The University Of Sheffield.

General Engineering

Interim Report Assignment Cover Sheet

Surname: WILLIAMSON Forenames: RIK

Registration number 200175672

Module code: GEE401

Module name: INDIVIDUAL ADVANCED PROJECT

Assignment title: Data-driven Reinforcement Learning control of a slot-die manufacturing process trained in a simulated environment

Supervisor: Prof. George Panoutsos

Date submitted: 14/12/23

Student's signature: [illegible] Date: 14/12/23

Final Year Project Interim Report: Data-driven Reinforcement Learning control of a slot-die manufacturing process trained in a simulated environment

Author
Rik Williamson

Supervisor
Prof. George Panoutsos

GEE401 Individual Advanced Project

Department of General Engineering

December 2023

1 Introduction

This Interim report outlines aims and objectives of the project, relevant research through a literature review, the current state of work, reflection on current approach through a Self-review, followed by a Project Management and Planning section detailing steps for optimal project progression. The Literature review focuses on the field of Reinforcement Learning for process control, including contextualisation of Machine Learning and time-series Recurrent neural networks relevant to creation of a dynamic slot-die process simulation environment. Current status of work details exploration of the given experimental data and development of a prototype dynamic LSTM process simulator, including preliminary findings, problems encountered and how they were solved.

2 Aims and Objectives

2.1 Project Title

Data-driven Reinforcement Learning control of a slot-die manufacturing process trained in a simulated environment

2.2 Key aims of the project

- Develop a time-series neural network to dynamically simulate the slot-die manufacturing process
- Create a Reinforcement learning (RL) controller to interact with and learn from the simulated environment
- Use an RL controller to control film thickness of the simulated process through modification of the process conditions

2.3 Tasks

2.3.1 Semester one

Literature Review

- Determine most important slot-die process parameters
- Research Machine Learning model structures
- Research Reinforcement Learning
- Research Reinforcement for process control
- Write Interim Report Literature Review

Modelling

- Create prototype LSTM Process Model

2.3.2 Semester two

Modelling

- Choose Reinforcement Learning algorithm
- Create prototype RL controller
- Evaluate RL Controller
    - Against simplified process
    - Against simplified Process with noise
    - Against Real Process
- Finalise RL controller

Final Report

- Results section
- Discussion section
- Conclusions section

Oral Presentation

- Finalise Literature Review
- Write Slides
- Write Script
- Practice Presentation

For the Gantt Chart, please see Figure 9.

3 Literature Review

3.1 Machine Learning

A field of study since the 60s, Machine Learning (ML) is a subset of Artificial Intelligence (AI) concerned with the development of statistical algorithms that generalise and perform tasks without explicit instructions through experience. This makes ML models most appropriate where there doesn't exist a simple, known process model. Most recently, deep learning systems using gradient-based optimisation and multilayered networks have effectively been used for object classification and speech recognition [1].

Common ML approaches can be classified into 3 categories: Supervised Learning, Unsupervised learning and Reinforcement Learning. With Supervised learning, a data set containing both inputs and the desired output are given and a mathematical model is built to predict an output given new input data. Unsupervised learning uses un-labelled data and attempts to find structure through methods such as classification [2]. The most common form of ML for process optimisation and control is Reinforcement Learning, concerned with the refinement of actions taken by agents in an environment to maximise a reward function, discussed in Section 3.1.2.

3.1.1 Neural Networks (NN)

Inspired by the structure of animal brains, neural networks aggregate input data through interconnected nodes/neurons organised into layers, applying transformations using nodal weights and producing an output through an activation function. During training, weights are iteratively adjusted using algorithms such as gradient descent to minimise a loss function representing the difference between the actual and predicted value.

3.1.2 Reinforcement Learning (RL)

Overview Briefly described in the above section 3.1, Reinforcement relies on the probabilistic description of a system as a Markov Decision process and uses Dynamic Programming algorithms to compute the action leading to the maximisation of a reward function. This differs from conventional supervised and unsupervised ML approaches as no labelled input/out pairs are required and sub-optimal actions do not need to be explicitly corrected. To determine the optimal action given a state s, the actor, a, develops a policy $\pi(s, a)$ through interaction with an environment in discrete time steps where the future reward function at the next time step, r_{t_1} is determined and the policy function π iterated upon to maximise the expected cumulative reward [3]. A value function is also defined, representing the expected accumulative, discounted future reward of a given state-action pair, measuring how 'good' the state is [4].

