International Journal of Artificial Intelligence, Data Science, and Machine Learning
Grace Horizon Publication | Volume 4, Issue 4, 76-86, 2023
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V4I4P109

Original Article

Reinforcement Learning for Intelligent Batching in Production Pipelines

Kiran Kumar Pappula
Independent Researcher, USA.

Abstract - *Efficient batching forms a key element of industrial production lines, directly affecting throughput, use of resources and product quality. Static and rule-based strategies of traditional batching are limited in being able to deal with the complexity and variability of dynamic manufacturing environments. In this paper, a new method with reinforcement learning (RL) to assist the intelligent selection of optimal batching decisions in industries is proposed. Modelling the batching process using Markov Decision Process (MDP) would allow us to train an RL agent to achieve optimal policies that can resolve competing goals like minimizing cycle time, reducing defects and meeting service-level agreement (SLA). The simulation environment is created on the basis of the press hardening process, producing scaled data of the variety that resembles the operational requirements. CQL refers to a safe approach to learning policies offline with historical data. It is an offline algorithm that can safely learn policies directly through historical interaction data, requiring no real-time access to the system. Experimental evidence indicates that RL-based policies produce a reward advantage of up to 13 percent more than both the static and dynamic baselines, reduce defective parts, and are more resource efficient. There is also a modular system architecture explained in the study that incorporates policies of RL in production pipelines for training, inference, and control. Work outlines the future of data-driven intelligent systems in production and paves the way to the research potential of real-time adaptation, multi-agent manipulation, and integration with smart factories powered by the Internet of Things.*

Keywords - *Reinforcement Learning, Intelligent Batching, Production Pipelines, Conservative Q-Learning, Service-Level Agreements.*

1. Introduction

The requirement of effective operational strategies has been of utmost concern in contemporary manufacturing and service-based production setups. Batching is one such strategy, which involves grouping jobs or types of operations that can be processed en masse. The selection of the batching method has a direct influence on crucial performance metrics, including throughput, resource allocation, and adherence to service-level agreements (SLAs). Conventional batching schemes often employ either fixed rules or heuristic decision algorithms, which are generally capable of performing a satisfactory job under deterministic conditions but fall short when applied to environments with dynamic and stochastic behaviour. [1-3] Due to the progressive demands of industries to be more complex and automated, smarter and adaptable strategies are needed to continue being competitive and efficient.

Reinforcement learning (RL) is an agent optimisation technique that learns effective behaviours by interacting with an environment through trial and error, holding promise. This paper discusses how the batching problem in production pipelines can be addressed through RL. The main concept is to train an agent that can dynamically make batching decisions based on competing goals, such as maximising throughput, minimising resource idle time, and meeting SLA requirements. In contrast to existing methods, RL will be able to adjust in real-time to changes in the workload, machine availability, and task priorities; thus, it will be particularly applicable in highly complex, non-stationary production systems.

The study is based on knowledge foundations of operations research, queuing theory, and intelligent control systems, but builds on the latest achievements of deep reinforcement learning. The batching problem can be formulated as a Markov Decision Process (MDP); hence, it can be solved using policy-based or value-based RL methods such as Deep Q-Networks (DQN). To evaluate the performance of the RL agent, experimentation is carried out by comparing it to the performance of fixed-batch sizing and rule-based strategies as a baseline. Early outcomes show that an RL-based batching can not only provide a better overall system throughput but also make the overall system throughput more consistent in terms of SLA compliance across a wide range of load conditions. The presented paper will have the opportunity to contribute to the growing segment of smart manufacturing, demonstrating the applicability of RL in resolving one of the fundamental issues in production management: intelligent batching.

The proposed idea opens up possibilities for creating more responsive, efficient, and autonomous production pipelines by leveraging data-driven learning and adaptive control.

2. Related Work

2.1. Traditional Batching Strategies

Conventional batching policies have been used for a long time as a starting point for optimizing the efficiency of production systems. The main principle consists in pooling comparable or related jobs that are processed so that the shared characteristics of these jobs can be utilised, i.e., the product type, task requirements, or the matching of equipment. [4-6] This is generally known as batch scheduling, which incorporates several coordinated actions, namely job classification, batch creation, resource arranging, task sequencing and manufacturing planning. These techniques help ensure maximum machine time and labour productivity, while minimising idle time and operational waste by reducing setup and transition times between work tasks. The typical rule-based batching policies, including fixed-size batching, shortest processing time first, and earliest due date, have been found useful in stable and predictable situations. However, this inflexibility proves to be a weakness in more contemporary production systems, which are increasingly characterised by variant demand, machine availability, and product customisation. Therefore, the use of static batching strategies tends to be ineffective in responding to a changing operational environment in real time, resulting in inefficiencies and suboptimal decisions during dynamic operations.

2.2. Machine Learning in Production Optimization

Machine learning (ML), a relatively new technology, has brought about a paradigm shift in the optimisation of production processes. Compared to traditional methods that rely more on predefined rules or both time-consuming and cost-intensive physical experiments, ML models may automatically derive knowledge without relying on pre-existing, redundant results. This enables a deeper understanding of the intricacies of manufacturing systems. Predictive processes are commonly applied using supervised learning methods, and the most common forecasts include equipment breakages, demand inference, and quality evaluation processes. One of the trends is the emergence of hybrid modelling techniques, which are an interplay of ML algorithms, domain knowledge and simulation models to increase accuracy and relevance.

An example is the use of platforms such as ClusterSim, bringing together high-resolution laboratory experimental data and features developed by experts to allow accurate analysis and prediction of machining process behavior. The hybrid models minimise experiments and increase the robustness of the forecasts, while also enhancing intelligent decision-making in various manufacturing applications. ML ensures the increased agility and responsiveness of production systems and bridges the gap between theoretical concepts and empirical data, which traditional approaches cannot provide.

