A Mixture-of-experts Approach to Production Capacity Planning for Diverse Demand Patterns via Deep Reinforcement Learning

Thachanon Danket [1], Sansiri Tanachutiwat[2], Vichai Rungreunganun[1*]

[1] Department of Industrial Engineering, Faculty of Engineering, King Mongkut's University of Technology North Bangkok, Bangkok, 10800, Thailand
[2] The Sirindhorn International TGGS, King Mongkut's University of Technology North Bangkok, Bangkok, 10800, Thailand

*Corresponding Email : vichai.r@eng.kmutnb.ac.th

Received April 16, 2024, Revised April 24, 2024, Accepted May 3, 2024, Published June 14, 2024

Abstract. *Ensemble Learning is gaining traction in Reinforcement Learning (RL) due to its ability to improve performance, robustness, and capabilities of RL models. This paper addresses the challenge of production planning with fluctuating demand by proposing a novel Mixture of Experts Deep Reinforcement Learning (MoE-DRL) model. We leverage a combination of Proximal Policy Optimization (PPO), a powerful reinforcement learning algorithm, and Ensemble learning, a technique that combines multiple models. We propose a mixture of expert ensemble learning model that combine multiple expert PPO-DRL agents through a gating model (MoE PPO-DRL). The gating model learns to select the best expert agent for predicting the most suitable production plan for each situation's different demand patterns. The proposed model was trained and tested against the results obtained from the Mixed Integer Linear Programming model and the individual expert PPO agents. The MoE PPO-DRL model achieved a total average profit that was 25.9% higher than an average of all expert single-agent models. It also achieved a 11.02% optimality gap, which is significantly lower than the 22.93% average gap of all expert single-agent models.*

Keywords:

Production capacity planning, Deep reinforcement learning, Ensemble learning, Mixture of experts, Proximal policy optimization, Mixed integer linear programming

1. Introduction

The cement industry, a significant heavy industry sector, contributes substantially to numerous economies worldwide [1]. Developing efficient production planning models for cement plants can revolutionize this industry. Cement plants exemplify complex production systems, demand volatility, and applicability to other industries. Research from this case study can inform effective reinforcement learning models for production planning in cement plants and other industries facing similar challenges [2].

Production capacity planning (PCP) is crucial for factories to determine the appropriate production levels to meet customer demand, considering the available production resources such as the number of machines and labor hours. Effective production planning enables production departments to meet customer demand in full and on time, reduce production costs, and improve operational efficiency, which ultimately leads to increased factory profitability [3]

PCP involves balancing numerous factors to find the best solution. As problems get more intricate, finding an optimal solution becomes exponentially harder due to the increasing number of variables and constraints. This complexity makes PCP problems NP-hard. While traditional optimization methods are still used for modelling [4-9], recent research explores metaheuristics to find near-optimal solutions faster for increasingly complex problems [10-15].

The application of artificial intelligence (AI) to tackle industrial problems is gaining traction. Within the realm of production planning and scheduling, Takeda-Berger [16] identified a surge in machine learning research, with 31 studies published between 2015 and 2020. Of these, Artificial Neural Networks (ANNs), regression models, tree-based algorithms, and genetic algorithms emerged as the dominant approaches.

A comprehensive review by Kulmer [3] analyzed research leveraging Reinforcement Learning (RL) and other methods like Genetic Algorithms (GA) and optimization techniques to tackle PCP problems. Their

search within the Scopus database up to November 2021 yielded 53 relevant articles. The review highlights RL's growing popularity for such dynamic problems, noting that RL outperforms rule-based approaches and heuristics in solution quality. They also noted that most RL research has focused on short-term PCP, leaving a research gap in medium-term applications.

In recent years, research has begun to explore the use of Reinforcement Learning (RL) principles to solve PCP problems. Hubbs [17] deep reinforcement learning (A2C) principles to solve the chemical production scheduling process. This is a production scheduling process in which one machine produces several products to meet the demand for the delivery of products in each period. The model took into account the uncertainty of the parameters that may change at each period from the online database system. The scheduling results with the RL model were benchmarked against a mixed-integer linear programming (MILP) model. The RL model achieved a gap of only 19% versus the optimum, demonstrating its effectiveness. They noted that RL has a significant advantage in the speed and flexibility of solving a scheduling system's real-time optimization compared to other methods. Mowbray [18] applied Distributional Reinforcement Learning to address production scheduling problems in the presence of uncertainty in chemical production processes. The results were benchmarked against MILP models.

In addition, several other research studies have applied RL principles to solve production planning and scheduling problems [3, 17-22]. Some studies have compared RL to meta-heuristic methods and found that RL can provide better-quality answers [21, 23].

However, RL algorithms can be inefficient and computationally expensive, especially in high-dimensional state spaces where input data exhibits high variance. This limitation hinders RL's applicability in real-world scenarios with complex dynamics [24]. Therefore, several methods have been developed to improve the performance of RL models and address the issues in such situations. Transfer learning utilizes knowledge learned from one task to improve performance on another. Curriculum learning gradually increases the difficulty of tasks for the RL agent. Exploration bonuses reward the RL agent for exploring new states and actions. Ensemble learning combines multiple models to improve overall performance.

Ensemble learning is a machine learning technique integrating multiple base learners to construct an ensemble learner. Its primary goal is to enhance the system's overall learning performance. This technique has garnered significant attention and achieved remarkable success in various artificial intelligence applications [25] Ensemble learning has been used in various research studies to improve the performance of prediction and classification tasks [26, 27].

Researchers have recently explored various advanced techniques of Ensemble Learning Algorithms to improve predictive performance beyond that of single-agent models. Adaptive Boosting (AdaBoost) has been employed as an EL mechanism, adjusting the weights of individual models within a boosting algorithm based on their performance [28, 29]. Gradient Boosting Machines (GBM) utilize decision trees as base learners, constructing models sequentially to correct errors in previous models [30]. XGBoost further enhances GBM by incorporating regularization and tree-pruning techniques [31]. The mixture of Experts (MoE) divides tasks into sub-tasks, training specialized models for each and employing a gating model to select the optimal expert for each input [32],[33].

Yuksel et al.[34] survey Mixture of Experts (MoE) models, discussing their use in regression, classification, and various training methods. They found that ME has been applied in various fields like electricity demand prediction, robot navigation, and financial forecasting. MoE can be used for optimization tasks where the goal is to find the best solution. By dividing the problem into subtasks. Cheng et al. [35] propose an attention-based approach for multi-task reinforcement learning. It utilizes separate experts for different task aspects and combines them with an attention mechanism, achieving better sample efficiency and performance in benchmark environments than baseline methods. This demonstrates the potential of applying MoE principles to RL. However, from our review of MoE and RL research, more research is still needed on applying MoE in conjunction with RL principles to improve the efficiency of solving industrial production planning problems. This is the gap that our research will fill.

