computation of the value function. So that rather than summing over multiple time steps, we can break a complex problem down into simpler, recursive subproblems and finding their optimal solutions. The Bellman equation for the Q-value is described below, where γ is the discount factor:

$$Q(s,a) = r(s,a) + \gamma max_a Q(s', a) \tag{3.1}$$

3.4.3 Policy Gradient Methods

Different Deep RL algorithms can generally be categorized into two types: value-based and policy-based. In value-based RL, such as Deep Q-Learning (DQN) [20], the algorithms focus on learning the value, either the state value or the state-action value. While in policy-based RL, the agent directly learns the policy function that maps states to actions. In our case, we chose policy-based algorithms because it can learn stochastic policies.

The key algorithm in policy-based RL is the Policy Gradient methods [21]. The fundamental idea behind policy gradients is to push up the probabilities of actions that lead to higher reward return, and push down the probabilities of actions that lead to lower reward return, until the agent arrives at the optimal policy. Policy gradient works by updating the policy network's parameters via stochastic gradient ascent, where π_θ denotes the policy with parameters θ, and $J(\pi_\theta)$ is the expected return of the policy:

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_{\theta_k}) \tag{3.2}$$

3.4.4 Actor Critic Methods

A disadvantage of the vanilla policy gradient method in RL is that it updates the model based on full sampled returns, which are calculated at the end of episodes. This may lead to instability of the model because of the high variance caused by vanilla policy gradient's way of updating.

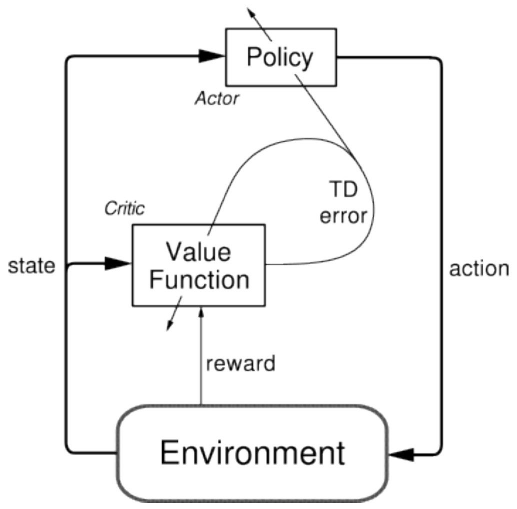

Figure 3.5: Actor Critic.

Actor Critic methods [22] [23] deal with these drawbacks by having the value estimates in the model update to be based on temporal difference bootstrapped from a single step and the Bellman function. By having this design, the agent is capable of having better policy refinement, since it can learn during the episode. Another advantage of Actor Critic is that it can learn in continuous environments, which is essential to our experiment because the simulation of the SRS involves continuous components.

Actor Critic contains mainly two components: an actor and a critic. The actor learns the policy distribution in the direction which is suggested by the critic. And the critic estimates the value function, in this case the state-action value Q. Both the critic and actor functions are powered by neural networks. The architecture of an Actor Critic agent is shown in Figure 3.5.

3.4.5 Soft Actor Critic

The algorithm we use for our experiments is a variation of Actor Critic, called Soft Actor Critic (SAC) [24] [25]. It has better capability for dealing with continuous action spaces. As mentioned before, our environment has continuous components built in. Specifically, the control signal at every time step should be continuous, so that we have a continuous action space. The biggest feature of SAC is that it uses a modified objective function. An SAC agent not only seeks to maximize the cumulative rewards, but it also update itself in order to maximize the entropy of the policy. This feature is called Entropy Regularization.

Entropy is a quantity which says how random a random variable is. Generally, a high entropy has the following advantages:

- Encourage exploration.
- Encourage the policy to assign equal probabilities to actions with equal Q values.
- Ensure that the model does not collapse into repeatedly selecting an action that could invoke inconsistency in the approximated Q function.

Another difference between traditional Actor Critic and SAC is that SAC has 3 networks: 1 actor network and 2 critic networks. This is called the Clipped Double-Q trick. In every update, the minimum of the two estimates is taken for calculation. This trick is applied in order to reduce the bias of the estimations.