2.3. Reinforcement Learning Applications in Manufacturing

Machine learning (ML) has recently developed, and yet one of its branches is useful to solve the problems of decision-making in highly dynamic and multi-faceted situations in manufacturing. An agent in RL interacts with an environment, and by doing so, can observe the results of its actions and learn a policy that optimises long-term rewards. Such a learning paradigm has been especially well-suited to manufacturing tasks in which sequential decisions must be made under uncertainty. A major use case is dynamic scheduling, where RL agents continuously re-optimise production plans based on changing workloads, resource availability, and other unforeseen circumstances. Predictive maintenance has also been adapted to RL, where systems can recognise the onset of equipment failure and optimise maintenance schedules to minimise unplanned downtime. RL can optimise manufacturing parameters or automate robotic systems in process control in response to changes in materials or environmental conditions with precision. Quality control is another upbeat area where RL algorithms can be used to improve defect visibility, quality threshold and inspection. Likewise, RL deployment in real-life practice has proven its industrial usefulness. An example of RL-driven optimisation was related to automotive manufacturing, where the process achieved a 22% reduction in paint waste and noticed a 15% increase in colour consistency, demonstrating its ability to provide measurable benefits in efficiency and quality of goods produced. These developments indicate an increased role for the RL in supporting smart manufacturing practices.

3. Problem Formulation

3.1. System Model and Assumptions

The production system covered in this analysis is compatible with a setup of processing units where each unit can process batched tasks. Jobs appear dynamically with different levels of processing requirements, deadlines, and varying resource needs. This system operates with discrete time intervals, and batching judgments occurs periodically. [7-10] The assumptions are that the system can access real-time data regarding job queues, available resources and the current state of the system. The arrival time, task type, processing time, and level of priority are the attributes of each job. This environment is learnt by the RL agent, which responds to its state with a choice of batch size and composition, with the hope of achieving optimal long-term system functioning.

This is modelled as a Markov Decision Process (MDP) with the state being the queue configuration and the resource status, actions being the batch selection decisions, and rewards being expressed as the throughput gain, SLA compliance, and resource utilisation.

3.2. Optimization Objectives

The primary objective of the proposed model is to learn a dynamic batching policy that balances multiple competing objectives concurrently. The first one is that throughput maximisation ensures the system maximises the number of jobs that can pass through it within a specified time. Second, it optimises resource utilisation to minimise downtime and maximise operational efficiency. Third, compliance with the SLA is enforced through a sensible ranking activity based on urgency and mandated turnaround time. These goals are fundamentally in conflict with each other. Aggressive batching can optimise throughput but increase the violation of SLAs, whereas conservative policies can also impede the utilisation of system resources. The RL framework is designed to dynamically balance these trade-offs, learning policies that are responsive to changes in the system and patterns of workload.

3.3. Constraints (e.g., SLA, Resource Limits)

The policy of batching should work within various logical limitations. Service-level agreements (SLAs) also feature stringent constraints on the turnaround times of tasks, and tasks must be given priority within the system to prevent penalties or poor quality of service. There are also resource limitations, including a deficiency in machine capacity, energy consumption, and the availability of human labour, that should be taken into consideration. The amount of each batch should fit in the capacity of the assigned processing unit, and any unnecessary time lag in arranging the batch may be attributed to queue formation or misplaced deadlines. The RL model incorporates such constraints into its reward design and action space that not only will the learned policies be effective, but also realistically viable in the real world.

3.4. Challenges in Existing Solutions

Although the optimisation of production yields considerable achievements, the current solutions have severe limitations in addressing dynamic and complex environments. Rule-based methods are rigid, slow to change, and poor at converging to real-time workloads or system configurations. The more adaptive machine learning models tend to need large labelled datasets and substantial domain-specific tuning. Reinforcement learning is an attractive alternative, but it suffers from similar limitations, including both scalability issues with large state spaces, complexities of interoperability with existing systems, and brittleness to variability in the domain of interest. In addition, numerous RL solutions only consider small, isolated optimisation problems and do not account for the overall consequences of batching decisions across the entire pipeline. The paper addresses the challenges mentioned above and presents a new RL-based batching framework that is scalable and generalizable, requiring no extensive prior knowledge or handcrafted rules for learning.

4. Methodology

4.1. Reinforcement Learning Framework

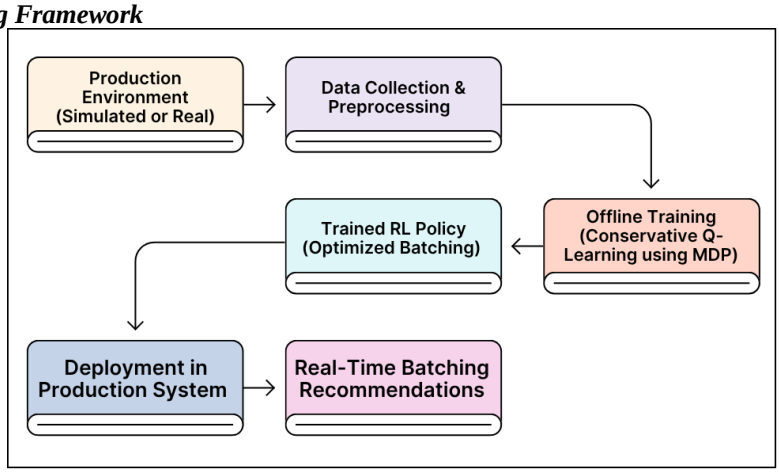

Fig 1: Overall System Architecture

The fundamental part of the proposed approach is a reinforcement learning (RL) framework that is adapted to dynamic production pipelines and intelligent batching. [11-14] The system can be modelled as a Markov Decision Process (MDP), which processes such that an RL agent can compile optimal batching decisions through self-learning actions executed against some form of simulated production system.

