systems

MDPI

Article

A Learning-Based Decision Tool Towards Smart Energy Optimization in the Manufacturing Process

Choumicha El Mazgualdi [1,*], Tawfik Masrour [1], Noureddine Barka [2] and Ibtissam El Hassani [1]

[1] Laboratory of Mathematical Modeling, Simulation and Smart Systems (L2M3S), Artificial Intelligence for Engineering Sciences Team (IASI), Department of Mathematics and Computer Science, ENSAM-Meknes, Moulay Ismail University, Meknes 50050, Morocco
[2] Mathematics, Computer Science and Engineering Department, Université du Québec à Rimouski, 300 Allée des Ursulines, Rimouski, QC G5L 3A1, Canada
* Correspondence: c.elmazgualdi@edu.umi.ac.ma

Abstract: We developed a self-optimizing decision system that dynamically minimizes the overall energy consumption of an industrial process. Our model is based on a deep reinforcement learning (DRL) framework, adopting three reinforcement learning methods, namely: deep Q-network (DQN), proximal policy optimization (PPO), and advantage actor–critic (A2C) algorithms, combined with a self-predicting random forest model. This smart decision system is a physics-informed DRL that sets the key industrial input parameters to optimize energy consumption while ensuring the product quality based on desired output parameters. The system is self-improving and can increase its performances without further human assistance. We applied the approach to the process of heating tempered glass. Indeed, the identification and control of tempered glass parameters is a challenging task requiring expertise. In addition, optimizing energy consumption while dealing with this issue is of great value-added. The evaluation of the decision system under the three configurations has been performed and consequently, outcomes and conclusions have been explained in this paper. Our intelligent decision system provides an optimized set of parameters for the heating process within the acceptance limits while minimizing overall energy consumption. This work provides the necessary foundations to address energy optimization issues related to process parameterization from theory to practice and providing real industrial application; further research opens a new horizon towards intelligent and sustainable manufacturing.

Keywords: deep reinforcement learning; process parameters self-configuration; energy self-optimization; autonomous process control; intelligent manufacturing; glass tempering process; dual-optimization problem; Industry 4.0

Citation: El Mazgualdi, C.; Masrour, T.; Barka, N.; El Hassani, I. A Learning-Based Decision Tool Towards Smart Energy Optimization in the Manufacturing Process. *Systems* **2022**, *10*, 180. https://doi.org/10.3390/systems10050180

Academic Editor: Vladimír Bureš

Received: 13 September 2022
Accepted: 3 October 2022
Published: 7 October 2022

1. Introduction

1.1. Background and Motivation

Energy consumption is a major sustainability focus in industrial manufacturing sector [1]. Furthermore, due to the rapid technological development and increasing industrialization, energy-efficient handling is required for competitiveness purposes. Thus, large-scale industrial installations are extremely affected, which is the case in the processing industry, particularly in automobile construction. Electricity accounts for a significant share of energy resources in modern industrial processes among the various energy sources. It represents more than 50% of the total energy consumption of this sector [2,3].

As a pillar of the automotive industry, automotive glass manufacturing faces challenges in optimizing its electrical energy consumption. Extravagantly, the tempering process of automotive glass, as one of the largest consumers of electrical energy, is concerned in the present work. As a result, stakeholders are increasingly interested in reducing energy consumption at the process level, which can lead to being competitive, becoming sustainable and significantly reducing costs.

By the emergence of Industry 4.0, a new trend towards smart manufacturing systems and autonomous process control is revolutionizing the manufacturing industry [4,5]. Among the technological portfolio enabling this leading revolution, the internet of things (IoT), cyber-physical systems (CPS), and artificial intelligence (AI) are playing a pivotal role, transforming ordinary industrial processes into smart, connected, and autonomous processes. From a mathematical perspective, physics-based processes can be accurately represented by mathematical equations, making them easily modelable and learned. However, from an industrial point of view, representing a process is a more complicated task requiring, in addition to the mathematical representation, a high level of process knowledge and environmental conditions awareness [6,7]. Reinforcement learning (RL) is a branch of AI originally inspired by research into animal learning. Roughly speaking, an RL algorithm is able to incorporate learning capabilities into an ordinary process controlled by humans, thus allowing it to become self-configurable without the need for human intervention [8–11].

In summary, the present work aims at developing a decision system for solving a dual-optimization problem in the glass tempering process. The optimization problem consists of optimizing both the final product quality and the overall energy consumption during a heating cycle. For this purpose, we developed a decision model using a self-prediction Radom Forest (RF) model combined with three RL agents namely, deep Q-network (DQN), proximal policy optimization (PPO), and advantage actor–critic (A2C) algorithms. The agents are led to learn the optimal strategy to identify the suitable parameter values without prior knowledge of the operating environment. The simulated environment was realized using a self-predicting machine learning (ML) model trained on background data obtained from ANSYS software. The trained model will be able to identify the appropriate recipe of parameters to meet production requirements and the energy effectiveness. Finally, the feasibility and convergence of the proposed methods are confirmed by practical experiments in an automotive glass company.

1.2. Tempering Process of Automotive Glass Manufacturing

Glass tempering is a complex manufacturing process that consists of heat treatment to obtain a type of safety glass in high demand within the automotive industry. The process involves extreme heating of the glass to a specific temperature and then rapid cooling which hardens the glass both thermally and physically [12,13].

Before the tempering process, the glass is cut to the desired size to avoid any decrease in strength after the treatment. The cut glass is then examined for imperfections such as cracks, bubbles, and inclusion to prevent breakage during the tempering process, and it is washed to remove any debris that may interfere with the thermal tempering. Once cut, examined and washed, the glass begins a heat treatment process in which it travels, either in batches or continuously feed, through a tempering furnace which exposes the glass to a temperature of over 600 degrees Celsius. After that, a high-pressure cooling process is applied to the heated glass. During this process, the glass surface receives jets of high pressure air from a set of nozzles in varying positions. The cooling process; also called "quenching", makes the outer surfaces of the glass cool much faster than the center of the glass and contracts [14]. As a result, quenching puts the outer surfaces into compression, while the interior of the glass remains in tension. The existence of compressive stress in the surface and tensile stress in the center make the tempered glass stronger than ordinary glass. Thus, tempered glass can handle more stress than ordinary glass due to its ability to balance the force applied to it. In addition, it causes the glass, when broken, to fall apart into small fragments instead of splintering into sharp shards as plate glass does. The smallish cube-like chunks are considerably less able to cause serious injuries, making the tempered glass very used in those environments where human safety is an issue [14,15]. Tempered glass is used in the cars surrounding windows and the back window, known in the automotive glass industry as sidelites and backlites.

The temperature inside the tempering furnace and the glass traveling speed are the two main control parameters of the tempering glass process, especially in automotive glass manufacturing. Those parameters are defined as the process parameters that have a direct impact on the final quality of the product. At the launch of a new project, those two parameters are defined based on the accumulated experience of process engineers, but often the operators need to adjust it, which is time and resources consuming. Thus, defining those process parameters for any new project, bypassing destructive tests, will be of great interest.

In our previous work [16], we investigated the applicability of the DQN algorithm to identify and master the tempering process parameters in industrial electric furnaces, reaching the self-configuration level of our process. We proved its convergence to an optimal policy, and we exposed its interesting and relevant results. The developed DQN model was able to identify the good process parameters with deviation not exceeding process limits [16]. In the present work, we will focus more on the overall energy optimization problem with the purpose of achieving the self-optimization level of the studied process. Our main contribution concerns the resolution of a double optimization problem using DRL techniques to deal with one of the critical industrial problems, namely, having the right quality while optimizing energy consumption. Furthermore, in the present context of digitalization, energy consumption is one of the sustainability concerns that must be carried out carefully. The paper is tackling this issue from an industrial point of view, further contributions could be made for more innovative achievements towards a sustainable environment.

The paper is structured as follows: Section 2 outlines a number of published papers handling the problem of energy optimization in the manufacturing industry. Section 3 provides some background on reinforcement learning and its related algorithms used in the present work. Section 4 is about the problem statement providing a concise description of the heating process control formulation. In Section 5, we expose, in a detailed way, the result of the experiments. Finally, Section 6 offers some final remarks and suggestions for further work.

2. Literature Review

Recently, there have been numerous studies carried out to optimize energy consumption in various manufacturing sectors. Moreira et al. [17] used a combination of experimentation and Taguchi analysis to investigate the impact of the key machining parameters of a CNC machining process on the energy consumption. They developed a novel improved multi-swarm Fruit Fly optimization algorithm (iMFOA) for solving the multi-objective optimization model. The effectiveness of optimization approaches has been proven in fine-tuning key parameters to improve energy efficiency while meeting other technical production requirements. In [18], the fast and elitist non-dominated sorting genetic algorithm (NSGAII) technique was applied to provide a set of Pareto multiple optimum solutions of steam cycle power plant. The authors first used data from the operating power plant to perform thermo-economic modeling of the steam cycle power plant. Then, the turbine inlet temperature, the turbine extraction pressures, the boiler pressure, the condenser pressure, the isentropic efficiency of the turbines and pumps as well as the reheat pressure were used as control variables, while yield and total cost ratio were considered as two objective functions for the optimization problem. Compared to the actual data of the power plant in operation, the optimization results at some points simultaneously show a 3.76% increase in efficiency and a 3.84% decrease in the total cost rate. Finally, a correlation between the two objective functions and the decision variables was provided with acceptable accuracy using artificial neural network. Seo et al. [19] developed a physics-based model capable of mimicking the dynamic behavior of a prototype electrically amplified glassware furnace. They presented a dynamic optimization strategy that offers an optimal balance between the use of electricity and natural gas given fluctuations in the price of electricity. Optimal demand response (DR) decisions for two case studies of practical interest were analyzed.