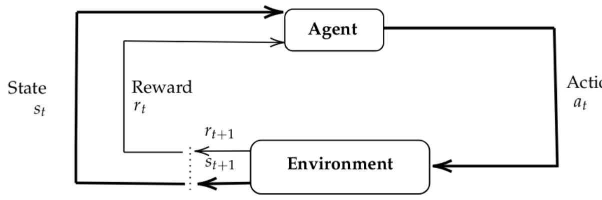

Figure 1: Figure from [5], showing a simplified outline of the RL process

RL for Process Control Recent work in RL for process control centers around model-free, actor-critic architectures, as these combine the advantages of historically used value function based methods and policy gradient methods [5]. This constitutes the creation of an Actor which approximates the optimal policy, and a Critic which estimates the value function for a given state-action pair. Modern State-of-the-Art algorithms primarily use an off-policy approach, such as Deterministic policy gradient (DPG) which takes the form of the expected gradient of the action-value function. To achieve this, policies are assumed to be deterministic instead of stochastic, depending only upon the state-space [6]. Gradient ascent is then performed with respect to policy parameters and the actor network updated.

Building upon this is Deep deterministic policy gradient (DDPG), first developed by [7], which adds the ability to learn from previous actions (experience replay), target networks to address convergence and learning stability and noise exploration to address limited exploration introduced from deterministic policy definition. Target networks are duplicates of the main actor and critic networks which are periodically 'blended' with the learned weights and biases and provide learning stability. Deep Neural Networks (DNNs) are used to represent both the actor and critic functions, mapping the relationships between states, actions and rewards.

Proximal Policy Optimisation (PPO) was introduced by [8] in 2017, and differs slightly from the deterministic policy gradient approaches described previously by being on-policy and stochastic in nature. Despite this, the algorithm achieves good sample efficiency due to its usage of surrogate objectives. These surrogate functions serve as approximations of the extent to which the new policy improves upon the previous one, while clipping policies to prevent large changes. This combines the advantages of Trust Region Policy Optimisation (TRPO) [9], without the associated complexity and difficulty of implementation.

3.1.3 Recurrent Neural Networks (RNNs)

To account for time-series dynamics, RNNs maintain a 'memory' of previous inputs through hidden states with feed-back connections. This allows the output to be dependent upon both past states and the current input [10]. These networks are made up of layers of recurrent cells, which range in complexity; the most simple being a 'standard recurrent sigma cell' (see [10]) with the structure shown in Figure 2. Networks made up of these simple cells, however, are limited in their ability to learn from long-term dependencies. To deal with this problem arising from long-term error signals either blowing up or vanishing, [11] proposed the LSTM cell, which improved the remembering capacity of the cell through introduction of a 'gate'. This original architecture consisted of only an input and output gate, however has since been iterated on by various researchers [12][13][14]. Notably, work by [12] extended the LSTM cell by adding a forget gate and introducing a peephole connection, allowing it to inspect its current internal states and thus learn stable and precise timing algorithms without teacher forcing (see Figure 3) [15]. Since then, endeavours to reduce computational complexity while retaining performance have been performed by [16][17][18], where the configuration, types and number of gates were modified and performance evaluated. These yielded reduced computational requirements, however also performed worse than the original LSTM cell, failing to be taught to count or solve context-free language [10].

LSTM NNs The simplest NN consisting of mostly LSTM cells is the stacked LSTM network, which can be treated as a multi-layer fully connected structure. Due to simple and efficient architecture, it has been used by many researchers to solve complex problems such as prediction of the intent of vulnerable road users by [19]. More complex structures have since been proposed, incorporating bi-directional training [20], multiple dimensions [21] and convolutional connections [22] to better solve specific forecasting problems.