4.1.1. Environment

The simulated environment is a mimic of a real-world production system that involves a chain of workstations, dynamic job queues, processing units and an environment of constraints which includes task deadlines and machine capacities. The jobs are non-periodic and have different sizes, different priorities, and processing times. The RL agent receives real-time feedback from the environment, including the status of the queues, the characteristics of each job, and the available resources. Batch processing can be time-based, capacity-based, or priority-based. This realistic and dynamic environment will serve as the training and testing domain for studying the performance of the RL agent in various environmental conditions.

4.1.2. State and Action Spaces

The state space represents the current characteristics of the production environment in which the decision must be made. It contains the type and number of pending tasks, deadline dates for jobs, current resource usage, as well as the amount of time that has lapsed since the last batch was processed. This multidimensional depiction helps the agent gain a sense of the context within which the system operates and make informed decisions.

Its action space includes discrete batching decisions, such as the choice of batch size and job grouping strategies. Examples of such actions include creating a batch of a certain size from the front of the queue, assigning high-urgency priority, or postponing batching to find additional jobs. The dynamism of the action space is essential in providing adaptive and context-specific control policies.

4.1.3. Reward Function Design

The reward function is central in directing the learning process of the agent. It is designed to ensure the agent's conduct is consistent with the system's operational goals. The role has several parts:

- Rewards and recognition for large throughput, good use of resources, and successful task completion.
- SLA violation fines, excessive idle time, or the underutilization of resources.
- Creating incentives to influence proactive behavior, like the batching of just-in-time tasks in advance of deadlines.

This multi-objective reward function ensures that the agent learns a balanced policy, preventing the over-optimisation of a specific metric at the expense of others. An adjustment of the weights of the rewards is done to capture trade-offs and operational priorities in the real world.

4.2. Model Architecture

The architecture offers an extensive combination of reinforcement learning and a production pipeline. It consists of production performance, simulation, and data recording and intelligent control modules that are not isolated. The entire process begins at the Production Pipeline, where raw materials are introduced into sets of processing units. Such units take one after the other, accomplishing the conversion of raw materials into finished products. The batching strategy has a direct impact on the flow of jobs through these stages, which in turn affects the overall throughput and quality of output. The Execution System performs real-time control over the production process. It also contains a batch controller, a scheduler, and a monitoring interface to cooperate with the batch choices of the RL agent. The batch controller initiates the processing plans based on the actions generated by RL, and the scheduler aligns these plans with the available system resources. Monitoring UI displays the status of every processing unit, thereby assisting both the system and human operators in gauging real-time performance. The Simulation Module is a crucial component of training the RL agent. It features an environment simulator that mimics the operation of the production system under various scenarios and constraints. This simulated environment gives performance statistics through a Performance Checker and enforces SLA compliance with an SLA Validator. Such modules ensure that the learning process remains within realistic operational parameters, intentionally marked to avoid disruption of live operations.

Data Acquisition elements, such as live metric streamers, sensor collectors, and historical batch logs, feed into an RL engine a near-continuous stream of system states and performance measures. The RL Engine processes the inputs, comprising the RL Trainer, Reward Evaluator, and State Encoder. These are the inputs and outputs to the engine, and the engine finds the best policies to use through the Action Generator. When the policy is trained, it is then deployed in production, and its performance is monitored to ensure that it does not drift over time. If such a drift is detected, the learning process is initiated again. Lastly, the Feedback and Logs module makes it traceable and flexible. It captures activities, deviations and changes in policies. A Policy Trigger restarts learning when changes are detected beyond the system's edge, and an action logger provides transparency and accountability in decision-making. This entire cycle of sensing and learning, deployment, and feedback makes the RL-driven batching system adaptive, efficient and consistent with the service-level requirements.