Geng et al. [20] proposed an energy optimization and prediction model based on the improved convolutional neural network (CNN) integrating the cross-feature (CFeCNN) in order to improve the energy efficiency in the petrochemical industry. The developed method was applied to establish energy optimization and prediction model of ethylene production systems in the petrochemical industry. The capability of the proposed method is proved by achieving an energy utilization efficiency increase of 6.38%, which leads to a reduction of carbon emissions by 5.29%. Su et al. [21] presented a multi-objective optimization method of cutting parameters that take into account the response surface methodology (RSM) and the grey relational analysis. The proposed method was applied to turn AISI 304 austenitic stainless steel for reducing energy consumption without compromising the cutting quality and the production rate. The turning experiments were established using the Taguchi method, and then, using grey relational analysis, the optimization problem was transformed from multi-objective into simple objective optimization problem. Finally, the regression model based on RSM for the grey relational grade was developed to determine the optimal combination of the turning parameters. Based on this optimal combination, the followed optimizations were performed; surface roughness (Ra) decreases by 66.90%, material removal rate (MRR) increases by 8.82%, and finally a decrease in the specific energy consumption (SEC) by 81.46%. Sangwan et al. [22] proposed a multi-objective optimization model that minimizes the power consumption while maximizing the material removal rate for different targeted values of surface roughness during machining. The results of the proposed predictive model reveal a significant decrease of the specific cutting energy consumption in the present study. Furthermore, an analysis of variance (ANOVA) was performed in order to confirm the model's fitness and adequacy. Wang et al. [23] established a dual-objective optimization model to identify the milling parameters that minimize the power consumption and process time. Using the sixteen groups of experimental data, the power consumption model is obtained through nonlinear regression. Then, an improved artificial bee colony (ABC) intelligent algorithm is used to resolve the proposed optimization model with multiple constraints of milling processing conditions. Compared with the non-dominated sorting genetic algorithm (NSGA-II), the proposed methodology has good performance. Luan et al. [24] developed a multi-objective optimization model performing a trade-off between machining quality, cutting tool cost, and electricity cost for enhancing the environmental and economic impacts of the manufacturing process. The proposed model was designed to identify the optimal cutting conditions that minimize electrical energy and cutting tool costs without compromising manufacturing quality. An online tool wear prediction model was developed to map the relations between tool wear and cutting power based on an experimental study. Next, the multi-objective optimization model is used to solve the trade-off analysis problem by analyzing flank tool wear data, surface roughness, and the real-time cutting power data. Finally, the grey relational grade of the cutting parameters, on the three optimization objectives, was obtained using the grey relational analysis. Visualizations of the trade-offs among the parameters were performed using the tool wear-cutting power-surface roughness trade-off analysis. Han et al. [25] proposed a novel energy management and optimization model of complex petrochemical industries that uses the fuzzy extreme learning machine (FELM) method integrated with the fuzzy set theory. The triangular fuzzification is used to solve the problem of the fluctuation and uncertainty data. In addition, the training of the FELM is performed using the cross recombination of triangular fuzzy numbers (TFNs). Compared with the fuzzy error back-propagation network (FBP) and the fuzzy radial basis function network (FRBF), the FELM has the best predictive performance and the fastest convergence speed on the University of California Irvine (UCI) standard data sets. The proposed method was applied to manage and optimize the energy status of China's ethylene industry, proving its effectiveness and applicability in the energy-saving potential, which is calculated up to about 15%. Golkarnarenji et al. [26] developed an intelligent predictive model based on support vector regression (SVR) algorithm for energy optimization in the thermal stabilization process. The developed predictive model combined with genetic algorithm (GA)

optimizer returned a very satisfactory configuration, achieving energy savings of up to 43%, under both physical property and skin-core defect constraints. Despite the limited training data set, the developed stochastic-SVR-GA approach offers potential energy savings for similar chemical industries, including carbon fiber manufacturing. Finally, in [27], the authors developed a real-time monitoring system of manufacturing workflow for the Smart Connected Worker. Among others, an artificial neural network are introduced to enable real-time energy dis-aggregation to improve energy efficiency.

The literature studies discussed above reveal a wide range of industrial problems that has been converted into optimization problems, from single-objectives to multi-objective, and then solved using one or more optimization techniques. However, to our best knowledge, not much focus has been paid to modeling a decision system capable of providing two-level automation, namely, self-configure and self-optimize towards an intelligent manufacturing process.

3. Background on Reinforcement Learning

Inspired by behaviorist psychology, reinforcement learning (RL) is an aspect of machine learning where an agent learns a strategy to behave in an environment using a trial-and-error paradigm [28,29]. At each iteration, the agent interacts with the environment and changes its actions under a policy π. On the other hand, the environment changes state and provides a reward in response to the action taken. The agent's goal is to maximize the reward received from the environment via a learned value function [30].

By performing an action and receiving a response, the agent creates its own experience via a trade-off between exploration and exploitation of its environment [31]. As such, this paradigm is not limited to specific conditions as it considers the whole problem of a goal-directed agent that interacts with an uncertain environment without explicitly dividing and isolating sub-problems [32]. In this regard, reinforcement learning is always well adapted to changing environments and variable conditions, moreover, it easily adapts to different control and multi-optimization tasks.

The Markov decision process (MDP) is an idealized form for reinforcement learning problems providing precise theoretical statements. The tuple <S, A, P, R> of the Markov decision process framework describes in a perfect way any reinforcement learning problem. S denoted the state-space, A denoted the action-space, P is the transition probability, and R represents the reward function. Another parameter can be added to this, which is the discount factor γ [33]. The discounted accumulated future reward from time t, also called the expected return is then given by:

$$R_t = r_t + \gamma r_{t+1} + ... + \gamma^{T-t} r_T = \sum_{k=0}^{T-t} \gamma^k r_{t+k} \quad t \in \mathbb{N} \tag{1}$$

where r_t is the reward received from the transition at time t, and T is the time when the episode terminates.

This accumulated reward holds the information about which actions were taken in each state, in other words, it has the information about the agent strategy or policy π.

Then, the state-value function is defined as the expected return from the current state, s under the policy π and is then given by:

$$v_\pi(s) = E[R_t | s_t = s] \tag{2}$$

Since the state-value function only provides the estimation of how good it is for the agent to be in a given state, there is another function of a state value that provides a more complete estimate, including the action to perform in a given state. This function is called the action-value function and is defined as the expected return from state s, taking the action a, while following policy π. The action-value function can be expressed as follows:

$$q_\pi(s, a) = E[R_t | s_t = s,\ a_t = a] \tag{3}$$

Dynamic programming (DP) is one traditional method for solving MDP problems, it refers to a collection of algorithms namely, value iteration, policy iteration, and linear programming [34]. Such methods rely on a clear and perfect understanding of the environment in order to be able to model it perfectly, which involves enormous computation time. However, if the model of the environment is not available or the process is not finite, DP becomes more difficult to apply. In this case, model-free algorithms are such a good alternative that uses experience to approximate the value function, and then access the information to make decisions [11]. In many interesting applications, especially industrial case studies where assuming a perfect model of the environment is a difficult task, model-free algorithms are more suitable, providing a DP-like result with less computation.

Q learning and policy gradients are considered to be the two most popular classes of reinforcement learning. In policy gradient methods, the policy is directly parameterized as a distribution over actions, while Q learning is a type of value iteration method that approximates the Q function.

3.1. Q-Learning

Proposed originally by Watkins in 1989 [35], Q-learning is a model-free, off-policy RL algorithm that methodically updates the action value function q(s, a) without using the dynamics of the environment. Learning the long-term expected utility Q(s, a), also called the Q-value function, for each state-action pair is the basic idea behind the Q-learning algorithm. Thus, the expected utility Q(s, a) directly approximates the optimal action-value function q_*.

The standard formulation of Q-learning is stated in the pseudo-code in Figure 1.

Set the step size $\alpha \in (0, 1]$

Initialize $Q(s, a)$ arbitrarily except that $Q(terminal, .) = 0$

For each *episode* **do**

Initialize s

For each *step of episode* **do**

Choose action, a given state, s using a certain policy (e.g. ε-*greedy*)

Execute action, a, observe immediate reward R,

and perceive the transition state, s'

$Q(s, a) \leftarrow Q(s, a) + \alpha[R + \gamma Q(s', a) - Q(s, a)]$

$s \leftarrow s'$

End

Until s is terminal

Figure 1. Q-learning algorithm.

The parameter α is the learning rate or the step size, it denotes the update brought to the utility value each time an action is taken, and it is the application's specific parameter. An episode represents the learning cycle of Q-learning. The initialization of utility values Q(s, a) depends on the application, if there is no preference for certain actions in any specific state, the Q(s, a) values are the same for all state-action pairs. The Q-learning is considered as an off-policy learning algorithm, which means that the update is independent of any policy. However, a trade-off between exploration and exploitation is ensured by the ε-greedy policy.

In a Q-learning algorithm, a lookup table is used to represent the q-values since every state and action need to be visited multiple times to get a good approximation of

the q-value [36]. This makes the use of Q-learning reduced to cases where action-state spaces are finite and discrete. However, for advanced real-world applications, the state space is very large or even continuous which requires representing the q-function with a function approximation.

Function approximation is a kind of generalization that attempts to find a parameterized function Q(s, a; θ) satisfying Q(s, a) $\approx$ Q(s, a; θ). Therefore, instead of looking up a state-action pair and find its corresponding q-value, we make use of a predictive method to find the q-values even for unseen states. As one of the most powerful function approximation, ANN is very used especially, for nonlinear functions.

An ANN is a network of interconnected units that process information in a similar way biological neural networks do [37]. The neurons constitute the building blocks of an ANN, and they are connected to each other by synapses which are just weighted values. The network is composed of layers, where information is transmitted from the input layer, which is the data we provide to the ANN, through the hidden layers, to the output layer in which the finished calculations of the network are placed. With an appropriate choice of the architecture, ANNs can be used to approximate very complex functions [38].

- Deep Q-Learning

Combining the perception of deep learning (DL) with the decision-making ability of the reinforcement learning, deep reinforcement learning (DRL) is a robust and scalable approach widely used when dealing with complex real-world problems. The approach consists of using ANNs as approximators of the action-value function q(s, a) by providing end-to-end learning of appropriate representations and features for the relevant problem. When DL is applied to Q-learning, it is called a deep Q-network (DQN) [39]. The use of DRL instead of RL allows the good capability to capture complex, non-linear patterns which makes it able to solve classes of problems previously deemed unfit for RL. In DQN, the state of the RL algorithm is used as the input layer of the DNN, while the predicted q-values of the state-action pairs are used as the output layer (Figure 2).

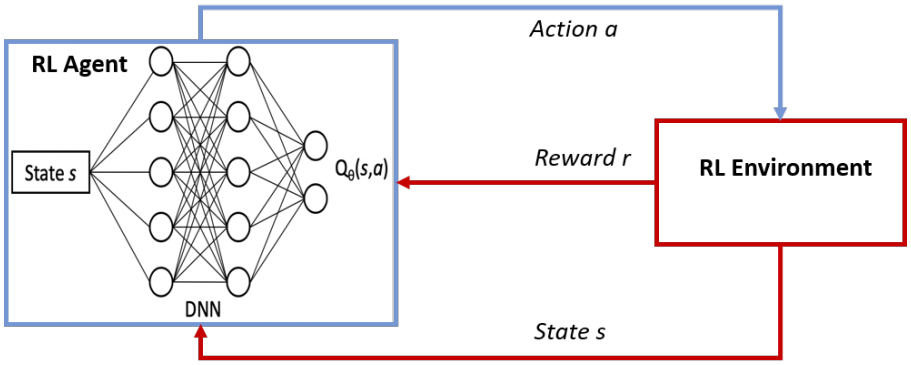

Figure 2. Schematic structure of Deep Q-Network (DQN) agent.