This paper investigates the effectiveness of a Mixture of Experts Deep Reinforcement Learning (MoE-DRL) model for Production Capacity Planning (PCP) under fluctuating demand. Using a cement factory as a case study, we developed and trained five individual PPO agents within a custom PCP environment. Each agent was trained on a different demand dataset to specialize them in different demand patterns and then combined into an MoE-DRL model. The performance of the single PPO agents and the MoE-DRL model was compared

against an optimal Mixed Integer Linear Programming (MILP) model. This research explores the potential of Ensemble Reinforcement Learning models for PCP using a mixture of experts approach, paving the way for future advancements in complex PCP systems.

This paper makes significant contributions by pioneering developing a Mixture of Experts Deep Reinforcement Learning (MoE-DRL) model for predicting production planning outcomes within a custom-designed environment. This environment mirrors real-world production complexities, catering to discrete-time medium-term batch production capacity and inventory planning under fluctuating demand scenarios.

2. Problem Statement

This article presents a case study of the cement production process. Raw materials are continuously supplied to two parallel cement grinding machines (M1 and M2) using a predetermined mixture formula. Each machine can produce three types of products.

Currently, factory employees utilize a spreadsheet program to execute a computational planning method. This method employs an experiential heuristic approach to assign three products (P1, P2, and P3) production across two machines (M1 and M2) for each planning period. Employees will create a monthly production plan divided into 60 shifts or 30 days to maintain inventory levels and ensure that products are delivered to customers on time and in full. However, this manual planning method proves inefficient and inflexible when faced with fluctuating product demand data. This inflexibility hinders cost optimization and necessitates frequent re-planning.

The proposed models serve as prototypes for industrial plants, enabling enhanced production planning efficiency. The models can be generalized to similar production systems within the scope of this research.

3. Formulation of MILP

To establish an upper benchmark for DRL models, we propose a novel MILP model that incorporates time-varying electricity prices into a discrete-time multi-parallel machine production setting under fluctuating electricity pricing. The constraints and objective function were derived from a combination of the research of Castro, Dogan, and Zhao[4, 5, 36].

3.1 Inventory Balance Constraint

The quantity of products r remaining at the end of period t, denoted by $R_{r,t}$, can be computed by Eq. (1).

$$R_{r,t} = R_{r,t-1} + \sum_{m=1}^{M} P_{m,r,t} - d_{r,t} - L_{r,t} \quad \forall r \in P, t \in T \tag{1}$$

where :$R_{r,t-1}$ is the number of inventories carried forward from the previous period,

$P_{m,r,t}$ is the number of finished products received by machine production m,

$d_{r,t}$ is the product demand to be delivered to the customer during period t, and

$L_{r,t}$ is the shortage quantity to be delivered to the customer at that time.

3.2 Production Capacity Constraints

Let $P_{r,m,t}$ denote the production capacity of machine m for product r in period t. Each period t has a duration of Tt hours, encompassing both on-peak and off-peak periods. As production capacity constraints, Eq. (2) specifies the limitation on $P_{r,m,t}$ as follows.

$$P_{r,m,t} = T_t pr_{r,m} N_{r,m,t} \quad \forall r \in P, m \in M, t \in T \tag{2}$$

$$\sum_r N_{r,m,t} \leq 1 \quad \forall r \in P, \forall m \in M, t \in T \tag{3}$$

where: P_m is the production rate of product r (tons per hour) of the machine m, $N_{r,m,t}$ is a binary variable indicating whether the machine m produces the product r or not at period t. Eq. (3) determines the constraints for each machine to produce no more than one type of product per period.

3.3 Level of Inventory Constraints

Eq. (4) specifies the product minimum and maximum inventory level in its storage tank.

$$R_r^{\min} \leq R_{r,t} \leq R_r^{\max} \quad \forall r \in P, t \in T \tag{4}$$

where R_{rmax} is the cement storage tank capacity, while R_{rmin} is the minimum possible inventory level, which is zero since lost sales are allowed.

3.4 Switch On-Off Constraints

For each machine m, the decision variable for turning the machine on or off, denoted by $SW_{r,m,t}$, can be computed by Eq. (5 and 6).

$$SW_{r,m,t} = \sum_{r} N_{r,m,t} \quad at\ period\ t\ = 1 \qquad (5)$$

$$SW_{r,m,t} \geq \sum_{r} N_{r,m,t} - \sum_{r} N_{r,m,t-1} \quad otherwise \qquad (6)$$

3.5 Objective Equation

The objective function aims to maximize total profit within the production capacity and inventory planning environment E. By letting G = [$N_{r,m,t}$|∀r ∈ R, m ∈ M, t ∈ T] be a production planning, we define the total profit as the gap between the total revenue and the incurred total cost, as shown in Eq. (7).

$$\Omega(G|E) = (\sum_{r\in R}\sum_{t=1}^{T} p_r(d_{r,t} - L_{r,t})) - \Psi(G|E) \qquad (7)$$

where: p_r is the price of product r [USD/ton],

$d_{r,t}$ is the demand of product r at period t [ton],

$L_{r,t}$ is the lost sale of product r due to stockout [ton], given in Eq. (2).

$\Psi(G|E)$ is the cost function for a production planning [USD].

The total cost for a production planning G with respect to the environment E is further defined as follows.

$$\Psi(G|E) = \sum_{r\in R}\sum_{m=1}^{M}\sum_{t=1}^{T}(cu_r P_{r,m,t} + ce_t pw_m P_{r,m,t} + f_m N_{r,m,t} + sc_m SW_{m,t}) + \sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{r\in R} co_{r'}\hat{C}_{r'} + \sum_{r\in R}\sum_{t=1}^{T} h_r R_{r,t} + \sum_{r\in R}\sum_{t=1}^{T} lsc_r L_{r,t} \qquad (8)$$

where: cu_r is the unit cost of product r [USD/ton],

ce_t is the electricity cost at during time interval t [USD/kWh],

pw_m is the amount of electricity used by machines m to produce product [kWh/ton],

f_m is the fixed cost incurred when there is a production on machine m [USD],

sc_m is the costs incurred from re-setup the machine m for production after it was stopped from the previous period [USD],

$co_{r'}$ is the cost incurred by changing production from producing any product to producing r' [USD],

$\hat{C}_{r',m,t}$ *is the changeover binary variable indicates the transition from producing product type r to product type r'*

h_r is the holding cost of product r per one period [USD/ton], and

lsc_r is the penalty cost per ton for the shortage of product r [USD/ton].