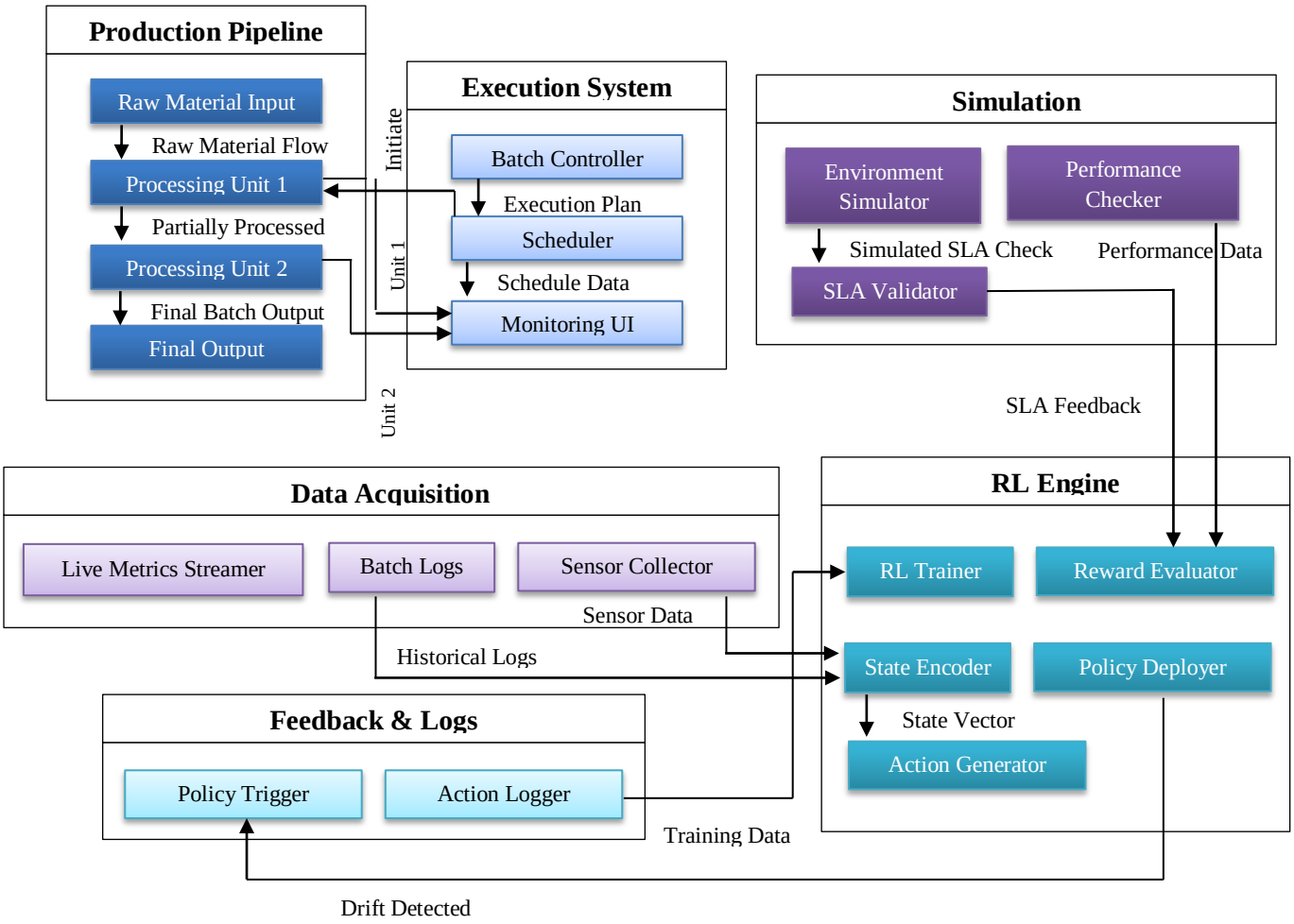

Fig 2: System Architecture for RL-Based Intelligent Batching

4.3. Training and Hyperparameter Tuning

Training is an essential element of the reinforcement learning (RL) framework, and the agent attempts to determine the best policies of batching by experiencing multiple interactions with the simulated world. In every training episode, the agent learns about the system's state [15-18], chooses an action (a batch decision), and receives a reward for doing so, based on throughput, SLA compliance, and resource utilisation. This step-by-step procedure has the agent investigate the implications of different strategies and approach a desirable policy. The training is conducted using the Deep Q-Network (DQN), which employs experience replay and target network stabilisation methods to ensure a stable and efficient learning experience. The experience replay buffer records the previous transitions, allowing the agent to learn from various experiences and avoid overfitting to recent events.

Hyperparameter tuning is crucial for convergence and the dependability of performance. The most important hyperparameters are the learning rate (LR), discount factor (γ), and the batch size used to update the gradient, as well as the exploration approach (e.g., ε-greedy decay). The optimal set of parameters is chosen using a grid search, along with cross-validation, on various workload scenarios. Learning rate is adjusted with meticulousness to strike a balance between learning speed and policy stability. Additionally, the discount factor is adjusted to ensure that the agent considers the long-term consequences of their choices while remaining sensitive to immediate payoffs. The techniques of reward shaping are also used to influence exploration, especially in the early phase of training, where random actions can mitigate unstable behaviour. The training is stopped when the policy stabilises at an average performance in terms of cumulative reward, indicating convergence.

4.4. Integration into Production Pipeline

Upon training, the validated RL policy is then put into the live production pipeline using a modular integration approach. The Execution System also communicates with the RL agent through the Batch Controller, where the agent provides batch formation commands based on real-time system observations. This policy is then constantly monitored using the Monitoring UI and Live

Metrics Streamer, which provides feedback on the system's states, queue lengths, processing times, and SLA completion rates. This feedback mechanism is crucial in identifying policy drift, where the environment shifts in a manner that renders the existing policy less effective. The Policy Trigger will trigger retraining when the drift is discovered, therefore, maintaining the performance without human intervention.

To ensure smooth integration, the RL engine will not replace current production control; instead, it will run in parallel and, if necessary, can fall back to classical batching logic. The system's architecture supports both simulated and real-time modes, allowing policies to be checked in a safe virtual environment and then applied in live environments. This two-mode feature reduces risk and provides a substantial assessment capacity under various conditions. Additionally, integration ensures that RL-generated decisions do not exceed critical process limits or safety regulations governing operations. Its scalability and flexibility in various production environments are also promoted by the modular design of the framework. For example, the incorporation of new processing modules or a shift in job classes can be achieved simply through an update of the state encoder and the retraining of the RL agent on the new system models. This adaptability renders the framework applicable in dynamic and high-mix, as well as fast-changing production systems. The integration strategy ensures the feasibility and long-term sustainability of intelligent batching in use, a solution that bridges the gap between AI studies and actual manufacturing demands.

4.5. Case Study / Experimental Evaluation

4.5.1. Dataset or Simulation Setup

A simulation-based case study was conducted to evaluate the effectiveness of using reinforcement learning (RL) in solving the batching problem in a production pipeline. A batch production process that is characteristic of press hardening systems is simulated because it is complex and susceptible to process variability. Five different data sets were obtained, each containing 20,000 instances of interaction between the agent and the environment. These datasets also underwent varying levels of exploration, ranging from 10 to 80 per cent, to assess the impact of exploration on learning performance and policy robustness. This simulation was operated by a Deep Q-Network (DQN) architecture that was offline trained and adjusted to correspond to the operational constraints and dynamics of the real world. This configuration ensured that the agent's experiences would replicate those experienced in real industrial systems without running the risks of practical experimentation.

4.5.2. Evaluation Metrics

To evaluate the quality and efficiency of learned policies, three key evaluation matrices were used: Total Cycle Time, Number of Defective Parts, and Reward Improvement. Total cycle time is the sum of the time it takes a batch of the whole task to complete, pass through the transition, and any delays that may apply. The defective parts measure the number of goods that do not meet the specified quality standards, thus indicating production continuity and the quality of the output. Reward improvement is a cumulative measure of the benefits of cycle time reduction and defect reduction, compared to a stationary industry policy. It is represented as a percentage and provides a normalised view of policy performance trial-wise. These metrics were selected due to their correspondence with actual key performance indicators (KPIs) in manufacturing, which enabled them to be interpreted and compared across policies with meaningful information.