3.4.6 Reinforcement Learning Results

Applying SAC, we have implemented and trained an RL agent that is capable of generating regulation signals.

Different from supervised learning, in RL, we use a window size of 10 (which is our state), because we want to capture the most recent changes in the random noised spill. To prevent the errors accumulating, we define a termination condition. If the SDF of the corrected spill drops below a predefined threshold, we immediately assign a large negative reward to discourage the agent from continuing down that path. If this condition is met more than 10 times in a spill, we stop the MDP trajectory and start a new one.

The action space is a continuous space of $[-1, 1]$. This value denotes the magnitude of the increase or decrease of the regulation signal relative to the previous one. The reward is calculated from the regulation signal and the outputted corrected spill at each time step.

The training of the model uses the experience replay technique. At each episodic iteration, we calculate the MDP trajectory of the spill using the agent's policy. We store the agent's MDP experience, which is defined as a tuple $s_t, a_t, s_{t+1}, r_{t+1}$, into a replay memory. We then select a batch of experience replays from the memory and use them to update the policy network and the two critic networks.

Figure 3.6 shows a sample spill after this agent was trained. The red line denotes the original noised signal, while the blue line denotes the corrected spill using the regulation signals generated by the RL agent. From this figure, we can see that the trained agent is capable of pushing the spill closer to 1 (denoted by the horizontal red dotted line), without losing the smoothness of the spill, which is essential for keeping the SDF at a stable level.

Figure 3.7 shows the SDF value of the training session. The horizontal red line denotes the SDF of the noised spill. The blue line denotes the SDF of the corrected spill calculated at each episode.

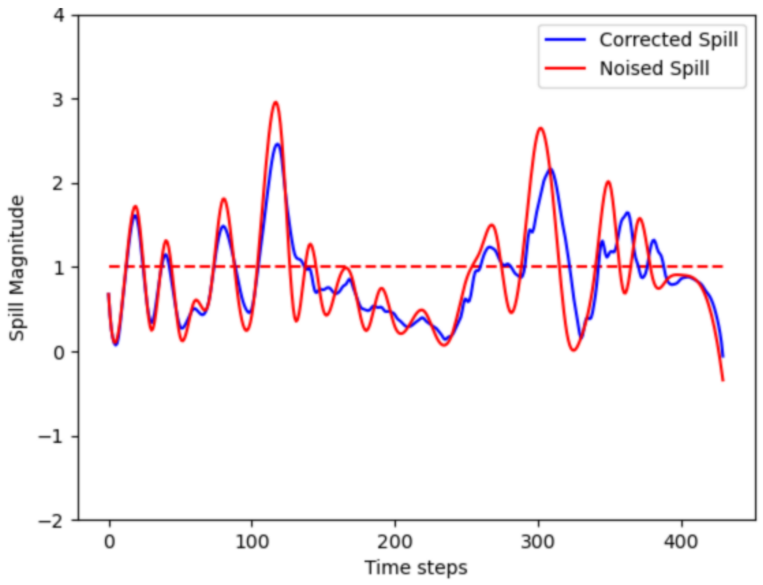

Figure 3.6: A spill sample after trained with RL.

From the figure, we can see that the SDF increases over the training episodes. After around 700 episodes, we can see that the SDF of the corrected spill climbs over the input SDF. And it keeps increasing and converges at aroung the SDF of 0.8, a relatively very good value.

From the figures, we can see that the RL agent trained using SAC is capable of regulate the noised signal in a very effective way. However, there are also limitations using Reinforcement Learning based methods. For example, one challenge is that RL relies heavily on initialization. In our case, the action space is continuous as the RL agent can take an action value that is any real number, which is very time-consuming for the agent to find an optimal path of actions to improve spill regulation.

The hyperparameters and simulation settings for our RL agent training experiments can be found in Appendix B.