The parameters of the DNN are updated repeatedly, by minimizing the loss function (Equation (4)), until the difference between the target value and the predicted value converges.

$$L_i(\theta_i) = E\left[(r + \gamma max Q(s',\ a',\ \theta_{i-1}) - Q(s,\ a,\ \theta_i))^2\right] \quad (4)$$

3.2. Policy Gradients

Unlike the Q learning that approximates the Q function and uses it to infer the optimal policy $\pi*$ (i.e., $argmax_a Q(s,\ a)$), policy gradients (PG) aims to model and optimize the policy directly in the policy space [40]. A PG algorithm usually seeks to directly model the action probabilities using a neural network (or other function approximators). It consists of collecting a small batch of experiences, and then using this batch to do a gradient update of the policy. Thus, the experiences used for the update are immediately discarded away, and the newly updated policy is used to collect a new batch of experiences. This also means

that policy gradient methods are typically less sample efficient than Q-learning methods because they only use the collected experiences once for doing an update. The loss function depends on the policy $\pi_\theta(a|s)$ which is a parameterized function respect to θ, and it is defined as the expectation of the log of the policy actions multiplied by an estimate of the advantage function:

$$L^{PG}(\theta) = \widehat{E}_t\left[\log \pi_\theta(a_t|s_t)\widehat{A}_t\right] \tag{5}$$

where $\widehat{A}_t$ is the advantage function, it estimates the relative value of the selected action in the current state. Generally, it is equal to the discounted rewards minus a baseline estimate. By properly determining the baseline, we ensure that the advantage function is only positive if the action is good and negative if the action is bad. In practice, the baseline function is often replaced by the value function V(s), which is simply the expected value of total future reward at state s. So $\widehat{A}_t$ is really telling us how much better the chosen action was based on the expectation of what would normally happen regarding the state in which the agent was.

Policy gradient generally relies on a stochastic gradient ascent algorithm to minimize the loss (objective) function [40] described above (Equation (5)). The advantage-actor–critic (A2C) gradient estimator is commonly used and has the form:

$$\nabla_\theta L^{PG}(\theta) = \widehat{E}_t\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\widehat{A}_t\right] \tag{6}$$

Each time the agent interacts with the environment, the parameters of the neural network are tweaked so that "good" actions are more likely to be sampled in the future. Hence, the process is repeated until convergence to the optimal policy. However, traditional PG methods are sensitive to the choice of step size which can cause unbounded updates and make them suffer from high variance and low convergence. To overcome these drawbacks, trust region policy optimization (TRPO) was developed as an extension of the policy gradient method [41]. TRPO consist of constraining the step size of the policy update during training using the Kullback–Leibler (KL) divergence to limit the difference between the policy before and after the update. The objective function of TRPO is given by:

$$\underset{\theta}{max}\widehat{E}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\widehat{A}_t\right] \tag{7}$$

$$Subject\ to \qquad \widehat{E}_t\left[KL\left[\pi_{\theta_{old}}(\cdot|s_t),\ \pi_\theta(\cdot|s_t)\right]\right] \leq \delta \tag{8}$$

where $\pi_{\theta_{old}}$ is the old policy before updating, and π_θ is a new policy which will be changed in every iteration of updating. Although its reliability and data efficiency, TRPO is a complicated algorithm that cannot handle noise or the sharing of parameters between neural networks.

- Proximal Policy Optimization (PPO)

Proximal policy optimization (PPO) was introduced in 2017, by the OpenAI team, as an enhancement of the TRPO, and has easily established itself as one of the most popular RL methods preempting the deep-Q learning technique [42,43].

Retaining the advantages of TRPO, PPO aims to compute a gradient update at each step that ensures a minimized deviation from the previous policy while reducing the cost function. This allows PPO to strike a balance between ease of implementation, ease of tuning, and sample complexity. This simplified optimization process of PPO is expressed via a clipped surrogate objective function as the policy network loss function that is formulated as follows [42]:

$$L^{CLIP}(\theta) = \widehat{E}_t\left[min\left(r_t(\theta)\widehat{A}_t,\ clip(r_t(\theta),\ 1-\varepsilon,\ 1+\varepsilon)\widehat{A}_t\right)\right] \tag{9}$$

where the probability ratio $r_t(\theta)$ shown in Equation (9) is defined as:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{10}$$

The probability ratio $r_t(\theta)$ is between 0 and 1 when the action is less probable for the current policy than for the policy before updating, and it is greater than 1 when the action is more probable for the current policy than for the policy before updating.

By clipping the objective function, the ratio is limited and thus so are the policy updates. These methods have been shown to offer improved performance when compared to the KL divergence of TRPO while constraining the update step in a much simpler manner. PPO's pseudo code is shown in Figure 3.

For iteration = 1, 2, ...do
  For actor = 1, 2, ..., N do
    Run policy $\pi_{\theta_{old}}$ in environment for T time steps
    Compute advantage estimates $\hat{A}_1, \hat{A}_2, ..., \hat{A}_T$
  End for
  Optimize surrogate L wrt θ with k epochs and minibatch size M $\leq$ NT
  $\theta_{old} \leftarrow \theta$
End for

Figure 3. PPOProximal Policy Optimization (PPO), actor–critic style algorithm.

3.3. Actor–Critic Methods

Actor–critic algorithms are a combination of policy-based and value-based methods by providing two networks: actor network and critic network. This combination allows actor–critic methods to have faster convergence compared to actor only or critic only techniques [44]. Taking the state as input, the actor controls the agent behavior by selecting the best action to learn the optimal policy. On the other hand, receiving the environment and the action made by the actor, the critic evaluates the selected action and provides adjustments by computing the value function. In principle, this evaluation is carried out by calculating the temporal difference error (TD Error) of the action just selected. Thus, if the TD error is positive, the critic suggests strengthening the tendency of selecting this action in the future. Otherwise, it suggests weakening its tendency to be selected. Figure 4, illustrates the architecture of actor–critic with temporal difference error (TD error).

The actor–critic has two main variants: the advantage actor–critic (A2C) and the asynchronous actor–critic (A3C).

- Asynchronous Advantage Actor–Critic (A3C)

The asynchronous advantage actor critic (A3C) algorithm is one of the most popular and recent reinforcement learning algorithms that was developed Google's DeepMind in 2016 [45].

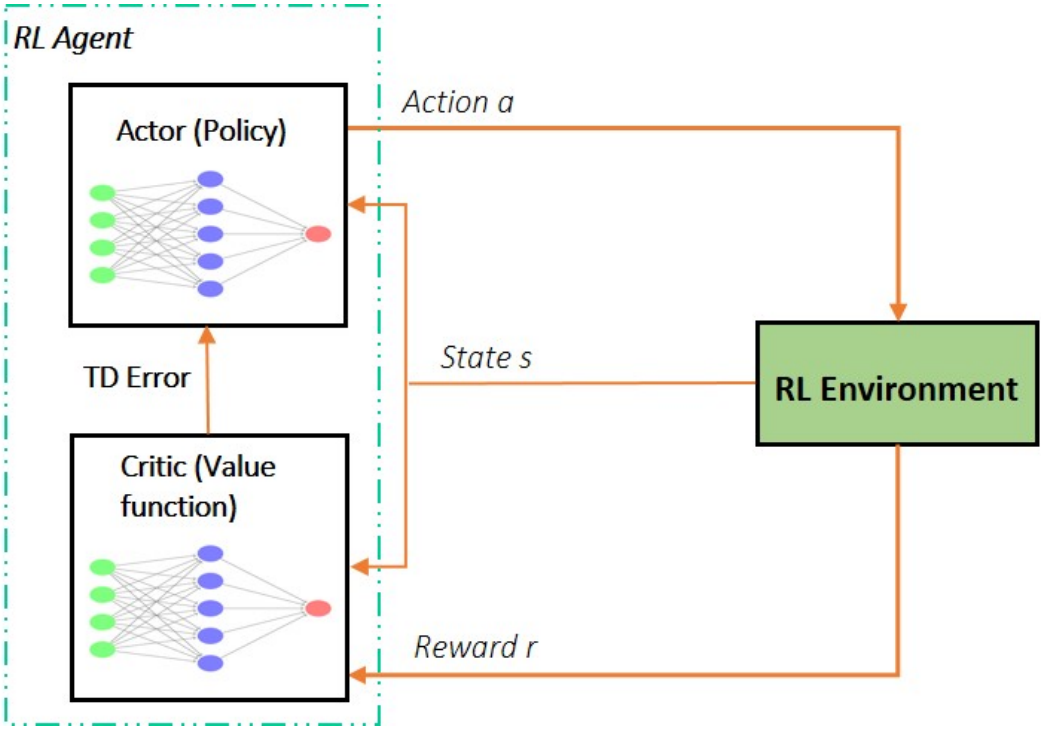

Figure 4. Actor–Critic Architecture.

The main idea behind A3C method is that it implements parallel training where multiple independent agents are exploring the environment. Each agent uses a different exploration policy to interact with its own copy of the environment asynchronously. The overall experience collected by the agents is aggregated and then used to update periodically the global network. As the updates are happening asynchronously, the agents need to reset their parameters, after each update, to the global network. The agents keep exploring their own environment, independently, until the next update as illustrated in Figure 5.

The asynchronous part of A3C, makes the learning process faster and more robust allowing efficient and effective exploration of the state space.

- Advantage Actor–Critic (A2C)

The main drawback of asynchrony is that some agents would be playing with an outdated version of the parameters and thus the global update would not be optimal. A2C is the synchronous deterministic implementation of the A3C method. This version of the actor–critic methods uses the same approach as A3C but in a synchronous way, meaning that the update of the global network will be conducted once all the agents have an update to perform. In the A2C method, a coordinator is added so that it waits for all the parallel workers to finish their segment of experience before updating the global network weights. Then, in the next iteration, a new segment of experience will begin with all parallel agents having the same policy as depicted in Figure 6.

With performance comparable to the A3C algorithm, the A2C implementation enables more consistent training due to synchronized gradient update.

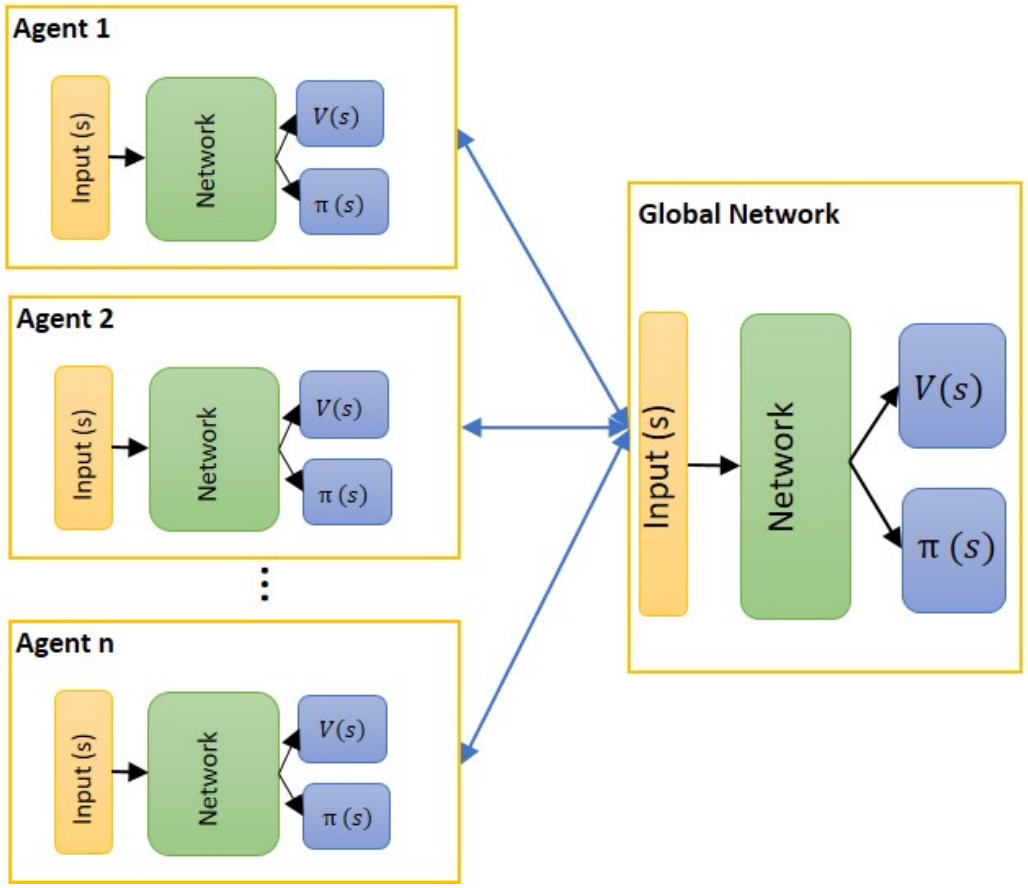

Figure 5. Simplified A3C Architecture.