The cost calculation parameters are based on real-world data collected during the cement production process described in Section 2. To find the optimal planning that maximizes the total profit concerning environment E, the objective equation can be written as Eq. (9) subject to the constraints in Eqs. (1) to (6)

$$G^* = \underset{G}{argmax}\,\Omega(G|E) \qquad (9)$$

4. Reinforcement Learning Models

4.1 Reward Functions

The reward function guides the RL agent toward the desired goal (Sutton & Barto, 2018). We define the reward function of our RL model as the total profit function.

$$Q = \Omega(G|E) \qquad (10)$$

where Ω is the total profit defined in Eq. (10), and G represents production capacity planning, derived from the action trajectory generated by the policy learned from environment E. This equation incorporates total revenue and cost, resembling the objective function of the MILP model.

4.2 Proximal Policy Optimization (PPO)

PPO, introduced by Schulman [37], represents a significant advancement in policy-based methods, outperforming algorithms like A2C [38]. Our implementation is based on Lapan's [39] code with our algorithm modifications. PPO aims to maximize policy improvement, as formulated below.

$$J(\theta' \mid \theta) = \mathrm{E}_{s_t\sim p, a_t\sim \pi_\theta}[I(\theta' \mid s_t, a_t, \theta)] \qquad (11)$$

where I is the improvement ratio of the new parameters θ' given the state s_t, the action a_t, and the previous parameters θ:

$$I(\theta' \mid s_t, a_t, \theta) = \frac{\pi_{\theta'}(a_t|s_t)}{\pi_\theta(a_t|s_t)} A(s_t, a_t|\pi_\theta) \qquad (12)$$

where $A(s_t, a_t|\pi_\Theta)$ is the advantage function of the policy π_Θ

To limit the size of gradient updates, The PPO method added a term to the equation for clipping the

objective. Therefore, the objective function (11) is converted to Eq. (13):

$$J_{\text{CLIP}}(\theta' \mid \theta) = \mathrm{E}_{s_t \sim p, a_t \sim \pi_\theta}[\min[I(\theta' \mid s_t, a_t, \theta), g(\varepsilon, A(s_t, a_t \mid \pi_\theta))]] \quad (13)$$

where g is the clipping function defined below:

$$g(\varepsilon, A(s_t, a_t \mid \pi_\theta)) = \begin{cases} (1+\varepsilon)\, A(s_t, a_t \mid \pi_\theta) & \text{if } A(s_t, a_t \mid \pi_\theta) \geq 0 \quad (14) \\ (1-\varepsilon)\, A(s_t, a_t \mid \pi_\theta) & \text{if } A(s_t, a_t \mid \pi_\theta) < 0 \quad (15) \end{cases}$$

The objective of the old and the new policy ratio is limited to the range $1-\varepsilon$ to $1+\varepsilon$. Therefore, adjusting ε will affect the learning performance.

4.3 Mixture of Experts (MoE) model

A mixture of experts (MoE) is one type of ensemble learning model that utilizes a trainable gating network to combine classifiers through weighted rule assignment. Each expert focuses on a specific subtask. Trained via decision trees, expectation-maximization (EM), or neural networks, the gating network assigns weights to each expert based on the input data. This allows MoE to efficiently learn the feature space and act as a classifier selection system, choosing the most relevant expert for the task [40].

$$(y_i|x_i, z_i) = g \sim f_g\left(y_i \middle| \theta_g(x_i)\right), \quad P(z_i = g|x_i) = n_g(x_i) \quad (16)$$

Equation 16 expresses the output y of a neural network as a weighted sum of expert agent predictions, where the weights represent the probabilities of selecting each expert. To predict an output y from an input feature vector x, a Mixture of Experts (MoE) employs a gating network (NN) to combine the predictions of multiple expert models ($\boldsymbol{\theta}$ or Thetas). Each expert model $\boldsymbol{\theta}$, potentially a neural network, processes x and generates a prediction. These predictions are then fed into a corresponding gating network NN, which weights each expert's prediction based on the input data. Finally, the weighted predictions are aggregated to produce the final output. Unlike traditional ensemble methods, which simply average predictions, MoE dynamically selects the most relevant expert for each input, improving performance.

5. Ensemble MoE Deep Reinforcement Learning Formulation

This section describes the components and steps involved in creating an Ensemble learning - Mixture of experts DRL model. The process begins with establishing a production environment for training agents. Next, a learning mechanism for the PPO-DRL model is designed. Individual PPO-DRL models are trained on each dataset to obtain expert agents for each type of demand data. Finally, all generated expert agents are combined to form an Ensemble learning - A mixture of expert models.

5.1 Creating an Training Environment of Expert Single Agent

The environment defines the rules and limitations of the agent's actions, prompting the agent to learn and select the most rewarding action within constraints [38, 41]. For this study, the production capacity and inventory planning environment is modeled as the production system (see Section 2) to train individual DRL agents. The agent's action space consists of 16 discrete actions shown in **Fig. *1***. **Fig. 2** illustrates the interaction pattern between the environment and the agent learning. The agent receives state information in the form of 79 observational parameters. These variables can be divided into four groups based on product r ∈ [1, 3], machine m ∈ [1, 2], and future period k ∈ [1, 2], these variables can be divided into four groups:

- Inventory level of each product: I_r and I_r^k
- Incoming shipment demand for each planning period: d_r and d^k_r
- The demand type randomly selected for each episode in the simulation: D_{p}
- Machine production information: $N_{m,r}$
- Electricity rates during the planning period: E_{on}

These variables and parameters serve as inputs to the agent with the neural network mechanism in the RL model. The agent utilizes these inputs to create and develop a policy that guides the selection of one among 16 actions. As the action is made, it results in

$$\mathbf{a}^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \mathbf{a}^{(2)} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \mathbf{a}^{(3)} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \mathbf{a}^{(4)} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$
$$\mathbf{a}^{(5)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{a}^{(6)} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{a}^{(7)} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{a}^{(8)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$
$$\mathbf{a}^{(9)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{a}^{(10)} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{a}^{(11)} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{a}^{(12)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$\mathbf{a}^{(13)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{a}^{(14)} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{a}^{(15)} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{a}^{(16)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Fig. 1 List of actions. For an action matrix a, each element $a_{m,r}$ represents a flag for machine m to produce the product type r.