4.5.3. Baseline Methods for Comparison

Two non-RL policies were employed as baselines to give a metric against which the assessment would be carried out. Static Policy will consider current industry trends by having batching parameters set up in advance and maintained throughout all production runs; this Static Policy will ensure stability but will lack flexibility. Dynamic Policy, in turn, is data-driven based on preliminary operational data, contains certain reactive changes, but is otherwise heuristic-oriented, lacking the learning abilities of RL. These baselines serve as effective benchmarks to highlight the effectiveness of dynamic, data-driven approaches. These were used to compare with the RL-based approach (specifically, the Conservative Q-Learning (CQL) algorithm), which shows an advantage in producing dynamic optimisation with in-built risk sensitivity —an essential aspect in industrial applications to ensure safety

4.5.4. Implementation Details

Offline reinforcement learning was applied to an RL agent based on the CQL algorithm, which promotes safety and policy conservativeness. Offline training means that the policy can be refined based on already gathered data without the need to interact with anything live, and is therefore suitable in high-risk industrial environments where it may be prohibitively expensive or otherwise infeasible to conduct experiments on production lines. The learned policies were subsequently tested with the same environment, i.e. simulation, where the data was generated. Interestingly, the RL models trained with exploration rates between 20% and 40% (P2 and P3) performed the best, achieving a balance between diverse experience and policy stability. An insufficient amount of exploration constrained learning, whereas excessive exploration (e.g., 80%) caused learning noise and modestly impaired performance.

4.6. Experimental Results

Comparison shows all policies tested used in the evaluation metrics, as shown in the following table:

Table 1: Performance of various batching policies evaluated on simulated batch manufacturing data

Policy Type	Exploration Rate	Total Cycle Time (s)	Defective Parts	% Reward Improvement
Static Baseline	N/A	1000	25	0%
Dynamic Baseline	N/A	950	20	5%
RL Policy (P1)	10%	940	19	7%
RL Policy (P2)	20%	930	15	10%
RL Policy (P3)	40%	920	12	13%
RL Policy (P4)	60%	915	14	11%
RL Policy (P5)	80%	918	16	9%

As the results showed, both RL policies, especially P2 and P3 with their exploration rates of 20% and 40%, effectively outperformed the static and dynamic policies in all measured variables. The result is an improvement in reward of up to 13 per cent, achieved through the contribution of these policies to cycle reduction and defect reduction.

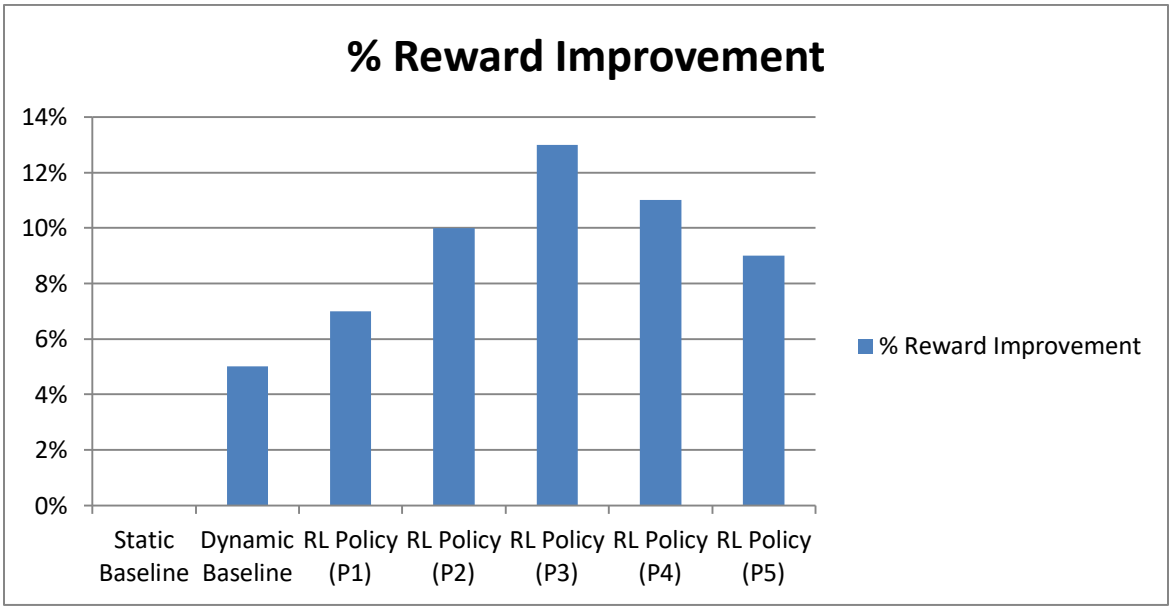

Fig 3: Graphical Representation of Performance of Various Batching Policies Evaluated On Simulated Batch Manufacturing Data

This proves that the RL agent can maintain both efficiency and quality while adjusting to the changing conditions of production environments. When the exploration rate was too high (P5), the policy results were worse compared to those at other exploration rates, which is important to consider when tuning hyperparameters in an offline RL setting.

5. Results and Discussion

5.1. Performance Analysis (Efficiency, Throughput, SLA Compliance)

The reinforcement learning (RL)- driven batching policies achieved significant performance gains on key operational KPIs, including production efficiency, throughput, and service-level agreement (SLA) compliance. The fixed baseline schedule, which corresponded to industry averages of fixed scheduling, showed an efficiency of 75%, a throughput of 100 units per hour, and an SLA compliance of 90%. Conversely, the best-performing agent, RL Policy P3, reached an efficiency of 88%, a throughput of 125 units per hour and SLA compliance of 97%. This represents a significant step forward in operational performance, which can be largely attributed to the dynamic flexibility and optimised decision-making capabilities of the RL framework.