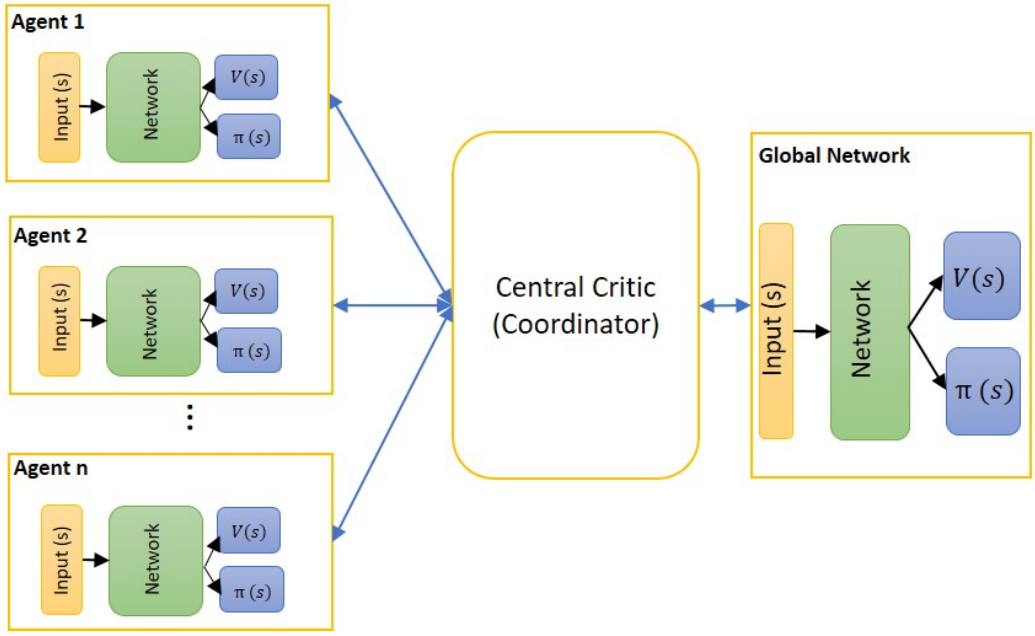

Figure 6. Simplified A2C architecture.

4. Problem Statement and Proposed Methods

The experiment was performed in a glass automotive company especially, within the heating process of tempered glass. The process consists on an electric heating furnace composed of nine control zones, a loading table upstream the furnace and an unloading table downstream the furnace for discharging tempered glass sheets (Figure 7).

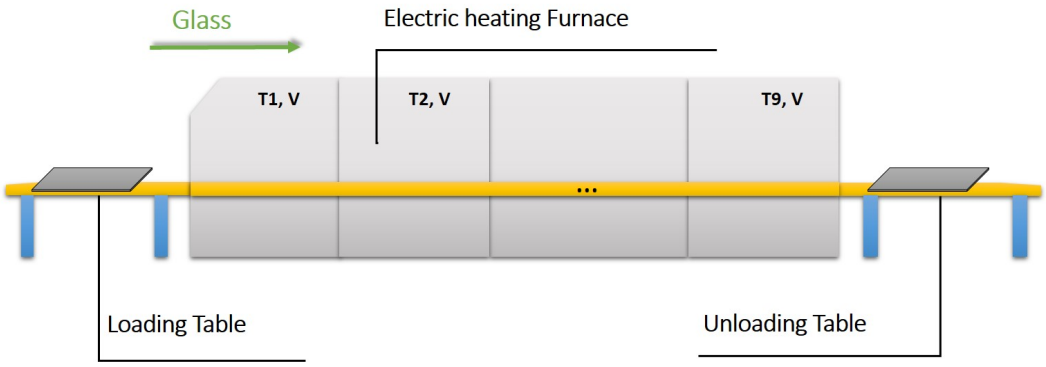

Figure 7. Industrial Electric furnace illustration.

Two main parameters characterize the nine control zones: the conveyor or the belt speed that also represents the glass speed through the furnace, and the temperature that is specific to each zone, from the nine control zones, and that may change from one zone to another. A new product means a new type of glass defined by a different thickness, a different color, or a slightly different geometry. Consequently, every type of glass has a specific recipe that consists of the glass speed and the temperature vector. The two parameters are generally fixed based on the expert process experience. Besides, some adjustments are then necessary to achieve the desired outlet temperature of the glass. Frequently, these adjustments are performed via trials and or destructive tests, which is time and resource-consuming. Furthermore, once the desired temperature of the glass is reached, the recipe is defined regardless of the relative energy consumption.

Our contribution consists of developing a decision model capable of mastering the process parameters to meet the two indexes; the quality index represented by the exit temperature of the glass and the energy index expressed by the overall energy consumption of the furnace for producing one piece.

The proposed system is a DRL algorithm dealing with our dual-optimization problem which consists of optimizing the overall energy consumption and controlling the process parameters of tempered glass to meet the quality requirements. In this work, we have implemented three DRL algorithms, which are PPO, A2C and DQN. Then based on these algorithms, we developed our decision model combined with a quality self-prediction model.

First, the ANSYS software (Canonsburg, PA, USA) was used to generate a set of background data that provides process parameters and the corresponding quality index. Second, using these background data, we trained a self-prediction model to predict the quality index for each set of parameters. Finally, we used this predictive model as our simulated environment (i.e., industrial furnace) for the training of our model, by applying the two proposed algorithms. Figure 8 provides a brief description of the steps needed to develop our decision model system. In the following three sections, a detailed explanation of the steps will be given.

4.1. Background Data Acquisition

Understanding the impact of the working conditions and the setting parameters on the electric tempering furnace's behavior is crucial to understand the tempering process and the relative energy consumption. To this end, the use of a simulation software, allowing mathematical modeling of the real asset to mimics as good as possible the real behavior of the furnace, seems to be a good solution. In the present work, the ANSYS simulation platform was used to simulate the behavior of glass in its working environment, given ANSYS' ability to provide high-fidelity virtual prototypes [46]. It consists on creating a virtual representation of the furnace based on mathematical modeling using finite elements. This model involves the physical laws of heat transfer that apply in the furnace in the heating process, namely, conduction, convection and radiation [47–49].

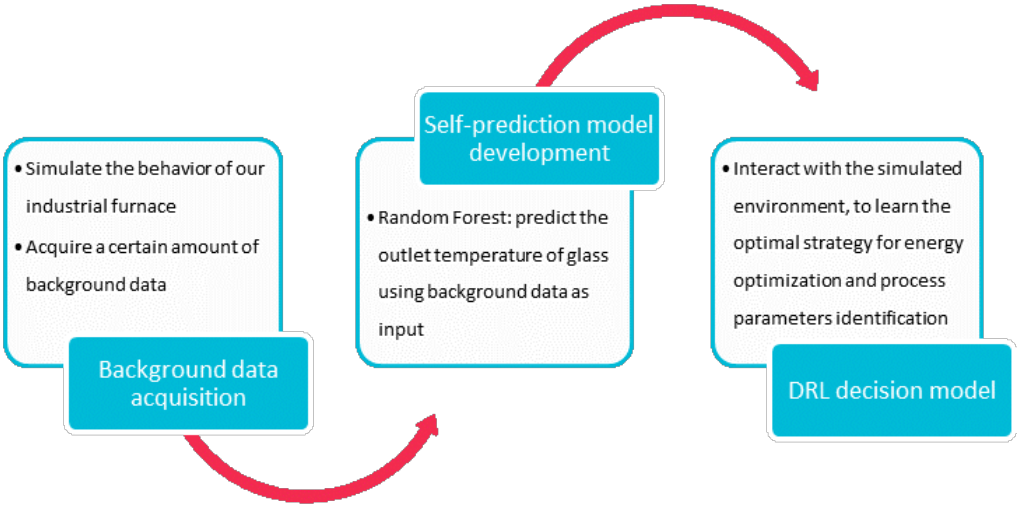

Figure 8. Overview of problem formulation steps.

The rate of radiation energy is given by the Stefan–Boltzmann law of radiation, and expressed as follows:

$$Q = \sigma e A T^4 \tag{11}$$

where σ is the Stefan–Boltzmann constant, e is the emissivity of the object, A is the radiating surface area, and T is its absolute temperature in kelvin.

The rate of heat transfer by convection is expressed by the following equation:

$$\dot{Q} = hA\triangle T \tag{12}$$

where $\dot{Q}$ is the heat transferred per unit time, h is the heat transfer coefficient, A is the surface area of the object, $\triangle T$ is the temperature difference.

The rate of heat transfer by conduction can be calculated by the following equation:

$$\frac{Q}{\triangle t} = -kA\frac{\triangle T}{\triangle x} \tag{13}$$

where $\triangle T$ is the temperature difference, A is the cross-sectional surface area, $\triangle t$ is the time interval during which the amount of heat Q flows through a cross-section of the material, and $\triangle x$ is the distance between the ends.

In addition to involving the physical laws of the heating process, the virtual model created by ANSYS faithfully reproduces the technical and geometric constraints of the real furnace.

Therefore, we used the simulation results to generate background data composed of the process parameters as inputs and its corresponding quality index as the dependent variable. Figure 9 describes the background data acquisition procedure performed using the ANSYS software (Canonsburg, PA, USA).

4.2. Self-Prediction Model Development

This step consists on providing a simulation environment for our RL agent using the background data collected in the first step. We tested different prediction mapping models to predict the quality index considering process conditions as model inputs. The models are trained using the background data and their accuracies were compared to choose the more accurate model. In this study, the Random Forest (RF) algorithm presents a good accuracy in predicting the quality index. RF is a robust supervised learning (ML) algorithm used for both regression and classification problems. It uses bagging and feature randomness to build an uncorrelated forest of decision trees, thus providing a better performance than

traditional ML algorithms [50,51]. In summary, the self-prediction RF algorithm will play the role of the simulated environment allowing the training of our DRL agents.