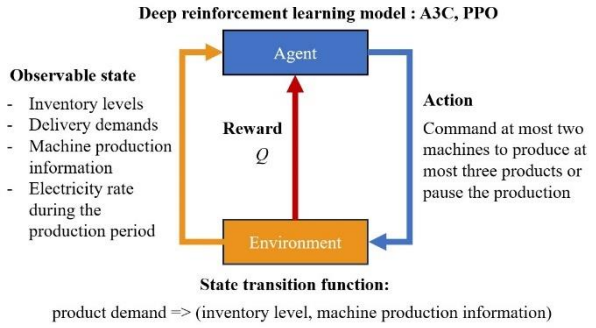

Fig. 2 Interaction between the learning agents and the environment.

state transition and the reward is computed in Eq. (10). The reward and observation of transitioned state s_{t+1} obtained from the environment will be sent back to the agent for processing and updating the policy to choose an action that gives better rewards.

During agent training, each agent is exposed to various product demand patterns, limited to 2-3 patterns each. The overall framework encompasses eight distinct demand patterns. For this experiment, four expert agents are trained on specific demand patterns, as shown in **Table *1***. The experimental data is comprised of random variation (RV) with values ranging from a specified minimum to maximum and seasonal indices with different peak levels: low seasonal index (LSI), medium seasonal index (MSI), and high seasonal index (HIS). Average demand values are randomized within specified intervals per delivery period within each demand pattern to reflect real-world scenarios. Each episode in the agent's training consists of 60 steps/states representing 60 production planning periods (1 month). In each state, the production quantity is determined by the agent's action and used to calculate state parameters like inventory balance and incurred costs, similar to the MILP model's equations.

Table 1 Types of data patterns used to train expert agents

Agent	Data		Max Season Index	Min value	Max Value
	Data Set	Type			
Agent1	1	RV	-	2,233	3,233
	2	RV	-	1,333	3,533
Agent2	3	LSI	1.31	2,167	2,500
	4	LSI	1.40	2,167	2,500
Agent3	5	MSI	1.43	2,167	2,500
	6	MSI	1.47	2,167	2,500
Agent4	7	HSI	1.55	2,167	2,500
	8	HSI	1.57	2,167	2,500

5.2 Expert Agents Training

This section focuses on training a single reinforcement learning (RL) agent using the Proximal Policy Optimization (PPO) algorithm. The goal is to develop an expert agent that experiences a diverse dataset of policy values under simulated conditions that vary across episodes. The trained expert agent's converged parameters will lead to an effective planning solution that minimizes total planning costs. During training, the agent explores various actions until the action distribution converges to a set of potential actions.

The hyperparameters play a crucial role in determining the learning performance of the agent. The discount factor and learning rate are two of the most important hyperparameters that affect learning [42]. In our experiments, we used a trial-and-error method to adjust the hyperparameters using Hubb's [17] default values. We observed the convergence characteristics of the reward curves until we found the appropriate hyperparameter values for the PPO models, which are shown in **Table 2**.

The architecture of a neural network, particularly its depth, is a critical factor that profoundly influences its learning capacity. we designed our policy network (actor) as a multilayer perceptron (MLP) with six hidden layers (820, 820, 656, 524, 420, and 328 nodes). The output layer has 16 nodes or units reflecting the log probability for each action. The value network (critic) has four layers (820, 820, 820, and 656 nodes), and the output layer has one node representing one scalar value specifying the corresponding value of a state. Since the actor and critic networks have more than three layers, we can identify our model as a deep reinforcement model (DRL). [43].

In the PPO model training, we set the learning rate to 5×10^{-6} to allow the agent to learn gradually. The discount factor γ was set to 0.95 to simulate a long enough trajectory of states. The clipping value 0.1 helps the PPO's agent appropriately limit the gradient update size.

5.3 Mixture of DRL-PPO Experts Model (M-DRL-PPO-E) Formulation

An ensemble of four pre-trained expert agents was constructed using the Mixture of Experts (MoE) model. The MoE model can learn via a neural network using the Proximal Policy Optimization (PPO)

algorithm, similar to single expert agents. Consequently, the model can learn a policy for selecting the most appropriate expert for each type of product demand encountered in each episode.

The MoE model incorporates a Gating agent as a central processing unit for selecting an expert agent. The selected expert agent executes a planning action in the PCP environment for an entire episode. Subsequently, the episode reward is returned to the Gating agent to refine its policy for selecting agents in subsequent rounds. **Fig. 3** illustrates the structure and components of the MoE model for this research. The diagram illustrates the workflow structure of the model. The model consists of a gating agent that acts as an agent selector. This agent learns to select the appropriate local or expert agent to generate production plans that align with the emerging product demand patterns. The performance of these plans is measured by the reward, which is the net profit obtained from the production plan. A detailed workflow of MoE is illustrated in **Fig. 4**.

The performance of a trained Mixture of Experts (MoE) model in production planning is assessed using a diverse set of demand data with varying patterns. The following section presents the results of training both single agents and MoE and model comparisons between individual expert agents, MoE, and the MILP model (used as the Upper upper-bound optimal solution).

6. Experiment Results

6.1 Single Expert PPO Model Training Convergence

All DRL models and MILP training and testing experiments were performed on a 2.30 GHz Intel i7-11800H CPU with 16 hyperthreads equipped with an NVIDIA RTX-3050 Ti GPU card with 2,560 cores at 1485 MHz and 4GB of VRAM. The MILP model was optimized to the optimal gap of 1% using Pulp 1.6.10 [44]. The proposed DRL models were implemented with PyTorch 1.7.0 [45] on Python 3.10 [46].

Using the hyperparameters settings in **Table 2**. Both models are trained by having the agent perform incremental learning in each training round until it yields the highest convergence stable reward. **Fig. 5** show an example of the convergences of the reward