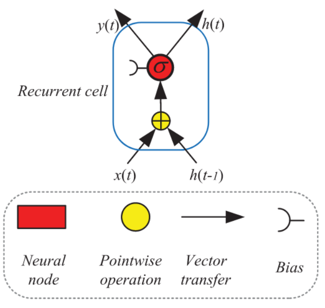

Figure 2: Diagram from [10] showing the structure of a standard recurrent sigma cell

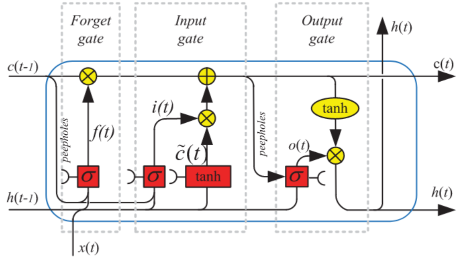

Figure 3: Diagram from [10] showing the architecture of an LSTM featuring input, output amd forget gates, with peepholes introduced by [12]

3.2 Slot-die Thin Film roll-to-roll Manufacturing

3.2.1 Process Overview

Slot-die coating involves the driving of a coating fluid through a slot-die onto a moving substrate to produce a thin film. Typical components of a slot-die roll-to-roll process include: Fluid reservoir, Pump, Slot-die, Substrate mounting system, Coating motion system. It must be noted that the slot-die itself does not interact with the film through any destructive physical contact or scraping [23].

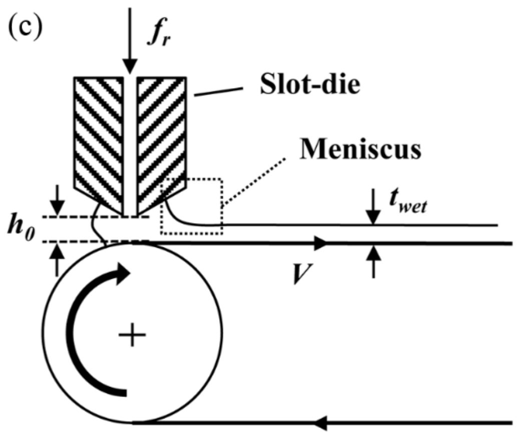

Figure 4: Figure from [23], showing a simplified diagram of the slot-die roll-to-roll process

3.2.2 Performance Parameters

Inputs

- **Volumetric Pump rate (Q):** Increasing this results in increased t_{wet}
- **Coating Velocity (U):** Increasing this reduces t_{wet}

- **Slot-die Gap (x):** Does not affect film thickness, but can impact film quality
- **Coating width (W):** Increasing this lowers T_{wet}

Outputs

- **Wet Film Thickness (t_{wet})**

3.2.3 Film quality control

Film quality depends on parameters from the following categories; coating window effects, downstream process effects and external effects. The coating window is defined as a multivariable map of key process parameters, including slot-die height (x), pump rate (Q), speed of substrate (U), fluid properties of the coating liquid and pressure difference between upstream and downstream faces of the meniscus [24].

4 Current state of work

Inputs	Description
Time	Timestamp in seconds
Velocity	Speed of substrate
Gap	Height of slot-die from substrate
Pump rate	Volumetric pump rate of Ink
Fan	State of fan; 'On' or 'Off'
Ink	Ink type: 4, 4b, 4c
Outputs	
Dynamic width	Value representing film width
Dynamic grey	Value representing film thickness

Table 2: Table showing variables present in the experimental data along with descriptions

4.1 Data Exploration

To gain insight into variables present in the experimental data, description via the Python library Pandas was performed, producing the data shown in Table 1. This revealed a sampling rate of roughly 0.2 s^{-1}, however this was irregular for some indexes. Time-series plots were also produced (see Figure 5), giving insight into the structure of the experiment, including when and how the process parameters were changed and the resulting affect on film thickness. To take a closer look at the dynamics present, Dynamic grey was plotted against time for the period between 260 and 400 seconds (see Figure 6).