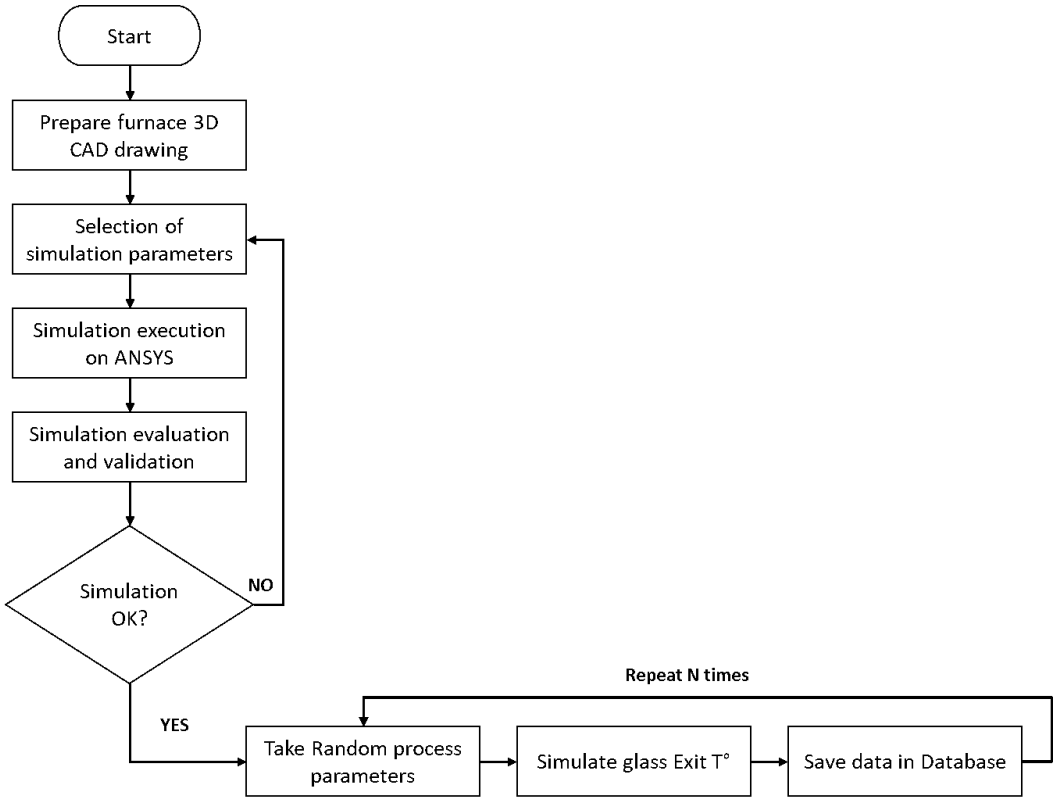

Figure 9. Flow chart of the data acquisition procedure.

4.3. RL Decision Model

As mentioned before, the trained RF model will be used as the simulated environment of the industrial furnace. Therefore, the last step is to develop an RL agent interacting with this simulated environment. The goal of our RL agent is to learn the optimal strategy for optimizing both the quality index and energy index. Figure 10 illustrates the proposed structure of our RL decision model.

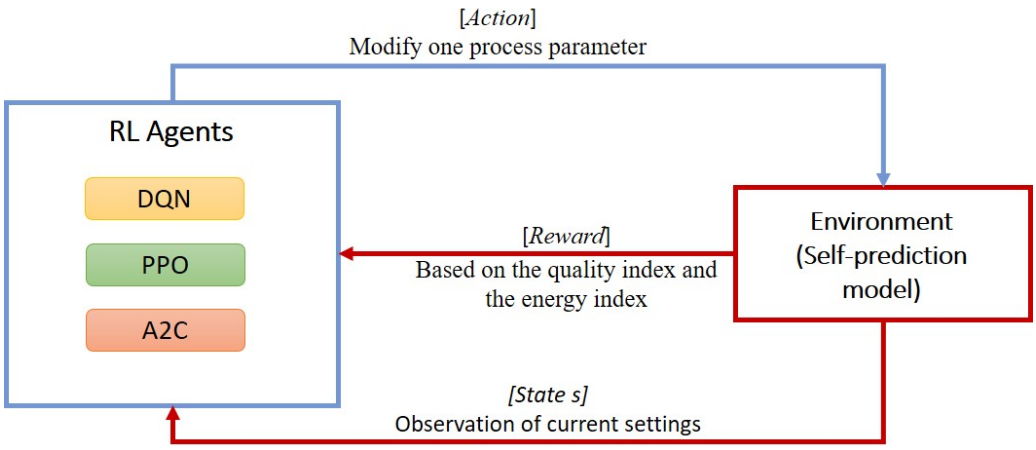

Figure 10. Proposed structure of RL decision model.

State space S

The state vector is composed of nine values of zone temperature, the glass speed, and the quality index as follows:

$$s_t = (T_1,\ T_2,\ \ldots,\ T_9,\ v,\ T_{exit}) \in S \tag{14}$$

Action space A

Considering our description of the problem, we identified the action space as a discrete set of values used to adjust both the glass speed and the zone temperature T_i. The agent can take an action that consists of increasing or decreasing glass speed or zone temperature. The set of possible actions is denoted as follows:

$$Possible\ actions = (a_1,\ a_2,\ \ldots,\ a_{20}) \tag{15}$$

where $a_i \in \{a_1,\ \ldots,\ a_9\}$ refers to the action increase the temperature of the ith zone by 10 °C, and $a_j \in \{a_{11},\ \ldots,\ a_{19}\}$ refers to the action decrease the temperature of the $(j-10)$th zone by 10 °C, while a_{10} refers to increase glass speed by 10 mm, and a_{20} refers to decrease glass speed by 10 mm.

Reward function R

Once an action is taken by the RL agent, a new set of process parameters is generated, and the quality index (T_{exit}) is predicted bu the RF model. Consequently, the quality of the action is expressed by a numerical reward received by the agent. Furthermore, at the end of the episode, the reward is adjusted to involve the energy index as well. Thus, the reward function is given by:

$$r_t = \frac{\left|PreviousT_{exit} - T_{target}\right| - \left|CurrentT_{exit} - T_{target}\right|}{T_{target}} + r_e \tag{16}$$

Based on the quality standards set by the quality department, the quality index is a priority, and it is defined as good when it is in range $\left[T_{target} - 5\ °\text{C},\ T_{target} + 5\ °\text{C}\right]$. In other words, if the quality index does not meet the quality requirement of the process, then the episode reward is null. Else, the episode reward is calculated as follows:

$$r_e = \begin{cases} 0 \quad if\ not\ done \\ 1 - (\sum_{i=0}^{8} T_i \times 0.01 + v \times 0.005)/C_{max} \end{cases} \tag{17}$$

where C_{max} is the maximum cost possible that relies on the energy consumption and is calculated as follows:

$$C_{max} = T_{max} \times Nbr\ of\ zones \times 0.01 + v_{max} \times 0.005 \tag{18}$$

The coefficients 0.01 and 0.005 were defined based on the internal historical data of the company, and they reflect the amount of energy consumed successively by the heating elements namely, resistances and the engines that generate the movements.

In order to avoid unnecessary trials, some assumptions are added:

$T_{min} = 500\ °\text{C};\quad T_{max} = 700\ °\text{C};\quad v_{min} = 150\ \text{mm};\quad v_{max} = 200\ \text{mm}$

Explicitly, if action a_2 was performed in the current state, the agent will move to a new state noted:

$$s_{t+1} = (T_1,\ T_2 + 10,\ T_3,\ \ldots,\ T_9,\ v,\ NewT_{exit}) \tag{19}$$

This means changing from the current set of process parameters to a new set by increasing the temperature of zone 2 by 10 °C, and replacing the current quality index by $NewT_{exit}$, the new quality index received from the self-prediction model. Thus, based on this predicted quality index, the agent will receive a reward r_t from the environment.

Further, by the end of every episode, and if the quality index is OK, the reward is adjusted to involve the energy index as well. Hence, the training process will ensure the achievement of optimal process parameters that optimize the quality index while guaranteeing significant energy savings.

5. Experiment Validation and Result

In the previous section, we have defined and explained all the theoretical statement of the studied problem. The current section will focus more on their practical side, providing the necessary explanations of the experiment settings. First, we performed a set of simulations on ANSYS to acquire background data needed to train both self-prediction model and RL agent. Next, we developed three versions of RL algorithms namely; DQN, PPO, and A2C to evaluate the performance of our decision model.

5.1. Offline Simulation and Prediction Model Training

The experimental case study was presented in Section 5. As a reminder, it consists of similar pieces of automotive glass (car sidelites and backlites) that undergo a thermal tempering inside an industrial electric furnace. The furnace is composed of nine zones characterized by their temperature values. Every type of glass has a specific passage speed in the furnace. The temperature values plus the glass speed are defined as the process parameters that we will try to identify in the present work in order to optimize both the quality index and the energy index. Given a new product, the temperature value of the glass at the furnace's output is affected by its tolerance interval in general of ± 5 °C. The purpose of the present work is to achieve self-control and self-optimization of process parameters to meet the quality requirements while optimizing energy consumption. It typically consists of solving a dual optimization problem.

Background data collection is the first step in the development of our decision model, but it is also a crucial task requiring high-fidelity simulations to provide meaningful data for the predictive task and the learning task. First, we created a virtual electric furnace, simulating the significant characteristics of the real furnace and reproducing its behavior under working conditions. The behavior of the real furnace was simulated using a set of features such as physical and thermal laws that describe and mimic what actually happens in the furnace during a heating process. Numerous tests was performed to achieve the final version of the simulated furnace which provides results very close to the behavior of the real furnace (see Table 1).

Table 1. The selected glass characteristics and mesh size for simulations.

Characteristic	Value	Unit
Isotropic thermal conductivity	1.4	W m^{-1} K^{-1}
Specific heat	750	J kg^{-1} K^{-1}
Mesh size	1 division following X and Y	—
	33 divisions following Z	—
Emissivity	0.9	—

Secondly, we checked the relevance and accuracy of the virtual furnace simulations by injecting a set of process parameters that was currently in production and for which we knew the glass outlet temperature. For example, in Table 2, we provide the process parameters for a given glass model for which the corresponding outlet temperature is 687 °C.

Figure 11 shows the result obtained using our simulated furnace, it can be seen that the temperature of the glass is 689.75 °C and then it is included in the interval of tolerance which is [687−5, 687+5].

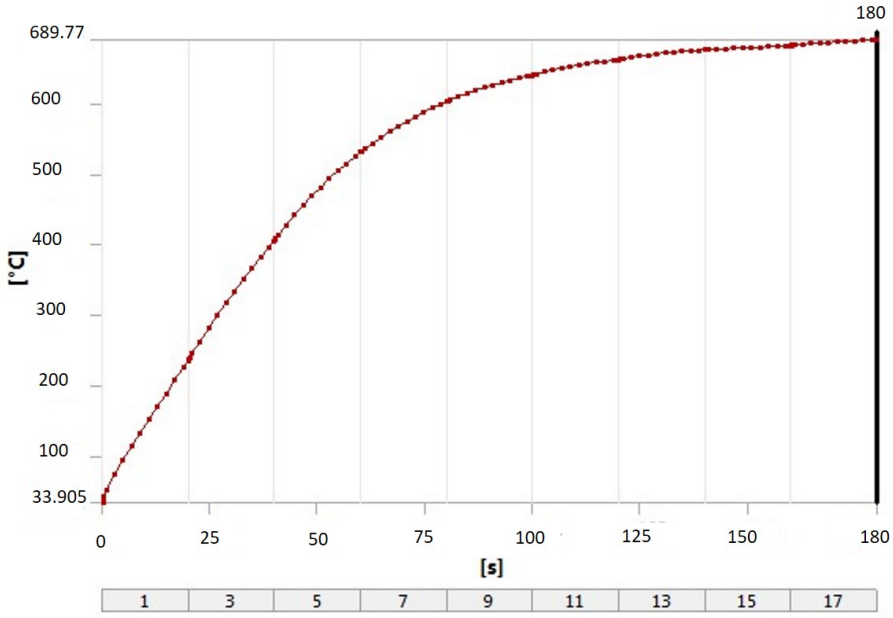

Figure 11. Simulation result for the given glass model.