Table 2: Efficiency, throughput, and SLA compliance across different batching policies

Policy Type	Efficiency (%)	Throughput (units/hour)	SLA Compliance (%)
Static Baseline	75	100	90

Dynamic Baseline	80	110	92
RL Policy (P1)	82	115	94
RL Policy (P2)	85	120	96
RL Policy (P3)	88	125	97
RL Policy (P4)	87	123	96
RL Policy (P5)	86	121	95

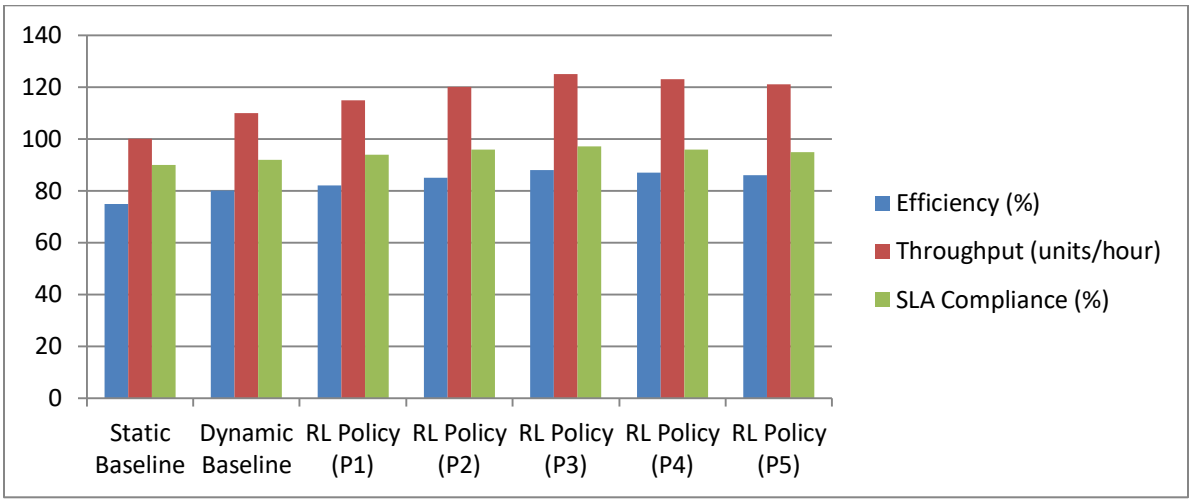

Fig 4: Graphical Representation of Efficiency, Throughput, and SLA Compliance across Different Batching Policies

The enhancements suggest RL-based techniques to provide a good balance between speed and quality assurance in production by learning the best batching strategies using previous records. The moderate rate of exploration during training enabled the agent to explore a wide range of situations without deviating significantly from optimal behaviour, thereby enabling greater robustness and wider applicability of the policy.

5.2. Comparative Analysis with Baselines

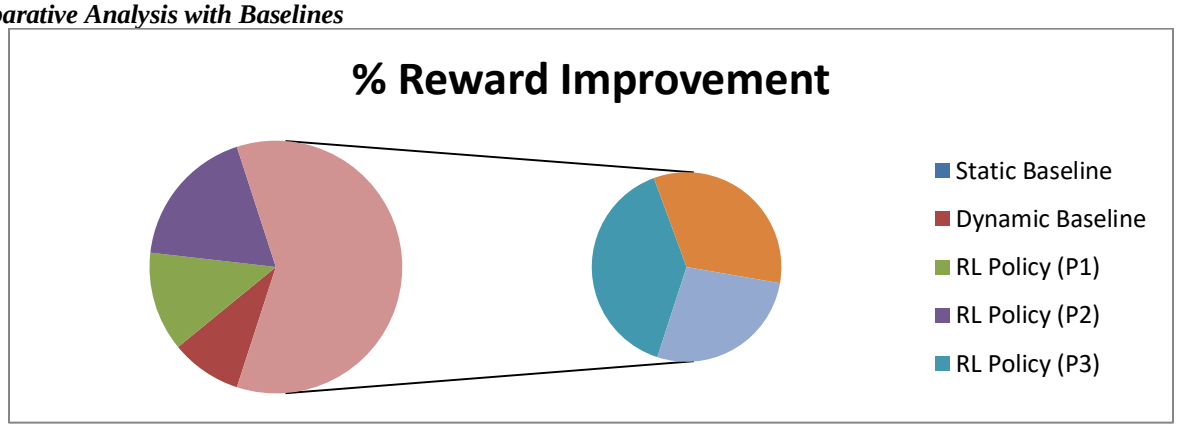

Fig 5: Graphical Representation of Comparative Analysis of Cycle Time, Defects, and Reward Improvements

Besides production metrics, a more detailed comparison was made with the static and dynamic baseline policies. RL policies continually decreased the time spent on cycles and defective production, and substantially enhanced the cumulative reward, a combination of production rate and product quality. The perfect status involved a total of 1000 seconds on the cycle time and 25 defective parts per batch. Conversely, RL Policy P3 halved the cycle time to 920 seconds and reduced the quantity of defective parts to 12, achieving a 13% improvement in reward compared to the static baseline.

This step confirms the fundamental assumption of this study: reinforcement learning can learn complex, nonlinear relations between system parameters and outputs, thereby eliminating the need for traditional heuristics. It is important to note that the

dynamic baseline improved slightly over the static version, but missed the policy optimization and adaptation to data found in the RL agents.