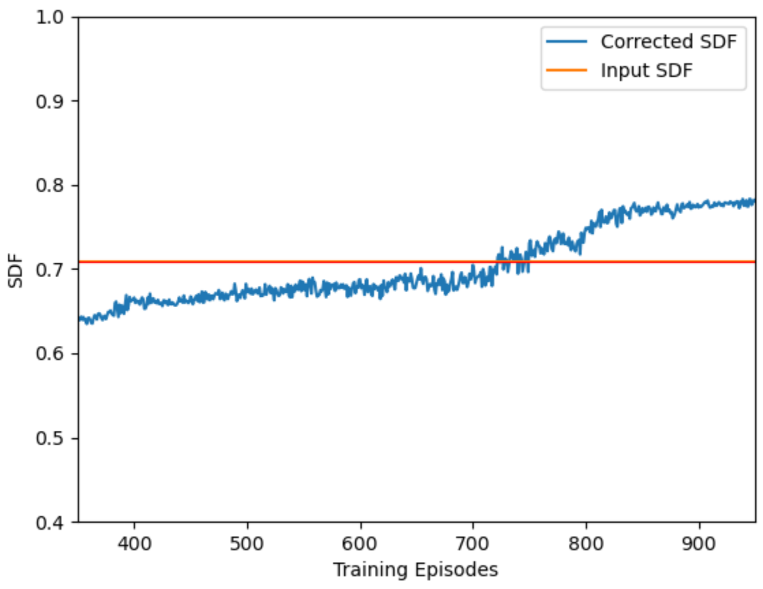

Figure 3.7: SDF of RL agent during training.

CHAPTER 4
CONCLUSION

This paper studies the application of machine learning methods, including Supervised Learning and Reinforcement Learning, to the Spill Regulation System in Fermilab's developing Mu2e experiment. Supported by the results, this paper presented the deep learning based architecture that can replace the PID controllers in simulated experimental setups.

To summarize, there are three main contributions of this paper. They are:

- Analyzed the possibility of replacing the PID controller with neural network based ML models.
- Applied a Supervised Learning based method that minimizes the difference between the model's output and the supervised label.
- Proposed a Reinforcement Learning based method with a smart agent that learns the control signal at each timestamp of the control loop based on real-time observations.

4.1 Future Work

There are many directions for better optimizing the machine learning models for the SRS in Mu2e. For Supervised Learning, we can try different RNN models, such as LSTM. We can also apply Attention Mechanism to our models. For Reinforcement Learning, we can try different action shaping and reward shaping methods when training RL agents.

REFERENCES

[1] L. Bartoszek *et al.*, "Mu2e technical design report," eScholarship, University of California, Tech. Rep., 2015.

[2] K. Seiya, K. J. Hazelwood, H. Liu, S. Memik, M. A. Ibrahim, V. P. Nagaslaev, D. J. Nicklaus, B. A. Schupbach, R. M. Thurman-Keup, and N. V. Tran, "Accelerator real-time edge ai for distributed systems (reads) proposal," *arXiv preprint arXiv:2103.03928*, 2021.

[3] K. Hazelwood, M. Ibrahim, H. Liu, S. Memik, V. P. Nagaslaev, A. Narayanan, D. Nicklaus, P. Prieto, K. Seiya, R. Shi, B. Schupbach, M. Thieme, R. Thurman-Keup, and N. Tran, "Real-time Edge AI For Distributed Systems (READS): Progress On Beam Loss Deblending for the Fermilab Main Injector and Recycler," in *Conference IPAC'21, Campinas, Brazil*, 2021.

[4] M. Thieme, J. Arnold, M. Austin, M. Ibrahim, K. Hazelwood, V. Nagaslaev, A. Narayanan, D. Nicklaus, G. Pradhan, A. Saewert, B. Schupbach, K. Seiya, R. Thurman-Keup, N. Tran, D. Ulusel, H. Liu, S. Memik, and R. Shi, "Semantic regression for disentangling beam losses in the fermilab main injector and recycler," in *Conference NAPAC'22, Albuquerque, New Mexico, United States*, NAPAC, 2022.

[5] J. Berlioz, M. Austin, J. Arnold, K. Hazelwood, P. Hanlet, M. Ibrahim, A. Narayanan, D. Nicklaus, G. Pradhan, A. Saewert, B. Schupbach, K. Seiya, R. Thurman-Keup, N. Tran, D. Ulusel, J. Jiang, M. Thieme, H. Liu, S. Memik, and R. Shi, "Synchronous high-frequency distributed readout for edge processing at the fermilab main injector and recycler," in *Conference NAPAC'22, Albuquerque, New Mexico, United States*, NAPAC, 2022.