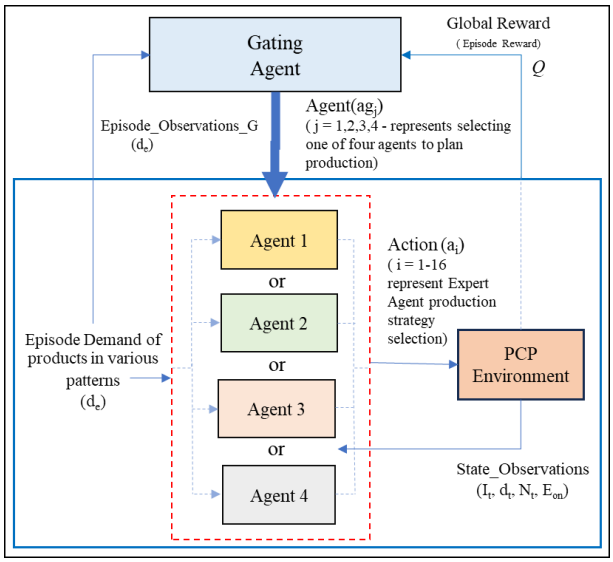

Fig. 3 The structure and components of the MoE model for this research

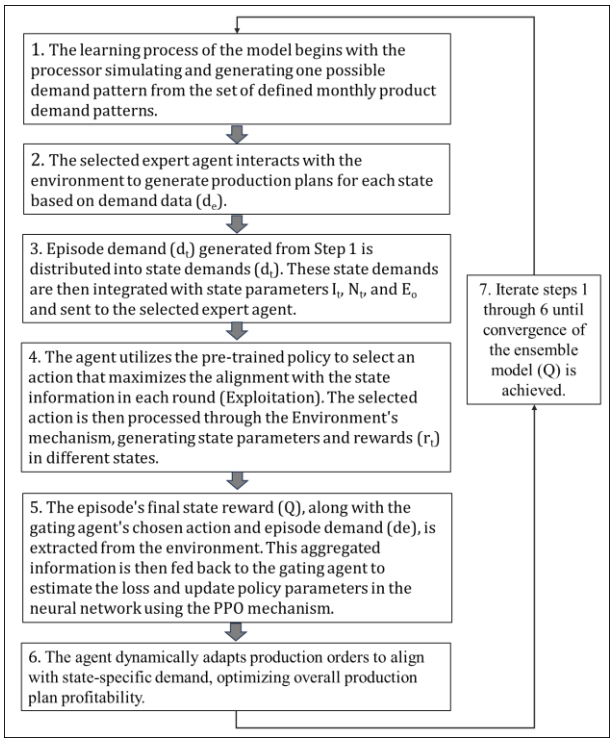

Fig. 4 A detailed workflow of MoE

from one PPO expert model called Agent1 (out of Agents 0 to 4). In **Fig. 5**a, the Agent1 model converges at 90 training time steps and yields consistent rewards hereafter. To validate the stability of our model, we trained it for a total of 12 hours, or 900 time steps (1,800,000 episodes). As shown in **Fig. 5**b, the reward quickly stabilizes at a value of 8.71 million USD, indicating that the model has reached a state of optimal performance.

The PPO model's stable reward value is due to its use of a surrogate clipping objective function mechanism. This mechanism enables the PPO to make more aggressive policy updates while still ensuring learning stability. As a result, the PPO agent is better able to accommodate input data with high variation, such as customer demand.

In addition to the previously trained Agent1, three additional expert models (Agent 2,3 and 4) were trained until convergence under different data configurations, as detailed in **Table *1***. Other single expert agents also exhibit similar convergence behaviors to Agent 1

Table 2 Hyperparameter values used to train the agent of each DRL model.

Types	Hyperparameters	Values
Common	Dimension of input vectors	27
	Actor's hidden layers	6
	Critic's hidden layers	4
	Units in the first hidden layer	820
	Entropy regularization	$1x10^{-4}$
	Dropout probability	0.2
	Discount factor	0.95
	Upper bound (U)	2,100
PPO	Learning rate	$5x10^{-6}$
	Number of epochs	30
	Early stopping (time steps)	900
	Batch size	100
	Number of trajectories	2,000
	Clipping value	0.1

6.2 Mixture of Expert PPO Models Training Convergence

Similar to single-agent training, the gating agent in the MoE model is trained to select the appropriate expert agent for each situation. Since the MoE learning mechanism employs the PPO algorithm, similar to single expert agents, most of the hyperparameters used are the same as those used in single expert agent training (**Table 2**).

The MoE model was trained using the architecture shown in Fig. 3 and the training procedure outlined in Fig. 4. As illustrated in **Fig. 6**, the model converged after 16,000 episodes, and was further trained for up to 400,000 episodes to confirm convergence and stability.

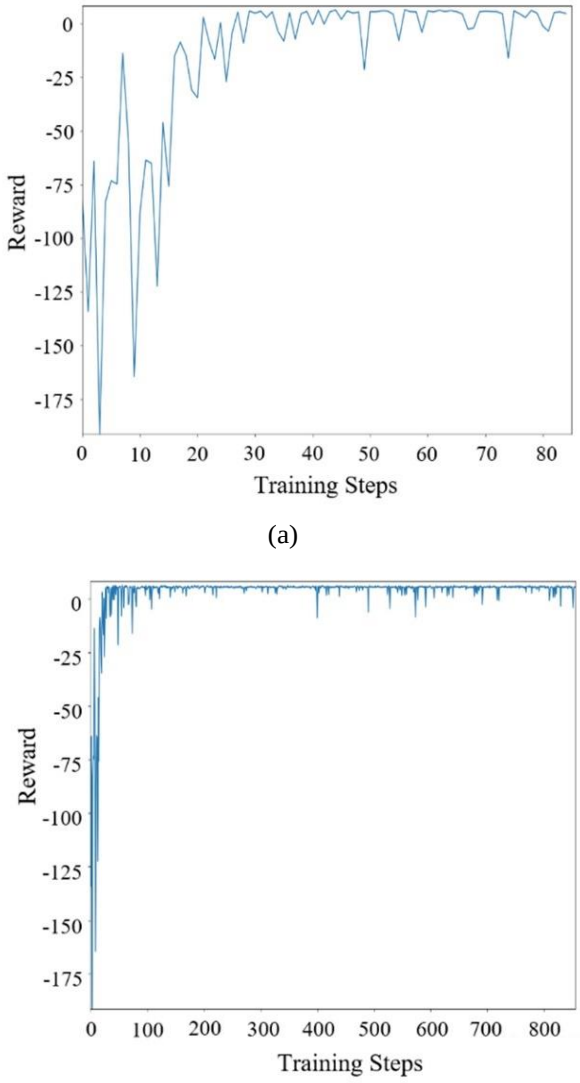

Fig. 5 Reward convergence of Agent1 Expert model trained for the first 90 (a) and 900 training time steps (b)

6.3 Testing Models Efficiency

To evaluate the performance of our models, we simulated eight demand datasets. The MILP model was used as an upper benchmark to compare the reward results.