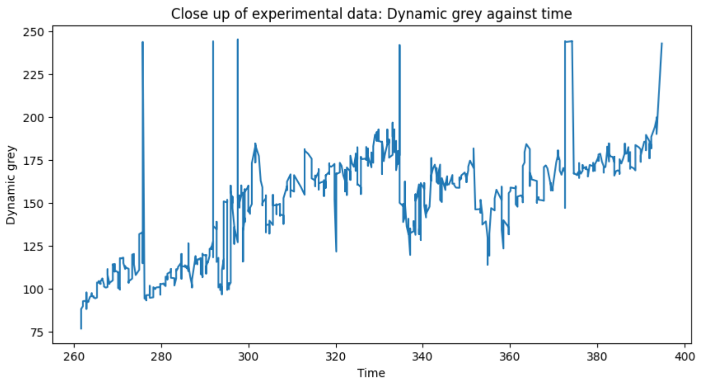

Figure 6: Graph showing smaller range of dynamic grey (film thickness) values from 260 to 400 seconds

4.1.1 Data Pre-processing

To provide a more appropriate input for the LSTM model, data in the 'Fan' column was mapped to booleans, with 0 representing 'Off' and 1 representing 'On'. Categorical label data in the 'Ink' column representing the ink type was encoded using one-hot encoding, creating separate columns for each Ink type (4, 4b and 4c) with boolean values representing Ink type used for the particular data entry. After this, numerical data was scaled by way of linear transform with scikit-learn's MinMaxScaler to fit between zero and one to reduce biases in the network resulting from input data magnitude differences.

4.2 Process Environment Model

4.2.1 Static DNN

First, to gain a deeper understanding of NNs and their practical implementation, a static NN was produced using PyTorch, with inputs being: Velocity of substrate, Pump flow rate, Gap between slot-die and substrate, Ink type and Fan state (on/off) and 'Final grey' as an output. Whilst being useful for gaining experience tuning hyper-parameters of neural networks and teaching the importance of data pre-processing, this is not an appropriate environment to train an RL Controller due to lack of dynamics.

4.2.2 Dynamic LSTM Network

To create a simulation environment for the RL controller, the network must take into account the time-series dynamics of the process, therefore work to create an LSTM network was undertaken. This in-

	Time	Gap	Velocity	Pump rate	Dynamic width	Dynamic grey
count	3836.0	3836.0	3836.0	3836.0	3836.0	3836.0
mean	217.20	0.20482	86.728	9.2129	7.4338	142.71
std	120.10	0.082467	33.681	7.5200	4.0807	29.938
min	0.0	0.1	31.0	0.5	0.04	76.59
25%	119.35	0.1	62.0	3.0	3.41	118.11
50%	221.77	0.2	93.0	8.0	7.655	139.61
75%	300.0	0.3	124.0	14.0	10.53	164.37
max	769.35	0.3	124.0	32.0	23.67	246.96

Table 1: Table showing slot-die process experimental data features of the columns of interest

