Sustainable Material Cutting Optimization Using Deep Q-Networks: A Reinforcement Learning Approach for Resource Efficiency

Linxuan Chen*

The University of Edinburgh, School of Informatics, Scotland

Abstract. This paper proposes an innovative approach to the Cutting Stock Problem (CSP) by integrating Graph Neural Networks (GNN) which effectively extract and process graph-structured data and Deep Reinforcement Learning (DRL) which utilizes the data generated by the GNN model to make sequential cutting decisions. The GNN model is embedded with Graph Convolutional Networks (GCN) layers, while the DRL model is structured with Deep Q-network (DQN). In my own study using KTH-TIPS dataset for model training, I have achieved promising experimental outcomes, decreasing loss functions and stabilizing total rewards, which demonstrates the model's ability to progressively decrease prediction errors in stages and reach the best cutting patterns.

1 Introduction

The cutting stock problem deals with questions related to optimal production which includes cutting available stock lengths into specified amounts of materials needed for completing the orders. It has been researched for a long time because of its positive impact to resource usage and saving of costs within manufacturing industry [1]. More recently, with certain companies going the extra mile in promoting more sustainable development, there has been a higher demand in the need for having a better way of solving this problem currently [2]. Historically, the solution of the cutting stock problem has been based on linear programming and heuristic algorithms [3]. There are cases in real life where stock sheets of multiple pieces may be in contact with each other, separated or overlapped. In this case, previous methods are often problematic in high dimensional spaces, this could greatly affect the effectiveness and optimization of the solutions [4]. Therefore, there is still limited success in the in the development of the solution to enhance efficiency against the time taken during its implementation in the statistics relative to the current industrial applications [5].

This research purposes an innovative approach to the CSP by integrating two advanced artificial techniques: Deep Reinforcement Learning and Graph Neural Network. DRL is a machine learning technique that, on the one hand, trains a Depp Q-network agent to make decisions by interaction with the environment and feedback in the form of rewards or penalties [6]. GNN, on the other hand, are a class of deep learning models designed to capture

* Corresponding author: s2231967@ed.ac.uk

the structural information in graph-structured data [7]. My research framework consists of two main components. One is a GNN agent that captures the intricate connections between different components of a cutting plan utilizing the KTH-TIPS datasets, which involves textures under varying illumination, pose, and scale to provide a rich informative representation that can enhance the decision-making capabilities of the DRL algorithm in the next stage [8]. The other one is a DRL agent that utilizes the Deep Q-Network to learn optimal strategies through interaction with complex environments [9]. My innovative approach aims to overcome the limitation of conventional methods and offer a more efficient and flexible solution to the CSP by combining these two advanced AI technologies.

This paper demonstrates how DRL and GNN would be integrated together in my proposed framework, including the design of the research and the training process of the system. Next, a description of the model's development process is given, along with the results of the experimental study, which highlight the model's effectiveness and suggest areas for future development. Finally, this paper discusses future directions for improvement, including applying my method to other domains, experimenting with different model designs, and utilizing real-world data.

2 Methodology

2.1 Dataset Description and Preprocessing

In this experiment, the KTH-TIPS dataset, which is widely used for texture classification and recognition [8], is utilized. This database consists of 10 texture types such as sandpaper, aluminum foil, rubber, sponge and much more with 81 pictures per each type out of 810 images total. In this dataset, which consists of colored images, every picture has the parameters 200 pixels × 200 pixels. Within the feature extraction and graph construction pipelines there exist several essential image preprocessing steps that are necessary for the later experimental procedures. Initially, the images are loaded from the corresponding paths and are turned into a gray scale image. All images are subsequently clipped and resized to 128×128 pixels. A standardized image size of 128×128 pixels is then imposed through resizing operations. Considering the fact that the deep learning framework is based on the images of the library, the processed pictures are converted into the format of Pytorch's Tensor. For these purposes images height and width is represented as graph node. In order to extract the texture information, local binary patterns are utilized and the output of this operation is encoded as input feature using histogram representation. The last stage involves flattening of the grayscale images into one-dimension arrays to make it easier in processing.

2.2 Graph Neural Networks Architecture

Graph neural networks are a specialized class of neural networks designed to process and analyse graph-structured data, which consists of nodes connected by edges [10]. GNNs prove to be quite efficient in considering both local and global aspects of this structure by handling the connection between a node and its surrounding neighbors, thus being capable of taking advantage of the structural information in graph datasets. As in the earlier case, to optimize the cutting strategy in CSP, parts of the cutting regions can bring about a GNN approach, specifically the graph convolutional network (GCN) [11]. As part of the GNN component, GCNs are employed, consisting of three layers within the governing structure. In developing the notion of GCN layer within a hybrid structure, it is common to come across the following equation:

$$H^{(k+1)} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(k)}W^{(k)}) \quad (1)$$

where $H^{(k+1)} \in \mathbb{R}^{(NxD)}$ represents the node feature matrix at layer k+1, $\hat{A}$ represents the adjacency matrix with self-connections ($\hat{A} = A + I_N$), D represents the degree matrix of $\hat{A}$, $W^{(k)}$ represents the trainable weight matrix, and σ is the ReLU activation function.

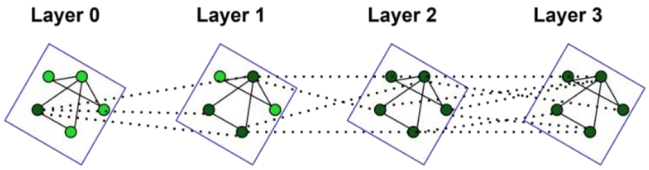

Fig. 1. Schematic of graph convolutional layer.