Demand datasets for testing the models: To simulate the uncertainty of customer demand, we prepared eight datasets with different demand patterns. Each dataset was re-simulated under the same framework with the same min-max and peak seasonal index values as Table 1.This resulted in eight demand datasets for testing the performance of the MoE model

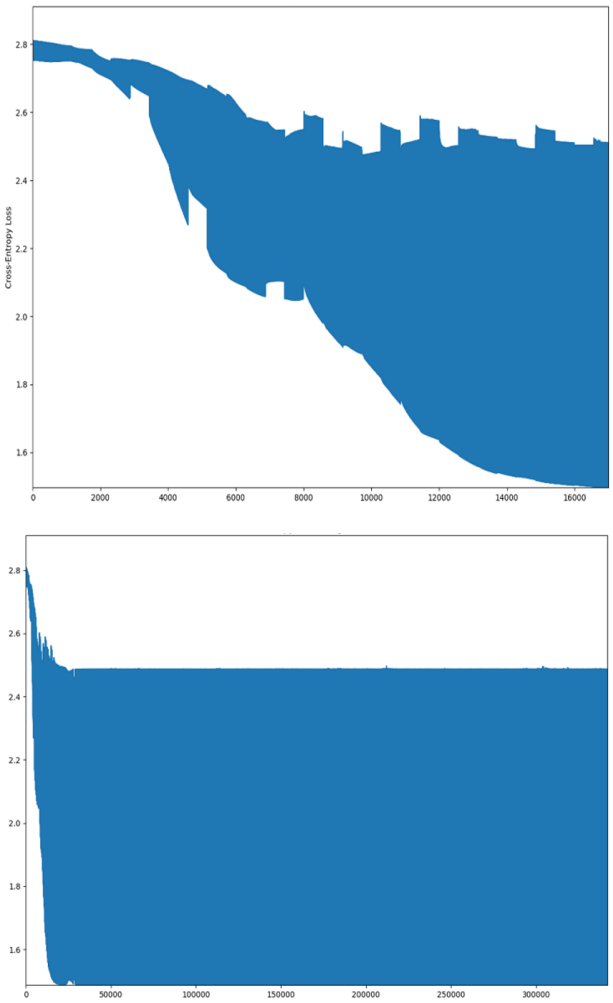

Fig. 6 Cross entropy loss convergence of Mixture of Experts model

compared to each individual expert agent. Agent0 was also included in the comparison, which is an expert agent trained on all eight datasets simultaneously.

Reward and Optimality gap: The reward values obtained for each dataset of every model are shown in **Table 3**. The MILP model can find the best answer for production planning from all demand data sets, representing an average total production profit of 9.92 million USD. The MoE model achieves an average total profit of 8.83 from the production plan, outperforming all individual expert agents (Agent 0-4). Further analysis of individual data sets reveals that Agent 1 performs poorly on data sets with high seasonality indices (Sets 5 to 8). Agents 2 to 4 struggle with random variation data due to their training on data sets with seasonality patterns.

The values in **Table 4** align with those in **Table 3**, representing the % Optimality gap of each model's profit compared to the MILP model. As shown, the MoE model achieves a %Gap of 11.02, which is the closest to the MILP model among all models.

Interestingly, Agent 0, trained on all eight data types, performed less effectively on datasets 7 and 8 than the MoE model. This is likely due to the increased data type diversity, making it challenging for a single agent to generate policies that effectively cover all scenarios.

Computing time: **Table 5** compares the efficiency of models in terms of the time taken to find the solution. The PPO and MOE have an average prediction time of 69.06 and 65.92 seconds per round. This is significantly faster than the MILP model about 2502 times, which takes an average of 178,000 seconds per round. However, it is essential to note that PPO and MoE require 43,740 and 40,400 seconds of training before they can be used to find a solution.

Table 3 Episode reward of production plans from different models under various data patterns

Testing Data			Average Reward from Each Model					
Set	Pattern	MILP	A0	A1	A2	A3	A4	**MoE**
1	RV	9.48	8.77	8.59	8.66	8.25	8.63	**8.55**
2	RV	9.64	-0.14	8.86	4.35	7.54	8.36	**8.63**
3	LSI.	10.08	8.63	8.62	8.81	8.83	8.71	**8.59**
4	LSI.	9.95	8.35	8.69	8.35	8.96	8.63	**8.61**
5	MSI.	10.18	9.28	-15.11	9.34	9.81	9.28	**9.46**
6	MSI.	10.00	8.90	-2.79	8.95	9.04	9.14	**9.11**
7	HSI.	9.98	3.11	-11.22	8.59	9.52	7.49	**8.92**
8	HSI.	10.05	6.23	-2.38	8.48	8.39	9.14	**8.76**
Average		9.92	6.64	0.41	8.19	8.79	8.67	8.83

Table 4 Percent Optimal Gap compares the reward value of DRL models with MILP

Testing Data		Model					
Set	Pattern	A 0	A1	A2	A3	A4	**MoE**
1	RV	9.56	9.36	8.69	12.99	8.93	**9.84**
2	RV	101.48	8.05	54.88	21.80	13.31	**10.50**
3	LSI.	14.46	14.52	12.61	12.41	13.65	**14.78**
4	LSI.	16.09	12.62	16.09	9.97	13.25	**13.51**
5	MSI.	8.87	248.42	8.24	3.69	8.89	**7.11**
6	MSI.	10.98	127.89	10.55	9.58	8.63	**8.95**
7	HSI.	68.82	212.36	13.94	4.66	24.97	**10.65**
8	HSI.	38.00	123.65	15.64	16.49	9.10	**12.82**
Average		33.53	94.61	17.58	11.45	12.59	**11.02**

Table 5 Model performance in terms of average training time and prediction time.

	MILP	PPO	MoE
Training time (s)	-	43,740	**40,400**
Prediction time (s)	172,800	69.06	**65.92**

6.4 Experimental Results Discussion

The experimental results validate the hypothesis that RL algorithms can be inefficient and computationally expensive, particularly in high-dimensional state spaces with high-variance input data. Agent 0, trained on a diverse set of demand data formats, exemplifies this challenge, exhibiting suboptimal performance due to the high variance of the input data. In contrast, the MoE model demonstrates significant improvements in handling all demand data formats. It achieves a higher average profit and lower optimality gap compared to Agent 0 and all individual agents. These findings underscore the effectiveness of MoE-DRL, particularly its ability to adapt to changing demand patterns. The integration of ensemble learning with RL strengthens the model's knowledge of the study and enables it to outperform traditional methods and individual RL agents.

Utilizing the model in actual situations is similar to the step-by-step training method shown in Figure 3. However, the train gating agent command is turned off, allowing the model to enter testing mode, which generates production plans based on various user input demand patterns into the system. This will generate production orders for each period throughout the month, as described in the production planning problem in Section 2. Each production plan prediction takes approximately one minute (as shown in Table 5). Additionally, if the demand pattern changes from the previously trained pattern or if the environment parameters change, users can train new expert agents and add them to the model. These principles make the model flexible and efficient for use in real-world situations and provide a roadmap for future model improvements.

6.5 Model Limitations and Future Directions

The MoE model's performance depends on the number of data formats assigned to each expert agent for pre-training. Therefore, future research directions should focus on enhancing the model's capabilities by incorporating a wider variety of demand data formats and increasing the number of expert agents. This would allow the model to handle the diverse demand patterns encountered in real-world scenarios more effectively.