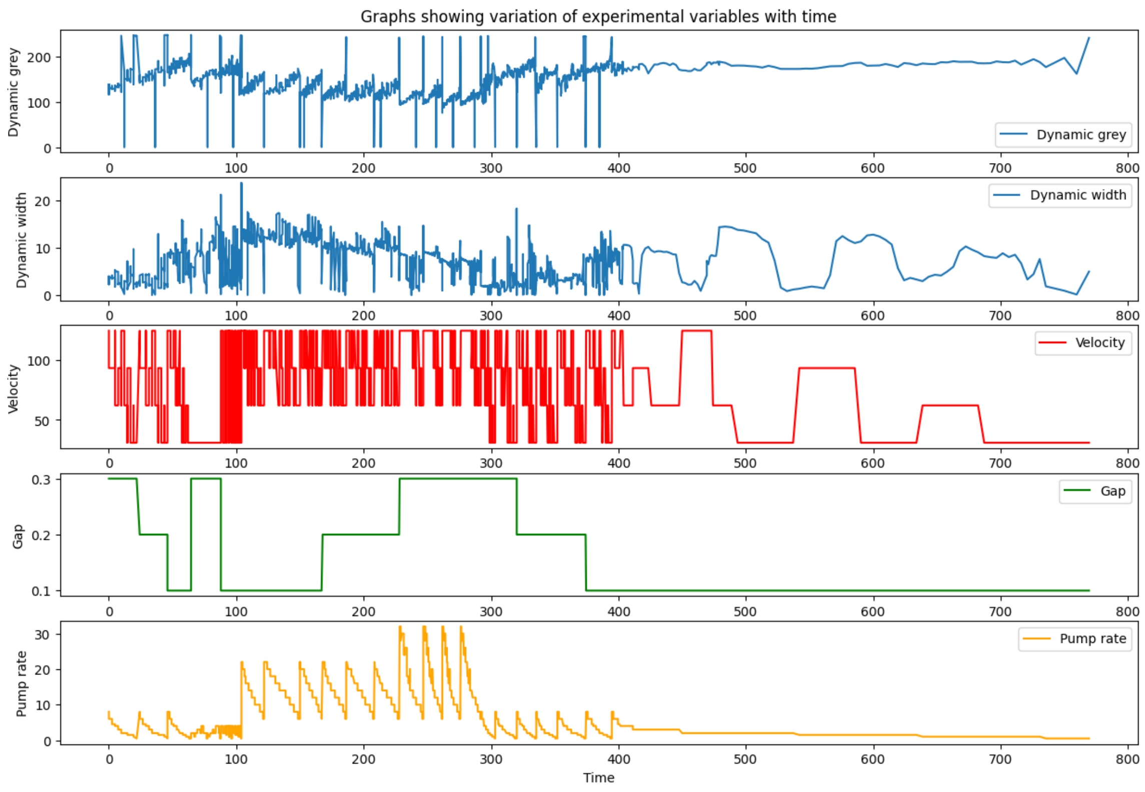

Figure 5: Graphs showing variation of input variables and film thickness (Dynamic grey) with time

volved use of the PyTorch library to create a stacked LSTM network trained with the 'Adam' optimiser and loss defined by Mean Squared Error. Training losses were printed during training and stored in a list to be plotted against epochs after training. Input-output pairs were created in the form of sequences with the length defined by a variable, allowing for systematic change during hyper-parameter tuning.

During training of prototype LSTM networks, problems with over-fitting were encountered, where training loss continually decreased with each loop, however validation loss against the validation dataset increased (see Figure 8). Network structure was altered in an attempt to fix this issue, however input data scaling options seemed to make the biggest difference. Scaling methods trialled were 'MinMaxScaler' and 'StandardScaler' provided by the scikit-learn library, which linearly transform features into the range (0,1) and remove the mean and scale to unit variance respectively. MinMaxScaler resulted in vastly improved validation loss trends, which is likely due to StandardScaler's normalisation being sensitive to outliers, which are present in the training data.

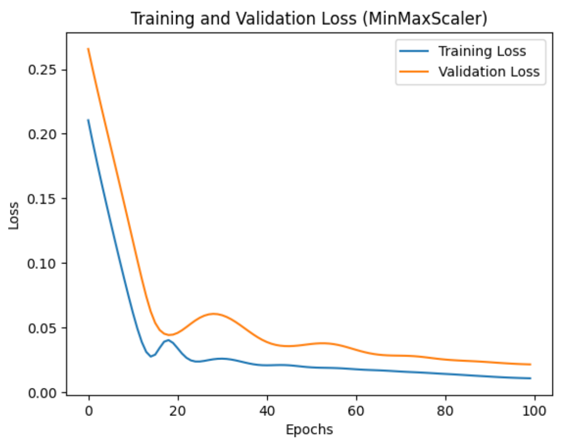

Figure 7: Graph showing training and validation loss for a prototype LSTM process model using MinMaxScaler

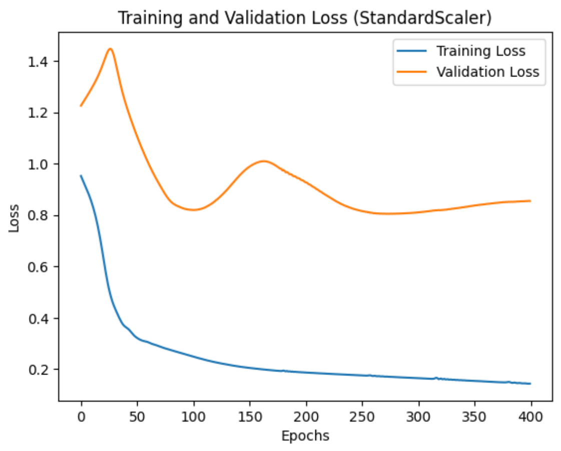

Figure 8: Graph showing training and validation loss for a prototype LSTM process model using StandardScaler