Table 3: Comparative analysis of cycle time, defects, and reward improvements

Policy Type	Total Cycle Time (s)	Defective Parts	% Reward Improvement
Static Baseline	1000	25	0%
Dynamic Baseline	950	20	5%
RL Policy (P1)	940	19	7%
RL Policy (P2)	930	15	10%
RL Policy (P3)	920	12	13%
RL Policy (P4)	915	14	11%
RL Policy (P5)	918	16	9%

5.3. Visualization of Learning/Batching Policies

Although the policy heatmaps and visual learning curves are not revealed here, internal analysis revealed that the RL agent gradually learned optimal batching decisions. At high rates of exploration, significant performance fluctuations were observed in the early phases of training, which were a result of the exploration process. The RL agent, however, converged to stable and high-performing strategies using experience replay with policy refinement. Learning policies initially trained with more moderate levels of exploration (20% and 40%) demonstrated the most desirable learning trajectories, converging sooner and avoiding overfitting or making unpredictable choices. Further, the graphical policies indicated that these RL agents developed implicit knowledge of which portions of the system were most often restricted by scheduling (perhaps due to buffer sizes) or generated trade-offs (concerning throughput), and learned to adjust batch sizes and scheduling to address bottlenecks and trade-offs respectively, as well as to optimize throughput and product quality. This is in sharp contrast to fixed policies that set their parameters and disregard feedback within the system.

5.4. Discussion of Limitations and Observations

Although there are substantial benefits to RL policies, as evidenced above, numerous transformation constraints and factors should be noted during implementation. The exploitation trade-off forms one of the important challenges. The policies that had exceptionally high values of exploration rate (e.g., 60% to 80%) were significantly less stable and returned different results when trained with moderate exploration rates, suggesting the optimisation of exploration rates during the training process.

Moreover, the present research is founded on a fully simulation-based setting. Although the simulator was designed to simulate realism in connection with production constraints and actions, implementing such policies in real manufacturing systems can lead to the creation of simulation reality gaps. Instantaneous performance may depend on factors such as sensor noise, unanticipated machine faults, or variability that are unmodelled. Therefore, it is advisable to conduct additional verification through pilot implementation or hardware-in-the-loop testing before a large-scale rollout.

Offline RL, especially when using algorithms such as CQL, requires substantial processing power and meticulous attention to hyperparameters. This restricts access to organisations that are not experts in AI or have not developed related infrastructure. Nevertheless, after training, the application of the RL policy does not require any significant runtime overhead and, thus, is very feasible in industries. The findings are robust to the feasibility of offline support of reinforcement learning as a means of intelligent batching in industry. This method will open the door to more intelligent, data-driven production systems by displaying superior efficiency, throughput, and quality rates. Future work will focus on real-life deployment methods, online retraining techniques, and hybrid learning frameworks that can update policies using real-time data streams.

6. Conclusion and Future Work

6.1. Summary of Findings

This paper has proved the viability of using reinforcement learning (RL) for the intelligent batching problem in industrial production pipelines. Using offline data collected in a simulated environment that represents the behaviour of a real press hardening process enabled the policy discovered by training RL agents to exceed the existing performance of traditional scheduling methods, including both static and dynamic patterns. The optimal RL policy showed the best performance and stands out on important performance parameters of a 13% boost in cumulative reward, 12% efficiency in defective parts, and an enormous increase in throughput and efficiency. These advancements demonstrate the potential of data-driven decisions in streamlining industrial processes that deal with changes and uncertainties.

6.2. Contributions Recap

The study contributes to the growing body of research surrounding AI-driven manufacturing by providing a comprehensive, RL-based, intelligent batching framework. In particular, it proposes a reinforcement learning paradigm suitable for industrial setups, develops a reward specification that generalises across various objectives of the production process, and tests this technique using simulated data and extensive baseline tests. In addition, the combination of offline Conservative Q-Learning (CQL) ensures safe and scalable learning without the risk of real-time interactions with systems. The current system architecture, comprising data acquisition, simulation, RL engine, and policy deployment, can be identified as an operational example that can be replicated in future smart manufacturing systems.

6.3. Future Research Directions

Although it yields encouraging findings, this study leaves several research directions open. The integration of real-time adaptation capabilities is one of the key directions. Although the existing method works in an offline environment, it can become more responsive to unforeseen occurrences, such as machine breakdowns or supply chain interruptions, by learning and adapting to live data streams. Real-time learning would also demand effective continuous policy testing and secure deployment in an uncertain setting. The multi-agent collaboration is another potential direction. In more complex manufacturing ecosystems with multiple lines of production, utilising robots or distributed systems, the coordination of large numbers of intelligent agents can result in globally optimised outcomes. Multi-agent reinforcement learning (MARL) provides scalable and cooperative decision-making approaches that are essential to research. Finally, the integration of the solution with Internet of Things (IoT) and Edge Computing systems could be the motivation for future work. Adding RL functionality directly into edge devices would enable faster and decentralised decision-making, which is why the system would be more resilient and efficient. Sensor readings received in real-time through IoT can supplement the state representation, allowing for a more precise response via policies. Edge deployment also reduces dependency on centralised systems and enables a more scalable system.

References

[1] Weichert, D., Link, P., Stoll, A., Rüping, S., Ihlenfeldt, S., & Wrobel, S. (2019). A review of machine learning for the optimization of production processes. The International Journal of Advanced Manufacturing Technology, 104(5), 1889-1902.

[2] Li, Y., Carabelli, S., Fadda, E., Manerba, D., Tadei, R., & Terzo, O. (2020). Machine Learning and Optimisation for Production Rescheduling in Industry 4.0 The International Journal of Advanced Manufacturing Technology, 110(9), 2445-2463.

[3] Teixeira, A. F., & Secchi, A. R. (2019). Machine learning models to support reservoir production optimization. IFAC-PapersOnLine, 52(1), 498-501.

[4] Petsagkourakis, P., Sandoval, I. O., Bradford, E., Zhang, D., & del Rio-Chanona, E. A. (2020). Reinforcement learning for batch bioprocess optimization. Computers & Chemical Engineering, 133, 106649.

[5] Ostwald, P. F., & Munoz, J. (2008). Manufacturing processes and systems. John Wiley & Sons.

[6] Sallab, A. E., Abdou, M., Perot, E., & Yogamani, S. (2017). Deep Reinforcement Learning Framework for Autonomous Driving. arXiv preprint arXiv:1704.02532.