The study also acknowledges the inherent limitations of DRL models. While PPO and MoE models offer faster predictions than MILP, they may compromise on solution quality. A key advantage of DRL lies in its ability to adapt to complex environments. Unlike MILP, DRL agents can continuously learn and improve through interaction, making them well-suited for real-world applications with dynamic and uncertain conditions.

7. Conclusions

This paper introduces the Mixture of Expert Deep Reinforcement Learning (MoE-DRL) model for production planning. The model is embedded within a custom-designed environment replicating real-world production intricacies and handles discrete-time medium-term batch production capacity and inventory planning under fluctuating demand scenarios. The proposed method effectively balances production costs and inventory levels, outperforming traditional simulation and real-world experiment methods. This research's contribution is the pioneering application of MoE in conjunction with DRL principles to enhance the efficiency of solving industrial production planning problems. This approach aims to be implemented in real-world factory planning and to further develop the model's capabilities in the future.

Our proposed model was evaluated against traditional Mixed-Integer Linear Programming (MILP) and individual expert PPO agents. The MoE PPO-DRL model achieved significant improvements compared to single-agent models. It delivered a total average profit 25.9% higher and achieved an 11.02% optimality gap, which is substantially lower than the average 22.93% gap observed with single-agent models. These results demonstrate the effectiveness of MoE PPO-DRL in production planning under fluctuating demand. Combining ensemble learning with RL, the model offers superior performance compared to traditional methods and individual RL agents.

PPO and MOE models offer faster prediction than MILP but may compromise solution quality. DRL's key advantage is its ability to adapt to changing and complex environments. Unlike MILP, DRL agents can continuously learn and improve through

interaction, making them suitable for real-world applications with dynamic and uncertain conditions.

To further enhance the proposed model, we suggest incorporating a wider variety of demand data formats and increasing the number of expert agents. This would allow the model to handle the diverse demand patterns in real-world scenarios more effectively. Additionally, we propose incorporating other DRL models, such as Adaptive Reinforcement Learning, into the experimentation and comparing their respective capabilities to refine the model's potential further.

References

[1] H. Golmohamadi, "Demand-side management in industrial sector: A review of heavy industries," *Renewable and Sustainable Energy Reviews*, vol. 156, p. 111963, 2022/03/01/ 2022, doi: https://doi.org/10.1016/j.rser.2021.111963.

[2] K. V. Ramani and B. H. Dholakia, "A decision support system to forecast cement demand," *Information Technology for Development*, vol. 8, no. 3, pp. 169-182, 1999/01/01 1999, doi: 10.1080/02681102.1999.9525305.

[3] F. Kulmer, M. Wolf, and C. Ramsauer, "Medium-term Capacity Management through Reinforcement Learning – Literature review and concept for an industrial pilot-application," *Procedia CIRP*, vol. 107, pp. 1065-1070, 2022/01/01/ 2022, doi: https://doi.org/10.1016/j.procir.2022.05.109.

[4] P. M. Castro, I. Harjunkoski, and I. E. Grossmann, "New Continuous-Time Scheduling Formulation for Continuous Plants under Variable Electricity Cost," *Industrial & Engineering Chemistry Research*, vol. 48, pp. 6701–6714, 2009, doi: https://doi.org/10.1021/ie900073k. ACS Publications.

[5] M. Erdirik-Dogan and I. E. Grossmann, "Simultaneous planning and scheduling of single-stage multi-product continuous plants with parallel lines," *Computers and Chemical Engineering*, vol. 32, pp. 2664–2683, 2008.

[6] Y. Nie, L. T. Biegler, C. M. Villa, and J. M. Wassick, "Discrete Time Formulation for the Integration of Scheduling and Dynamic Optimization," *Industrial and Engineering Chemistry Research*, vol. 54, no. 16, pp. 4303–4315, 2015.

[7] A. Obermeier, C. Windmeier, E. Esche, and J.-U. Repke, "A discrete-time scheduling model for power-intensive processes taking fatigue of equipment into consideration," *Chemical Engineering Science*, 2018.

[8] I. BACH, G. BOCEWICZ, and Z. BANASZAK, "Coonstraint Programming Approach to Multi-product Scheduling," vol. 3, ed: Applied Computer Science, 2007.

[9] C. A. Floudas and X. Lin, "Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review," *Computers and Chemical Engineering*, vol. 28, pp. 2109–2129, 2004.

[10] H. Mokhtari, I. N. K. Abadi, and S. H. Zegordi, "Production capacity planning and scheduling in a no-wait environment with controllable processing times: An integrated modeling approach," *Expert Systems with Applications*, vol. 38, no. 10, pp. 12630-12642, 2011/09/15/ 2011, doi: https://doi.org/10.1016/j.eswa.2011.04.051.

[11] Y. Ma, F. Qiao, and J. Lu, "Learning-based dynamic scheduling of semiconductor manufacturing system.," presented at the IEEE International Conference on Automation Science and Engineering (CASE 2016), 2016.

[12] H. Seidgar, M. Zandieh, and I. Mahdavi, "Bi-objective optimization for integrating production and preventive maintenance scheduling in two-stage assembly flow shop problem," *Journal of Industrial and Production Engineering*, 2016.

[13] X. Yao, N. Almatooq, R. G. Askin, and G. Gruber, "Capacity planning and production scheduling integration: improving operational efficiency via detailed modelling," *International Journal of Production Research*, vol. 60, no. 24, pp. 7239-7261, 2022/12/17 2022, doi: 10.1080/00207543.2022.2028031.

[14] L. Lin and M. Gen, "Hybrid evolutionary optimisation with learning for production scheduling: state-of-the-art survey on algorithms and applications," *International Journal of Production Research*, 2018, doi: 10.1080/00207543.2018.1437288.

[15] M. Larios-gómez, P. M. Quintero-flores, M. Anzures-garcía, and M. Camacho-hernandez, "Application of real-time fan scheduling in exploration-exploitation to optimize minimum function objectives," *Applied Computer Science*, no. *19*(2), pp. 43-54, 2023, doi: https://doi.org/10.35784/acs-2023-13.

[16] S. L. Takeda-Berger, E. M. Frazzon, E. Broda, and a. M. Freitag, "Machine Learning in Production Scheduling: An Overview of the Academic Literature," in *Dynamics in Logistics Proceedings of the 7th International Conference LDIC 2020*, Bremen, Germany, 2020: Springer Nature Switzerland AG, p. 409.

[17] C. D. Hubbs, C. Li, N. V. Sahinidis, I. E. Grossmann, and J. M. Wassick, "A Deep Reinforcement Learning Approach for Chemical Production Scheduling," *An International Journal of Computers Applications in Chemical Engineering*, 2020.