4.3 RL Controller

4.3.1 Problem definition

- **Environment:** LSTM Network Process simulator
- **State space:** Time, Inputs: Velocity, Gap, Pump rate, Fan state. Outputs: Dynamic grey, Dynamic width
- **Action space:** Velocity, Gap, Pump rate, Fan state

4.3.2 Controller details

The DDPG and PPO algorithms described in Section 3.1.2 will be trialled for the RL controller due to their compatibility with continuous state and action spaces, as well as their sample efficiency and relative ease of implementation (see Section 3.1.2).

5 Self Review

The project is progressing well, aligning with the targets set out by the Gantt chart (see Figure 9), despite time pressure resulting from a large number of front-loaded module due-dates. Supervisor meetings were approached with a set of pre-defined questions to make the most of the allotted time, notes taken of key information and next-steps explicitly noted to ensure task accountability. Due to having no prior experience with Machine Learning, the literature review took longer than expected, however resulted the gaining of a comprehensive view of recent advancements in the field and how they relate to process control. Development of a prototype RL controller will prioritised in semester two, as this was part of the original project scope and has not been explored to date.

Progress towards an initial prototype process simulator has been somewhat of a meandering path due to premature pursuit of a static rather than dynamic NN model. This was in part due to poor initial exploration of the experimental dataset resulting in incorrect assumptions being made, combined with the task being absent from the initial project brief, instead proposed by myself to gain experience working with NNs. Despite this slightly de-railing progress towards an initial prototype, development of these early models provided deeper understanding of these structures, which are integral parts of many state-of-the-art RL algorithms (see Section 3.1.2) which will likely be used later in the project.

Building an initial LSTM Network has given a deeper understanding of the effect of hyper-parameters on network performance, as well as how to navigate non-ideal data.

6 Project Management and Planning

6.1 LSTM Process Simulator future work

6.1.1 Data processing

During the creation of preliminary models, it was found that scaling methods significantly impacted loss convergence (see Figures 7 and 8), therefore different normalisation methods will be tested to determine the most appropriate.

The cleaning of data will be attempted by removing clipped max and min values seen in the 'Dynamic grey' graph (see Figure 5) subject to more information about the measurement process behind these values.

The effect of data-interpolation will be investigated, to standardise the sampling rate and provide more information to the model. This would allow for more consistent generation of input-output pair sequences (see Section 4.2.2) and could improve model accuracy. The 'Dynamic grey' and 'Dynamic width' measurements will also be investigated further to ensure interpolation is a valid assumption.

6.1.2 Hyper-parameter tuning

The effect of hyper-parameter tuning will be explored through systematic modification of the number of layers, number of nodes and the look-back window will be performed to identify trends and optimal values.

6.2 RL controller future work

To progress towards a prototype RL controller, further research into the practical implementation of algorithms discussed in Section 3.1.2 will be undertaken. If a sufficiently accurate LSTM environment is not developed, the existing pre-made state-space black-box model will be used to explore the different algorithms discussed previously.

Problems with environment uncertainty are anticipated given variability of the experimental data (see Figure 5), therefore models with robust policy iteration under these conditions will need to researched. This could include searching for papers where the authors are tackling similar problems and building upon their approach.