Table 2. Process parameters for the studied glass model.

Zone	1	2	3	4	5	6	7	8	9
Temperature (°C)	637	652	664	666	670	678	685	685	695
Glass Speed (mm)	20	20	20	20	20	20	20	20	20

Thirdly, after validating our simulated furnace, we performed a hundred simulations to extract background data. Then the self-prediction model (i.e., Random Forest) was trained using the extracted data. Figure 12 presents the graph that compares predictions with target values over 60 observations, the model accuracy is clearly highlighted with a mean absolute error MAE = 0.806. Accurately trained, this predictive model will allow us, subsequently, to predict the outlet temperature of the glass for any set of process parameters. Hence, this model constitutes, coupled with the reward function, the simulated environment with which our RL agent will interact.

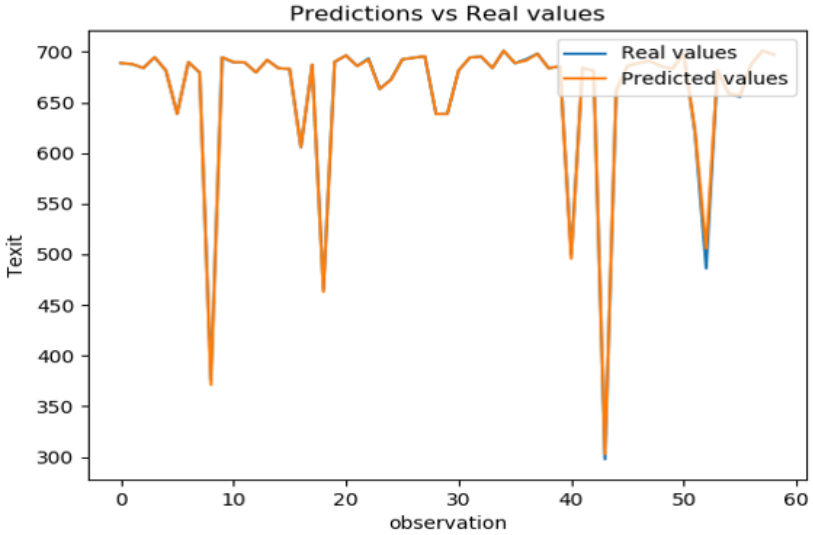

Figure 12. Plot of the predicted values versus real values (Target values).

5.2. Decision System

We used the implementation in the OpenAI Baselines project [52] to evaluate our selected RL algorithms. Interfacing with OpenAI Gym [53] provides various advantages especially by providing actively developed interface with multiple environments and features useful for training. The interface, also, provides a reference implementation of RL algorithms. For the three algorithms: DQN, PPO, and A2C, we set the controller policy to the default multi-layer perceptron (MLP), and we used the default hyper-parameters except for the learning rate and the number of simulation steps that are set, respectively, to 0.00001 and 1000 steps. Training and tests were run on Windows 10 with an eight-core i7-7700 CPU and an NVIDIA GeForce GT 1060 graphics card. The total time to execute 1 million steps was 4 h 40 min for DQN, 2 h 30 min for PPO and 3 h 57 min for A2C.

The training process consists, for several iterations, in performing a set of actions using the simulated environment under the three configurations. After training, different plots of the performance as well as other factors vital to the learning process, such as episode reward, losses, average last 100 steps reward, are gathered from TensorBoard. All graphs presented below are smoothed using the moving average, ensuring clarity of the data presented and good visualization of trends.

Figure 13, shows the performance results of the A2C agent during the learning process. Starting with the episode reward (Figure 13A), it oscillates slightly but in a decreased trend before getting increased around 400,000 time steps. The value loss function (see Figure 13C) shows that the agent performs perfectly before 250,000 time steps, then the performance drops drastically for the rest of the learning process, which is also confirmed while analyzing the plot of the average last 100 steps reward as depicted in Figure 13D. Since we are using the A2C from baselines implementation, the sudden explosion of the value loss might be connected to the entropy regularization. In short, A2C agent tends to learn a policy at the beginning but it suddenly diverges around 250 K time steps.

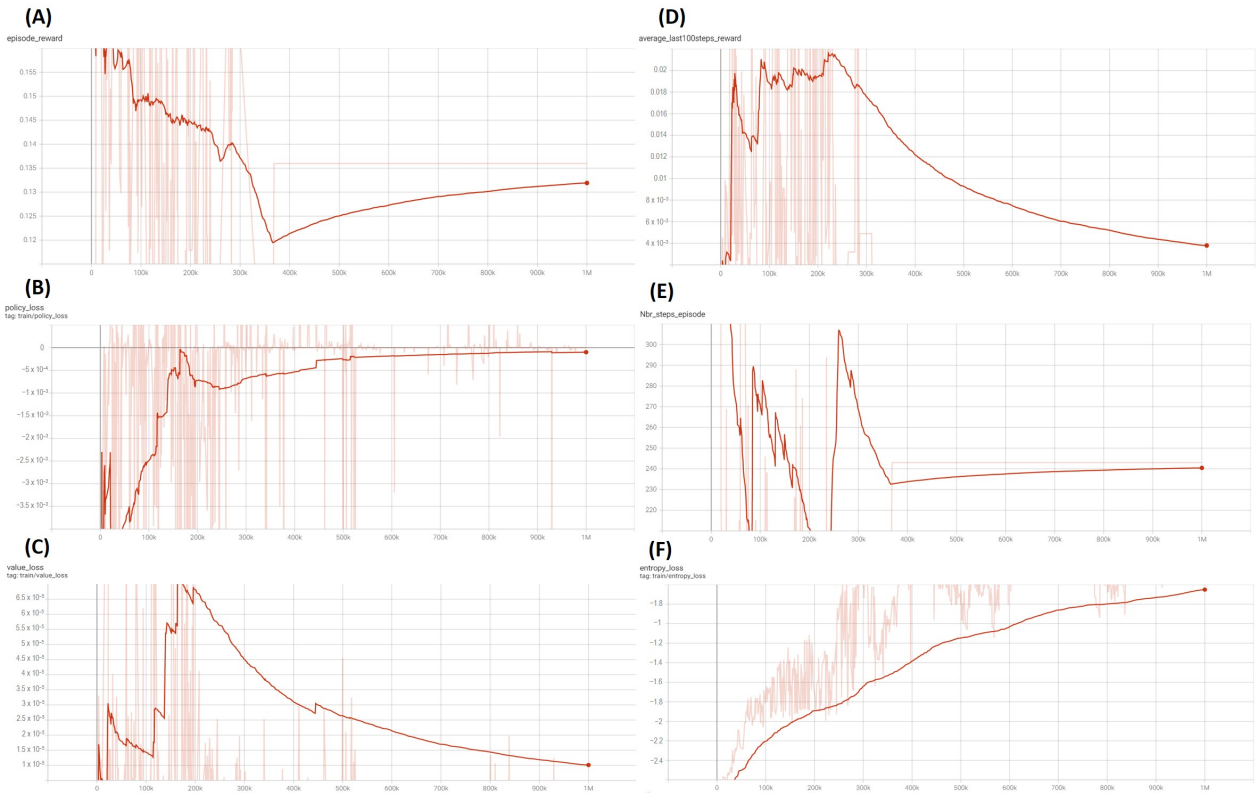

Figure 13. A2C results (per time steps): (**A**) Episode reward, (**B**) Policy loss, (**C**) Value loss, (**D**) Average last 100 steps reward, (**E**) Number of steps per episode, (**F**) Entropy loss.

Performance results of the PPO agent, during the training, are depicted in Figure 14. Upon initial review, PPO results seems to be much more promising. During the first learning phase (especially the first 400.000 actions), it can be seen that the reward rises significantly before start decreasing continuously. This is because the model free reinforcement learning needs to fully explore the entire state space to successfully learn a high-performance behavior, which is clearly observed during the remaining training process. From 400 K time steps, the reward increases drastically and converges.

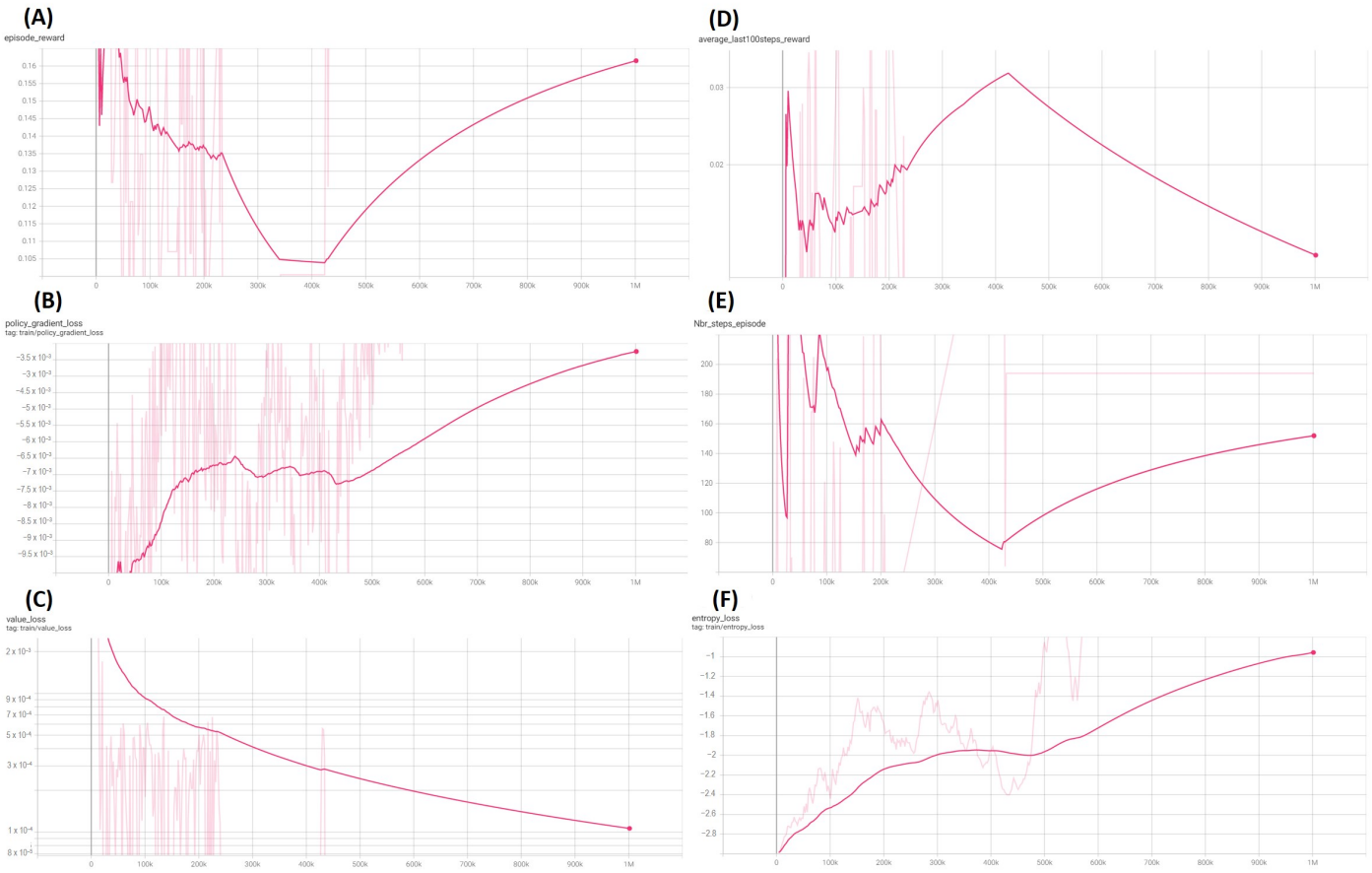

Figure 14. PPO results (per time steps): (**A**) Episode reward, (**B**) Policy gradient loss, (**C**) Value loss, (**D**) Average last 100 steps reward, (**E**) Number of steps per episode, (**F**) Entropy loss.