[18] Max Mowbray, Dongda Zhang, and E. A. D. R. Chanona, "Distributional Reinforcement Learning for Scheduling of Chemical Production Processes," *arXiv:2203.00636*, 2022.

[19] D. Weichert, P. Link, A. Stoll, S. Rüping, S. Ihlenfeldt, and S. Wrobel, "A review of machine learning for the optimization of production processes," *The International Journal of Advanced Manufacturing Technology*, vol. 104, no. 5, pp. 1889-1902, 2019/10/01 2019, doi: 10.1007/s00170-019-03988-5.

[20] D. Kalita. "An Overview and Applications of Artificial Neural Networks." Data Science Blogathon. https://www.analyticsvidhya.com (accessed.

[21] H. Rummukainen and J. K. Nurminen, "Practical Reinforcement Learning Experiences in Lot Scheduling Application," presented at the International Federation of Automatic Control (IFAC) Conference 2019, 2019.

[22] T. Quah, D. Machalek, and K. M. Powell, "Comparing Reinforcement Learning Methods for Real-Time Optimization of a Chemical Process," *MDPI*, vol. 8, p. 1497, 2020, doi: doi:10.3390/pr8111497.

[23] S. Windmuller, "Smart capacity planning: A machine learning approach," Master of Science, Department of Information Systems, Production and Logistics Management, University of Innsbruck, Innsbruck

, Austria, 2020.

[24] A. Srinivasan, "Reinforcement Learning: Advancements, Limitations, and Real-world Applications," *International Journal of Scientific Research in Engineering and*

Management (IJSREM), vol. 07, 08, 2023, doi: 10.55041/IJSREM25118.

[25] Y. Yang, H. Lv, and N. Chen, "A Survey on ensemble learning under the era of deep learning," *Artificial Intelligence Review*, vol. 56, no. 6, pp. 5545-5589, 2023/06/01 2023, doi: 10.1007/s10462-022-10283-5.

[26] C.-F. Chien, C.-C. Ku, and Y.-Y. Lu, "Ensemble learning for demand forecast of After-Market spare parts to empower data-driven value chain and an empirical study," *Computers & Industrial Engineering*, vol. 185, p. 109670, 2023/11/01/ 2023, doi: https://doi.org/10.1016/j.cie.2023.109670.

[27] W. Niu *et al.*, "An ensemble transfer learning strategy for production prediction of shale gas wells," *Energy*, vol. 275, p. 127443, 2023/07/15/ 2023, doi: https://doi.org/10.1016/j.energy.2023.127443.

[28] J. Sun, M.-y. Jia, and H. Li, "AdaBoost ensemble for financial distress prediction: An empirical comparison with data from Chinese listed companies," *Expert Systems with Applications*, vol. 38, no. 8, pp. 9305-9312, 2011/08/01/ 2011, doi: https://doi.org/10.1016/j.eswa.2011.01.042.

[29] G. Haixiang, L. Yijing, L. Yanan, L. Xiao, and L. Jinling, "BPSO-Adaboost-KNN ensemble learning algorithm for multi-class imbalanced data classification," *Engineering Applications of Artificial Intelligence*, vol. 49, pp. 176-193, 2016/03/01/ 2016, doi: https://doi.org/10.1016/j.engappai.2015.09.011.

[30] R. Guo, D. Fu, and G. Sollazzo, "An ensemble learning model for asphalt pavement performance prediction based on gradient boosting decision tree," *International Journal of Pavement Engineering*, vol. 23, no. 10, pp. 3633-3646, 2022/08/24 2022, doi: 10.1080/10298436.2021.1910825.

[31] R. Natras, B. Soja, and M. Schmidt, "Ensemble Machine Learning of Random Forest, AdaBoost and XGBoost for Vertical Total Electron Content Forecasting," *Remote Sensing*, vol. 14, no. 15, doi: 10.3390/rs14153547.

[32] M. Pérez-Ortiz, P. A. Gutiérrez, P. Tino, C. Casanova-Mateo, and S. Salcedo-Sanz, "A mixture of experts model for predicting persistent weather patterns," in *2018 International Joint Conference on Neural Networks (IJCNN)*, 8-13 July 2018 2018, pp. 1-8, doi: 10.1109/IJCNN.2018.8489179.

[33] R. Rambabu, P. Vadakkepat, K. C. Tan, and M. Jiang, "A Mixture-of-Experts Prediction Framework for Evolutionary Dynamic Multiobjective Optimization," *IEEE Transactions on Cybernetics*, vol. 50, no. 12, pp. 5099-5112, 2020, doi: 10.1109/TCYB.2019.2909806.

[34] S. E. Yuksel, J. N. Wilson, and P. D. Gader, "Twenty Years of Mixture of Experts," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1177-1193, 2012, doi: 10.1109/TNNLS.2012.2200299.

[35] G. Cheng, L. Dong, W. Cai, and C. Sun, "Multi-Task Reinforcement Learning With Attention-Based Mixture of Experts," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3812-3819, 2023, doi: 10.1109/LRA.2023.3271445.

[36] S. Zhao, I. E. Grossmann, and L. Tang, "Integrated scheduling of rolling sector in steel production with consideration of energy consumption under time-of-use electricity prices," *Computers and Chemical Engineering*, vol. 111, pp. 55–65, 2018.

[37] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[38] E. Bilgin, *Mastering Reinforcement Learning with Python*. Packt Publishing Ltd., 2020.

[39] M. Lapan, *Deep Reinforcement Learning Hands-On - Second Edition*. Packt, 2020, p. 606.

[40] I. C. Gormley and S. Frühwirth-Schnatter, *<h1 _ngcontent-product-detail-page-c37="" class="product-banner-title">Mixture of Experts Models*, 1st Edition ed. Taylor & Francis, 2018.

[41] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Illustrated edition ed. The MIT Press, 2016.

[42] M. Andrychowicz *et al.*, "What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study," *arXiv:2006.05990v1*, 2020.

[43] A. B. W. Putra, R. Malani, B. Suprapty, and A. F. O. Gaffar, "A Deep Auto Encoder Semi Convolution Neural Network for Yearly Rainfall Prediction," in *2020 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, 22-23 July 2020 2020, pp. 205-210, doi: 10.1109/ISITIA49792.2020.9163775.

[44] S. Mitchell, M. OSullivan, and I. Dunning, "PuLP: a linear programming toolkit for python," *The University of Auckland, Auckland, New Zealand*, vol. 65, 2011.

[45] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[46] G. V. Rossum and F. L. Drake, *Python 3 Reference Manual*. CreateSpace, 2009.