7 Appendix

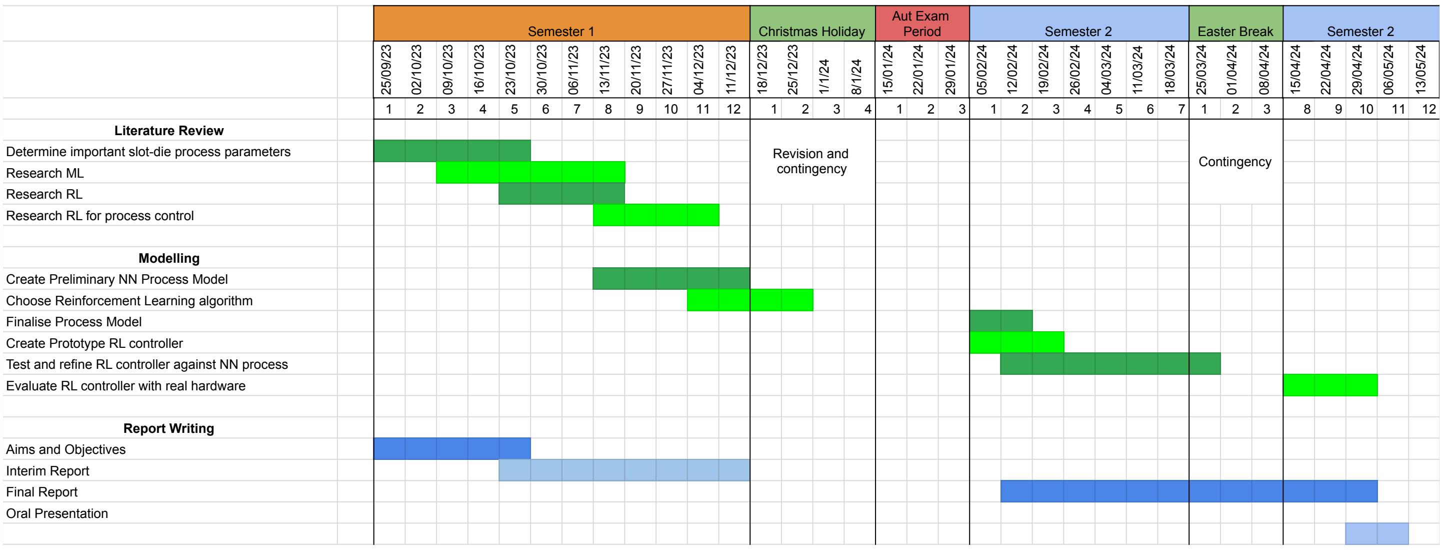

Figure 9: Gantt chart showing time allocation for tasks required for completion of the project

References

[1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, Jul. 17, 2015, ISSN: 0036-8075, 1095-9203. DOI: `10.1126/science.aaa8415`. [Online]. Available: `https://www.science.org/doi/10.1126/science.aaa8415` (visited on 11/29/2023).

[2] Z. Ghahramani, "Unsupervised Learning," in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, ser. Lecture Notes in Computer Science, O. Bousquet *et al.*, Eds., Berlin, Heidelberg: Springer, 2004, pp. 72–112, ISBN: 978-3-540-28650-9. DOI: `10.1007/978-3-540-28650-9_5`. [Online]. Available: `https://doi.org/10.1007/978-3-540-28650-9_5` (visited on 11/29/2023).

[3] K. Arulkumaran *et al.*, "Deep Reinforcement Learning: A Brief Survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, Nov. 2017, ISSN: 1558-0792. DOI: `10.1109/MSP.2017.2743240`. [Online]. Available: `https://ieeexplore.ieee.org/abstract/document/8103164` (visited on 11/30/2023).

[4] Y. Li. "Deep Reinforcement Learning." arXiv: `1810.06339 [cs, stat]`. (Oct. 15, 2018), [Online]. Available: `http://arxiv.org/abs/1810.06339` (visited on 11/30/2023), preprint.

[5] R. d. R. Faria *et al.*, "Where Reinforcement Learning Meets Process Control: Review and Guidelines," *Processes*, vol. 10, no. 11, p. 2311, 11 Nov. 2022, ISSN: 2227-9717. DOI: `10.3390/pr10112311`. [Online]. Available: `https://www.mdpi.com/2227-9717/10/11/2311` (visited on 11/26/2023).

[6] D. Silver *et al.*, "Deterministic Policy Gradient Algorithms,"

[7] T. P. Lillicrap *et al.* "Continuous control with deep reinforcement learning." arXiv: `1509.02971 [cs, stat]`. (Jul. 5, 2019), [Online]. Available: `http://arxiv.org/abs/1509.02971` (visited on 12/10/2023), preprint.