On the contrary, the oscillation phase of the DQN agent is less violent and shorter (around 200.000-time steps). It is characterized, in general, by less fluctuating and more stable reward (see Figure 15). Eventually, after 100,000-time steps, the reward starts improving and converging to an optimal policy that allows optimal process parameters' control for a given product. We can observe that the DQN agent converges more rapidly than PPO agent, but both have a good performance. Additionally, as we mentioned before, the DQN agent makes almost 2 h more in training for the same number of iterations. To prevent over-fitting, the training stops once the reward converges.

A brief comparison of the episode reward of A2C, DQN and PPO is presented in Figure 16, it can be observed that the learning curve of DQN is more stable but with lower performance. Due to exploding gradients, A2C and PPO showed less stable learning behavior during the first part, while the PPO curve suddenly shows a drastically hill-climbing process (around 400 K time steps).

In addition, we plotted the two performance metrics to advance the evaluation of our models, namely, energy cost and target temperature (see Figure 17). The purpose of our decision model is to minimize energy consumption while ensuring good quality expressed, in our case study, by reaching the target temperature. The energy cost obtained

by A2C, DQN and PPO agents is shown in Figure 17A. From the first glance, it is visible that the DQN agent performs poorly in the energy optimization task by providing higher energy cost during the whole learning process. However, the PPO agent learns the optimal policy to provide the lowest energy cost. On the other hand, Figure 17B, shows the target temperature metric for the three agents. Again, PPO agent outperforms DQN and A2C in reaching the desired temperature value in the present work.

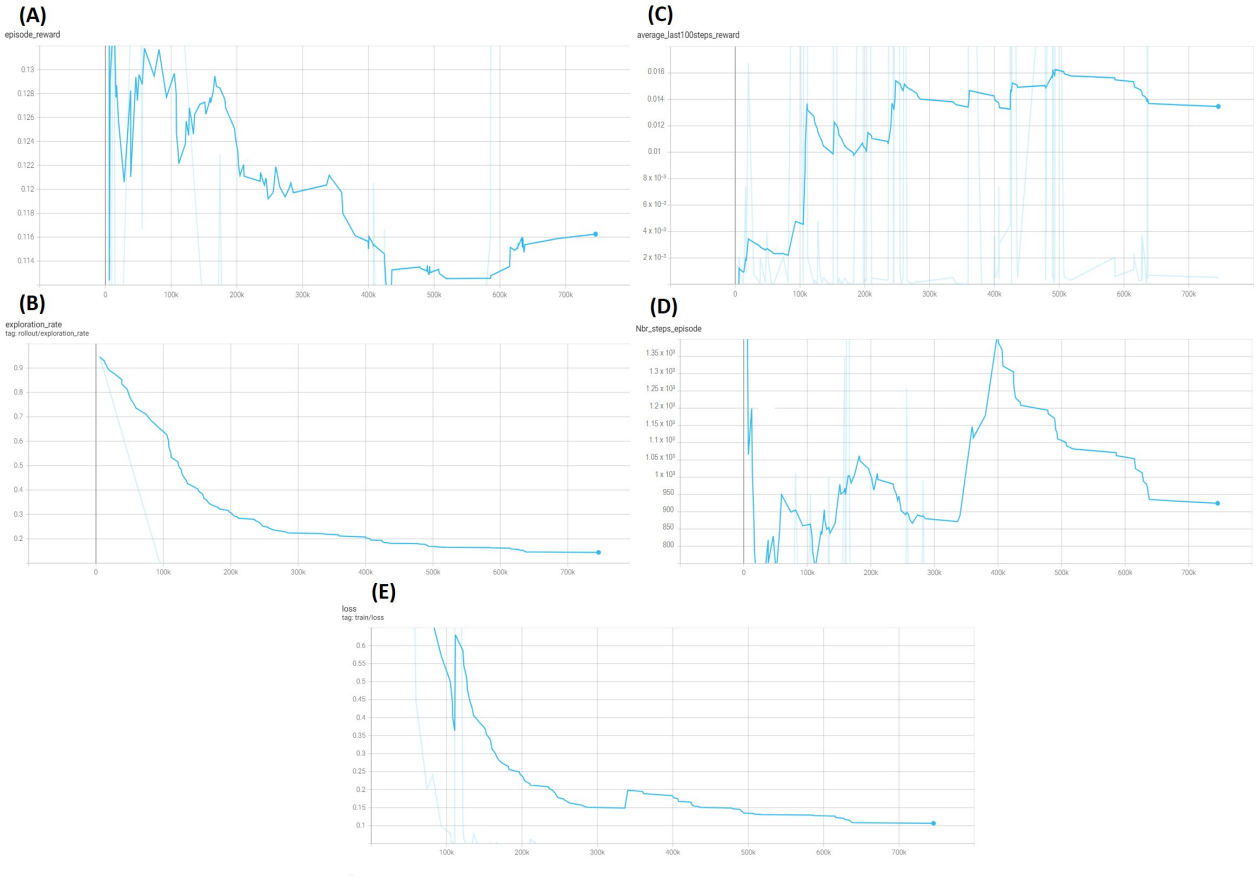

Figure 15. DQN results (per timesteps): (**A**) Episode reward, (**B**) Exploration rate, (**C**) Average last 100 steps reward, (**D**) Number of steps per episode, (**E**) loss.

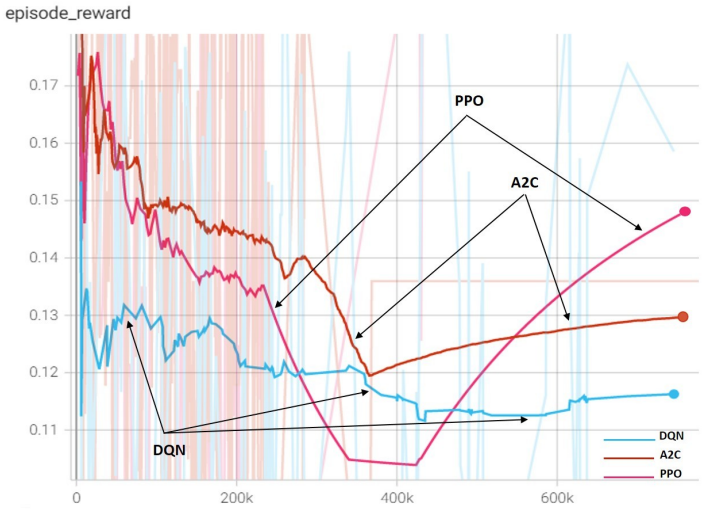

Figure 16. Comparison of the Episode reward of the three models

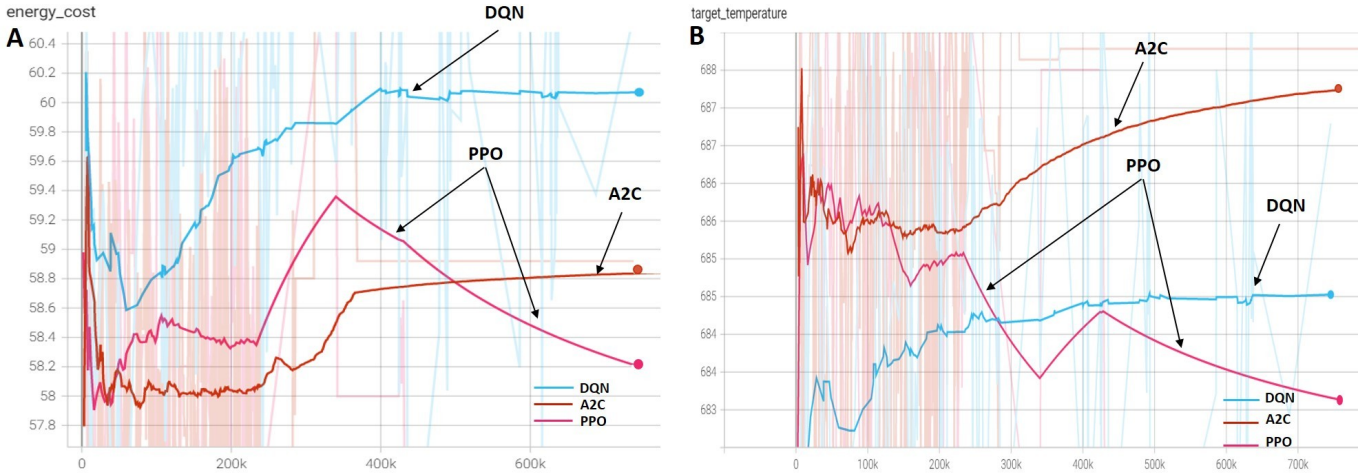

Figure 17. Performance metrics: (**A**) Energy cost, (**B**) Target temperature.

In summary, PPO and A2C are policy-based models, which means that they are efficient in learning stochastic policies with stochastic and high-dimensional action spaces. However, A2C showed poor performance and divergence after a given number of episodes, which is probably due to the fact that we are dealing with a double optimization problem with a value function that is relatively difficult to approximate. Conversely, the PPO algorithm achieves a balance between efficiency and performance when dealing with the current double optimization problem. At the same time, the DQN algorithm showed better stability compared to the other two models, which is the main characteristic of value-based RL techniques.

In addition, the robustness of the proposed decision model was tested for some process parameter configurations suggested by the heating process team and provided good results, especially with the PPO agent. However, no coherent and reasoned result is to be provided in this sense. The proposed decision model should be implemented in real time for a better and substantial evaluation.

Admittedly, the energy index of our decision system model must be developed and treated in depth, regarding the different laws of thermodynamics, to come up with robust configurations especially with the reward function. In this sense, the verification and validation of the relevance of the reward function, concerning the energy part, must be carried out by real applications in collaboration with the appropriate department of the company. Further works and research needs to be conducted in this direction providing plenty of insights that will lead to progress towards sustainable smart manufacturing.