[6] A. Narayanan, "Third-integer resonant extraction regulation system for mu2e," Ph.D. dissertation, Department of Physics, Northern Illinois University, 2022.

[7] V. Nagaslaev, J. Amundson, J. Johnstone, C. Park, and S. Werkema, "Third interger resonance slow extraction using rfko at high space charge," *arXiv preprint arXiv:1209.5967*, 2012.

[8] V. Nagaslaev, J. Amundson, J. Johnstone, L. Michelotti, C. Park, S. Werkema, and M. Syphers, "Third interger resonance slow extraction scheme for a mu->e experiment at fermilab," *arXiv preprint arXiv:1207.6621*, 2012.

[9] A. Narayanan, V. Nagaslaev, and M. Syphers, “Third-integer resonant slow extraction from the delivery ring at fermilab,” *Bulletin of the American Physical Society*, vol. 65, 2020.

[10] M. Ibrahim, E. Cullerton, J. Diamond, K. Martin, P. Prieto, E. Scarpine, and P. Varghese, “Preliminary design of mu2e spill regulation system (srs),” Fermi National Accelerator Lab.(FNAL), Batavia, IL (United States), Tech. Rep., 2019.

[11] A. Narayanan, M. Thieme, K. Hazelwood, M. Ibrahim, H. Liu, S. Memik, V. P. Nagaslaev, D. Nicklaus, P. Prieto, K. Seiya, R. Shi, B. Schupbach, R. Thurman-Keup, and N. Tran, “Optimizing Mu2e Spill Regulation System Algorithms,” in *Conference IPAC’21, Campinas, Brazil*, 2021.

[12] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[14] A. Narayanan, M. Thieme, J. Jiang, V. Nagaslaev, J. Arnold, M. Austin, J. Berlioz, P. Hanlet, M. Ibrahim, K. Hazelwood, D. Nicklaus, G. Pradhan, P. Prieto, B. Schupbach, K. Seiya, A. Saewert, R. Thurman-Keup, N. Tran, D. Ulusel, H. Liu, S. Memik, and R. Shi, “Machine learning for slow spill regulation in the fermilab delivery ring for mu2e,” in *Conference NAPAC’22, Albuquerque, New Mexico, United States*, NAPAC, 2022.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[16] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” *arXiv preprint arXiv:1409.1259*, 2014.

[18] A. F. Agarap, “Deep learning using rectified linear units (relu),” *arXiv preprint arXiv:1803.08375*, 2018.

[19] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

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

[21] 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*, vol. 12, 1999.

[22] V. Konda and J. Tsitsiklis, “Actor-critic algorithms,” *Advances in neural information processing systems*, vol. 12, 1999.

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

[24] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *International conference on machine learning*, PMLR, 2018, pp. 1861–1870.

[25] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, “Soft actor-critic algorithms and applications,” *arXiv preprint arXiv:1812.05905*, 2018.

APPENDIX A

MAIN PARAMETERS FOR RESONANT EXTRACTION AT DELIVERY RING

This appendix section provides some of the key parameters of the physics experiment setups for the resonant extraction process at Fermilab's Delivery Ring.

Parameter	Value
Beam kinetic energy	8 GeV
Spill duration	43 ms
Number of spills per super cycle	8
Number of bunches per spill	1
Initial proton intensity	10^{12}
Number of protons extracted per turn	$< 4 \times 10^7$
Time between proton micropulses	1.695 μs
Normalized Emittance (95 % expectation)	16π mm-mrad
Spill Duty Factor (SDF)	> 0.6
Reset time between spills	5 ms

Table A.1: Main Parameters for Resonant Extraction at Delivery Ring

APPENDIX B

HYPERPARAMETERS FOR TRAINING REINFORCEMENT LEARNING MODEL

This appendix section provides the hyperparameters for the training step of the RL agent.

Hyperparameter Name	Value
Discount factor γ	0.999
Learning rate	0.001
Entropy temperature parameter α	0.02
Batch size	32
Target smoothing factor τ	0.005
Hidden layer size	32
Action shaping regulation factor	0.6
Replay buffer size	10000000

Table B.1: RL Hyperparameters