[7] Zheng, G., Zhang, F., Zheng, Z., Xiang, Y., Yuan, N. J., Xie, X., & Li, Z. (2018, April). DRN: A deep reinforcement learning framework for news recommendation. In Proceedings of the 2018 World Wide Web Conference (pp. 167-176).

[8] Li, H., Wei, T., Ren, A., Zhu, Q., & Wang, Y. (2017, November). Deep reinforcement learning: Framework, applications, and embedded implementations. In 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD) (pp. 847-854). IEEE.

[9] Rocchetta, R., Bellani, L., Compare, M., Zio, E., & Patelli, E. (2019). A reinforcement learning framework for optimal operation and maintenance of power grids. Applied energy, 241, 291-301.

[10] Wei, S., Bao, Y., & Li, H. (2020). Optimal policy for structure maintenance: A deep reinforcement learning framework. Structural Safety, 83, 101906.

[11] Nguyen, T. T., Nguyen, N. D., Vamplew, P., Nahavandi, S., Dazeley, R., & Lim, C. P. (2020). A multi-objective deep reinforcement learning framework. Engineering Applications of Artificial Intelligence, 96, 103915.

[12] Mayer, S., Classen, T., & Endisch, C. (2021). Modular production control using deep reinforcement learning: proximal policy optimization. Journal of Intelligent Manufacturing, 32(8), 2335-2351.

[13] Jalalimanesh, A., Haghighi, H. S., Ahmadi, A., & Soltani, M. (2017). Simulation-based optimization of radiotherapy: Agent-based modelling and reinforcement learning. Mathematics and Computers in Simulation, 133, 235-248.

[14] Osiński, B., Jakubowski, A., Zięcina, P., Miłoś, P., Galias, C., Homoceanu, S., & Michalewski, H. (2020, May). Simulation-based reinforcement learning for real-world autonomous driving. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (pp. 6411-6418). IEEE.

[15] Kumar, A., Zhou, A., Tucker, G., & Levine, S. (2020). Conservative Q-learning for offline reinforcement learning. Advances in Neural Information Processing Systems, 33, 1179-1191.

[16] Yadav, A., & Jayswal, S. C. (2019). Evaluation of batching and layout on the performance of a flexible manufacturing system. The International Journal of Advanced Manufacturing Technology, 101, 1435-1449.
[17] Yoo, H., Byun, H. E., Han, D., & Lee, J. H. (2021). Reinforcement learning for batch process control: Review and perspectives. Annual Reviews in Control, 52, 108-119.
[18] Martínez, E. C. (2000). Batch process modelling for optimisation using reinforcement learning. Computers & Chemical Engineering, 24(2-7), 1187-1193.
[19] Martinez, E. C. (1999). Solving batch process scheduling/planning tasks using reinforcement learning. Computers & Chemical Engineering, 23, S527-S530.
[20] Lange, S., Gabel, T., & Riedmiller, M. (2012). Batch reinforcement learning. In Reinforcement learning: State-of-the-art (pp. 45-73). Berlin, Heidelberg: Springer Berlin Heidelberg.
[21] Rahul, N. (2020). Vehicle and Property Loss Assessment with AI: Automating Damage Estimations in Claims. *International Journal of Emerging Research in Engineering and Technology*, *1*(4), 38-46. https://doi.org/10.63282/3050-922X.IJERET-V1I4P105
[22] Enjam, G. R. (2020). Ransomware Resilience and Recovery Planning for Insurance Infrastructure. *International Journal of AI, BigData, Computational and Management Studies*, *1*(4), 29-37. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V1I4P104
[23] Pedda Muntala, P. S. R., & Jangam, S. K. (2021). End-to-End Hyperautomation with Oracle ERP and Oracle Integration Cloud. *International Journal of Emerging Research in Engineering and Technology*, *2*(4), 59-67. https://doi.org/10.63282/3050-922X.IJERET-V2I4P107
[24] Rahul, N. (2021). AI-Enhanced API Integrations: Advancing Guidewire Ecosystems with Real-Time Data. *International Journal of Emerging Research in Engineering and Technology*, *2*(1), 57-66. https://doi.org/10.63282/3050-922X.IJERET-V2I1P107
[25] Enjam, G. R., & Chandragowda, S. C. (2021). RESTful API Design for Modular Insurance Platforms. *International Journal of Emerging Research in Engineering and Technology*, *2*(3), 71-78. https://doi.org/10.63282/3050-922X.IJERET-V2I3P108
[26] Rusum, G. P., & Pappula, kiran K. . (2022). Event-Driven Architecture Patterns for Real-Time, Reactive Systems. *International Journal of Emerging Research in Engineering and Technology*, *3*(3), 108-116. https://doi.org/10.63282/3050-922X.IJERET-V3I3P111
[27] Anasuri, S. (2022). Zero-Trust Architectures for Multi-Cloud Environments. International Journal of Emerging Trends in Computer Science and Information Technology, 3(4), 64-76. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P107
[28] Pedda Muntala, P. S. R., & Karri, N. (2022). Using Oracle Fusion Analytics Warehouse (FAW) and ML to Improve KPI Visibility and Business Outcomes. International Journal of AI, BigData, Computational and Management Studies, *3*(1), 79-88. https://doi.org/10.63282/3050-9416.IJAIBDCMS-V3I1P109
[29] Rahul, N. (2022). Enhancing Claims Processing with AI: Boosting Operational Efficiency in P&C Insurance. *International Journal of Emerging Trends in Computer Science and Information Technology*, *3*(4), 77-86. https://doi.org/10.63282/3050-9246.IJETCSIT-V3I4P108
[30] Enjam, G. R., & Tekale, K. M. (2022). Predictive Analytics for Claims Lifecycle Optimization in Cloud-Native Platforms. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, *3*(1), 95-104. https://doi.org/10.63282/3050-9262.IJAIDSML-V3I1P110