[8] J. Schulman *et al.* "Proximal Policy Optimization Algorithms." arXiv: `1707.06347 [cs]`. (Aug. 28, 2017), [Online]. Available: `http://arxiv.org/abs/1707.06347` (visited on 12/11/2023), preprint.

[9] J. Schulman *et al.* "Trust Region Policy Optimization." arXiv: `1502.05477 [cs]`. (Apr. 20, 2017), [Online]. Available: `http://arxiv.org/abs/1502.05477` (visited on 12/11/2023), preprint.

[10] Y. Yu *et al.*, "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, Jul. 1, 2019, ISSN: 0899-7667. DOI: `10.1162/neco_a_01199`. [Online]. Available: `https://doi.org/10.1162/neco_a_01199` (visited on 12/11/2023).

[11] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: `10.1162/neco.1997.9.8.1735`. [Online]. Available: `https://ieeexplore.ieee.org/abstract/document/6795963` (visited on 12/11/2023).

[12] F. A. Gers *et al.*, "Learning to Forget: Continual Prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, Oct. 1, 2000, ISSN: 0899-7667. DOI: `10.1162/089976600300015015`. [Online]. Available: `https://doi.org/10.1162/089976600300015015` (visited on 12/11/2023).

[13] R. Jozefowicz *et al.*, "An Empirical Exploration of Recurrent Network Architectures,"

[14] J. Schmidhuber *et al.*, "Training Recurrent Networks by Evolino," *Neural Computation*, vol. 19, no. 3, pp. 757–779, Mar. 1, 2007, ISSN: 0899-7667. DOI: `10.1162/neco.2007.19.3.757`. [Online]. Available: `https://doi.org/10.1162/neco.2007.19.3.757` (visited on 12/11/2023).

[15] F. A. Gers *et al.*, "Learning Precise Timing with LSTM Recurrent Networks,"

[16] J. Chung *et al.* "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling." arXiv: `1412.3555 [cs]`. (Dec. 11, 2014), [Online]. Available: `http://arxiv.org/abs/1412.3555` (visited on 12/11/2023), preprint.

[17] R. Dey and F. M. Salem, "Gate-variants of Gated Recurrent Unit (GRU) neural networks," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug. 2017, pp. 1597–1600. doi: 10.1109/MWSCAS.2017.8053243. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8053243 (visited on 12/11/2023).

[18] G.-B. Zhou *et al.*, "Minimal gated unit for recurrent neural networks," *International Journal of Automation and Computing*, vol. 13, no. 3, pp. 226–234, Jun. 1, 2016, issn: 1751-8520. doi: 10.1007/s11633-016-1006-2. [Online]. Available: https://doi.org/10.1007/s11633-016-1006-2 (visited on 12/11/2023).

[19] K. Saleh *et al.*, "Intent prediction of vulnerable road users from motion trajectories using stacked LSTM network," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2017, pp. 327–332. doi: 10.1109/ITSC.2017.8317941. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8317941 (visited on 12/11/2023).

[20] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997, issn: 1941-0476. doi: 10.1109/78.650093. [Online]. Available: https://ieeexplore.ieee.org/document/650093 (visited on 12/12/2023).

[21] A. Graves *et al.*, "Multi-dimensional Recurrent Neural Networks," in *Artificial Neural Networks – ICANN 2007*, J. M. de Sá *et al.*, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2007, pp. 549–558, isbn: 978-3-540-74690-4. doi: 10.1007/978-3-540-74690-4_56.

[22] X. Shi *et al.* "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting." arXiv: 1506.04214 [cs]. (Sep. 19, 2015), [Online]. Available: http://arxiv.org/abs/1506.04214 (visited on 12/12/2023), preprint.

[23] J. Park *et al.*, "Roll-to-Roll Coating Technology and Its Applications: A Review," *International Journal of Precision Engineering and Manufacturing*, vol. 17, no. 4, pp. 537–550, Apr. 2016, issn: 2234-7593, 2005-4602. doi: 10.1007/s12541-016-0067-z. [Online]. Available: http://link.springer.com/10.1007/s12541-016-0067-z (visited on 10/27/2023).

[24] *Slot-die coating*, in *Wikipedia*, Jul. 28, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Slot-die_coating&oldid=1167591419 (visited on 11/08/2023).