The paper solved a dual optimization problem using three DRL models and obtained satisfactory results that certainly could be improved in future works. The main contribution of the present work is not only providing the basis statements and formulation of process parameters control when dealing with energy optimization in the industrial context, but also developing, training and testing the models in the real environment and proving their accuracies. The developed decision model is able to be generated to other processes and industries by conducting a detailed study of the given process, and bringing the necessary changes to the model.

6. Conclusions and Future Work

This paper has optimized the overall energy consumption by successfully applying three RL algorithms, namely, A2C, DQN and PPO. We have applied our self-optimization system to the glass tempering process. The developed decision system consists in solving a dual-optimization problem by identifying dynamically and automatically the process parameters that optimize the energy while ensuring the product quality. Validation experiments were carried out over three steps: First, background data were extracted using ANSYS simulations. Second, based on the extracted data, the self-prediction model was

trained to obtain accurate predictions of the glass exit temperature reaching a MAE of 0.806. Finally, using the prediction model as a simulated furnace environment, the RL decision model was trained to learn the optimal strategy for solving the previously mentioned dual-optimization problem without any prior expert knowledge. Based on the result of the experiment, A2C is rejected due to divergence, while PPO and DQN agents showed fast and stable convergence with a slight difference which makes DQN more stable and PPO faster. In the two remaining configurations, our decision model has demonstrated its ability to effectively replace the traditional method of identifying process parameters aiming for a high level of flexibility in a cost-effective manner. In future work, we would like to further investigate the energy optimization problem by improving the reward function and introducing real world constraints.

Author Contributions: Conceptualization, C.E.M. and T.M.; methodology, C.E.M.; software, C.E.M.; validation, T.M., I.E.H. and N.B.; formal analysis, C.E.M.; investigation, C.E.M.; resources, C.E.M.; data curation, C.E.M.; writing—original draft preparation, C.E.M.; writing—review and editing, C.E.M.; visualization, C.E.M.; supervision, T.M.; project administration, I.E.H.; funding acquisition, None All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Rabbani, M.; Mohammadi, S.; Mobini, M. Optimum design of a CCHP system based on Economical, energy and environmental considerations using GA and PSO. *Int. J. Ind. Eng. Comput.* **2018**, *9*, 99–122. [CrossRef]
2. Thiede, S.; Herrmann, C. Simulation-based energy flow evaluation for sustainable manufacturing systems. In Proceedings of the 17th CIRP International Conference on Life Cycle Engineering, LCE 2010, Hefei, China, 19–21 May 2010; pp. 99–104.
3. Seefeldt, F.; Marco, W.; Schlesinger, M. *The Future Role of Coal in Europe*; EUROCOAL: Berlin, Germany; Basel, Switzerland, 2007.
4. Muhuri, P.K.; Shukla, A.K.; Abraham, A. Industry 4.0: A bibliometric analysis and detailed overview. *Eng. Appl. Artif. Intell.* **2019**, *78*, 218–235. [CrossRef]
5. Lu, Y.; Xu, X.; Wang, L. Smart manufacturing process and system automation-a critical review of the standards and envisioned scenarios. *J. Manuf. Syst.* **2020**, *56*, 312–325. [CrossRef]
6. Zobeiry, N.; Humfeld, K.D. A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications. *Eng. Appl. Artif. Intell.* **2021**, *101*, 104232. [CrossRef]
7. Chakraborty, S. Simulation free reliability analysis: A physics-informed deep learning based approach. *arXiv* **2020**, arXiv:2005.01302.
8. Kober, J.; Bagnell, J.; Andrew, P.J. Reinforcement learning in robotics: A survey. *Int. J. Robot. Res.* **2013**, *32*, 1238–1274. [CrossRef]
9. Szepesvári, C. Algorithms for reinforcement learning. *Synth. Lect. Artif. Intell. Mach. Learn.* **2010**, *4*, 1–103.
10. Kaelbling, L.; Pack, L.M.L.; Moore, A.W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285. [CrossRef]
11. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
12. Mcmaster, R.A. Fundamentals of tempered glass. *Ceram. Eng. Sci. Proc.* **1989**, *10*, 193–206.
13. Zhang, X.; Hao, H.; Wang, Z. Experimental investigation of monolithic tempered glass fragment characteristics subjected to blast loads. *Eng. Struct.* **2014**, *75*, 259–275. [CrossRef]
14. Gardon, R. Thermal tempering of glass. In *Glass Science and Technology*; Uhlmann, D.R., Kreidl, N.J., Eds.; Elsevier: Amsterdam, The Netherlands, 2014.
15. Rantala, M. Heat Transfer Phenomena in Float Glass Heat Treatment Processes. Doctoral Thesis, Tampere University of Technology, Tampere, Finland, **2015**.
16. Mazgualdi, C.E.; Masrour, T.; Hassani, E.; Khdoudi, A. A Deep Reinforcement Learning (DRL) Decision Model for Heating Process Parameters Identification in Automotive Glass Manufacturing. In Proceedings of the International Conference on Artificial Intelligence & Industrial Applications, Meknes, Morocco, 19–20 March **2020**; pp. 77–87.
17. Moreira, L.C.; Li, W.D.; Lu, X.; Fitzpatrick, M.E. Energy-Efficient machining process analysis and optimisation based on BS EN24T alloy steel as case studies. *Robot. Comput. Integr. Manuf.* **2019**, *58*, 1–12. [CrossRef]
18. Hajabdollahi, F.; Hajabdollahi, Z.; Hajabdollahi, H. Soft computing based multi-objective optimization of steam cycle power plant using NSGA-II and ANN. *Appl. Soft Comput.* **2012**, *12*, 3648–3655. [CrossRef]
19. Seo, K.T.F.; Edgar, M.B. Optimal demand response operation of electric boosting glass furnaces. *Appl. Energy* **2020**, *269*, 115077. [CrossRef]
20. Geng, Z. Energy optimization and prediction modeling of petrochemical industries: An improved convolutional neural network based on cross-feature. *Energy* **2020**, *194*, 116851. [CrossRef]

21. Su, Y. Multi-objective optimization of cutting parameters in turning AISI 304 austenitic stainless steel. *Metals* **2020**, *10*, 217. [CrossRef]
22. Sangwan, K.; Singh, N.S. Multi-objective optimization for energy efficient machining with high productivity and quality for a turning process. *Procedia CIRP* **2019**, *80*, 67–72. [CrossRef]
23. Wang, W. Dual-objective program and improved artificial bee colony for the optimization of energy-conscious milling parameters subject to multiple constraints. *J. Clean. Prod.* **2020**, *245*, 118714. [CrossRef]
24. Luan, X. Trade-off analysis of tool wear, machining quality and energy efficiency of alloy cast iron milling process. *Procedia Manuf.* **2018**, *26*, 383–393. [CrossRef]
25. Han, Y. Energy management and optimization modeling based on a novel fuzzy extreme learning machine: Case study of complex petrochemical industries. *Energy Convers. Manag.* **2018**, *165*, 163–171. [CrossRef]
26. Golkarnarenji, G. Support vector regression modelling and optimization of energy consumption in carbon fiber production line. *Comput. Chem. Eng.* **2018**, *109*, 276–288. [CrossRef]
27. Bian, S.; Li, C.; Fu, Y. Machine learning-based real-time monitoring system for smart connected worker to improve energy efficiency. *J. Manuf. Syst.* **2021**, *61*, 66–76. [CrossRef]
28. Lopez-Martin, M.; Carro, B.; Sanchez-Esguevillas, A. Application of deep reinforcement learning to intrusion detection for supervised problems. *Expert Syst. Appl.* **2020**, *141*, 112963. [CrossRef]
29. Pane, Y.P.; Nageshrao, S.P.; Kober, J. Reinforcement learning based compensation methods for robot manipulators. *Eng. Appl. Artif. Intell.* **2019**, *78*, 236–247. [CrossRef]
30. Xuan, H.; Zhao, X.; Fan, J.; Xue, Y.; Zhu, F.; Li, Y. Vnf service chain deployment algorithm in 5g communication based on reinforcement learning. *IAENG Int. J. Comput. Sci.* **2020**, *48*, 1–7.
31. Rhazzaf, M.; Masrour, T. Smart Autonomous Vehicles in High Dimensional Warehouses Using Deep Reinforcement Learning Approach. *Eng. Lett.* **2021**, *29*, 1–9.
32. Otterlo, M.; Wiering, M. Reinforcement learning and Markov decision processes. In *Reinforcement Learning: State-Of-The-Art*; Wiering, M., Otterlo, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 3–42.
33. White III, C.C.; White, D.J. Markov decision processes. *Eur. J. Operational Res.* **1989**, *39*, 1–16. [CrossRef]
34. Bellman, R. Dynamic programming. *Science* **1966**, *153*, 34–37. [CrossRef] [PubMed]
35. Watkins, C.J.C.H.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
36. Werner, H.; Ehn, G. Reinforcement Learning for Planning of a Simulated Production Line. Master's Theses, Lund Central Station, Lund, Sweden, 2018.
37. Jain, A.K.; Mao, J.; Mohiuddin, K.M. Artificial neural networks: A tutorial. *Computer* **1996**, *29*, 31–44. [CrossRef]
38. Khdoudi, A.; Masrour, T.; Mazgualdi, C. Using Machine Learning Algorithms for the Prediction of Industrial Process Parameters Based on Product Design. In Proceedings of the International Conference on Advanced Intelligent Systems for Sustainable Development, Marrakech, Morocco, 8–11 July 2019; pp. 728–749.
39. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
40. Sutton, R.; Mcallester, D.; Singh, S.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Adv. Neural Inf. Process. Syst.* **1999**, *12*, 1057–1063.
41. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust region policy optimization. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 1889–1897.
42. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
43. Li, Y.; Chen, Y. Enhancing A Stock Timing Strategy by Reinforcement Learning. *IAENG Int. J. Comput. Sci.* **2021**, *48*, 1–10.
44. Grondman, I.; Busoniu, L.; Lopes, G.A.; Babuska, R. A survey of actor–critic reinforcement learning: Standard and natural policy gradients. *IEEE Trans. Syst. Man Cybern. Part C* **2012**, *42*, 1291–1307. [CrossRef]
45. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1928–1937.
46. Lee, H.H. *Finite Element Simulations with ANSYS Workbench 18*; SDC Publications: Kansas, MO, USA, **2018**
47. Bergman, T.L.; Lavine, A.S.; Incropera, F.P. *Introduction to Heat Transfer*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
48. Dhir, V.K. Boiling heat transfer. *Annu. Rev. Fluid Mech.* **1998**, *30*, 365. [CrossRef]
49. Mills, A.F. *Heat Transfer*; CRC Press: Boca Raton, FL, USA, 1992.
50. Ali, J. Random forests and decision trees. *Int. J. Comput. Sci. Issues* **2012**, *9*, 272.
51. Mazgualdi, C.E.; Masrour, T.; Hassani, I.E.; Khdoudi, A. Machine learning for KPIs prediction: A case study of the overall equipment effectiveness within the automotive industry. *Soft Comput.* **2021**, *25*, 2891–2909. [CrossRef]
52. Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; Wu, Y.; Zhokhov, P. Openai Baselines: High-Quality Implementations of Reinforcement Learning Algorithms. Available online: https://github.com/openai/baselines (accessed on 5 September 2022).
53. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai gym. *arXiv* **2016**. arXiv:1606.01540.