Figure 1 presents the visualization of GCN layers. Layer 1, the first layer, performs the initial graph convolution to generate the first set of hidden features from the input node features. Layer 2, the second layer, further processes these hidden features through a second convolutional layer to create more refined node representations. Finally, Layer 3, the third layer, applies another graph convolution to produce the third set of hidden features, which encapsulate higher-level structural information critical for optimizing the cutting strategy.

Additionally, the model employs a Jumping Knowledge Layer to adaptively combine node representations from multiple layers:

$$h_i^{(final)} = AGGREGATE(\{h_i^{(l)}: l = 0, 1, 2, 3\}) \quad (2)$$

where $AGGREGATE$ is a learnable function (in our case, a concatenation followed by a linear transformation). Subsequently, node-level features are aggregated into a graph-level representation via global mean pooling. Finally, the model utilized a Fully Connected Layer for final graph-level feature aggregation and classification output. The flowchart outlining this process is provided below. Jumping Knowledge Layer and Global Pooling Flowchart are shown in the figure 2 below.

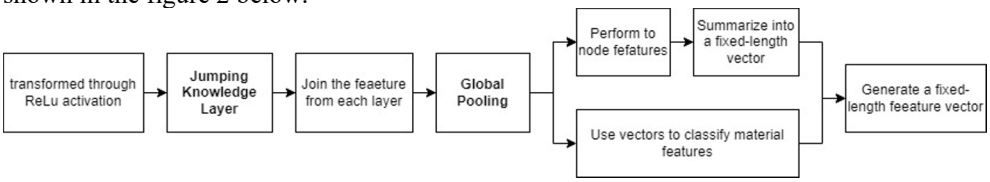

Fig. 2. Jumping Knowledge Layer and Global Pooling Flowchart.

2.2 Deep Q-Network Architecture

The Deep Q-Network component of my model is structured to learn optimal cutting strategies through iterative interaction with a simulated cutting stock environment. It is fundamentally based on the framework introduced by Mnih [6], with specific modifications implemented to accommodate the graph-structured input generated by the GNN component.

In the DQN component, the state representation s_t at time t is a composite structure, defined as:

$$s_t = [v_G, f_C] \quad (3)$$

where $v_G \in \mathbb{R}^d$ represents the graph-level output vector produced by the GNN component, encapsulating the structural and textural information of the current stock material, and $f_C \in \mathbb{R}^m$ is a vector of additional features describing the current cutting state (Dai et al., 2018). This vector is augmented with additional features that describe the current cutting state, such as the remaining stock area, the number of cuts already made, and the dimensions of the current piece under consideration. This rich state representation allows the DQN to

make informed decisions depending on both the material characteristics and the current progress of the cutting process [12].

The action space A is defined as a set of discrete cutting decisions, each defined by its position on the stock material and its orientation, expressed as:

$$A = \{(x, y, \theta) \mid x \in X, y \in Y, \theta \in \Theta\} \quad (4)$$

where X and Y are sets of possible x and y coordinates for the cut, and $\theta = \{0°, 90°\}$ represents the possible orientations (horizontal or vertical). The cardinality of A, denoted as $|A|$, is set to n = 100 in my implementation, providing a balance between precision and computational feasibility [13].

The DQN is responsible for estimating the action-value function: $Q(s, a; \theta)$, where θ denotes the learnable parameters of the network. The architecture of the Q-network is structured as a multi-layer perceptron (MLP), consisting of the three main layers. First is the input layer, where the number of units equals to function $\dim(v_G) + \dim(f_C)$, which represents the combined dimensions of the graph-level and cutting state features. Second is the hidden layer, where there are two fully connected layers with 128 and 64 units respectively, each followed by a Rectified Linear Unit (ReLU) activation function. The last one is the output layer, with $|A|$ units, representing the Q-values associated with each action in the action space. The Q-values for a given state s are computed as the following formula:

$$Q(s, \cdot; \theta) = W_3 \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot s + b_1) + b_2) + b_3 \quad (5)$$

where W_i and b_i represent the weight matrices and bias vectors for the respective layers.

To stabilize the learning process and mitigate the problem of correlated updates, we employ an experience replay mechanism. A replay memory D with a capacity of N = 10,000 transitions is maintained, where each transition is a tuple (s_t, a_t, r_t, s_{t+1}) representing the current state, action taken, reward received, and the resulting next state. During training, mini-batches of size 32 are uniformly sampled from D to update the Q-network, thereby breaking the temporal correlations in the training data that can otherwise hinder convergence. The Q-network is trained to minimize the loss function:

$$L(\theta) = E_{(s,a,r,s')\sim D}\left[\left(r + \gamma \cdot \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right] \quad (6)$$

where γ is the discount factor (set to 0.99), and θ^- represents the parameter of a target network. To address the moving target problem which is common in Q-learning, a target network $Q(s', a';\ \theta')$ is employed, with parameters θ'. This target network shares the same architecture as the primary Q-network but is updated less frequently. Specifically, θ' is synchronized with θ every 1,000 steps. This approach stabilizes the Q-value updates, thereby enhancing the overall stability of the learning process.

In summary, the DQN learns by interacting with a simulated environment. These decisions are informed by a combination of graph-structured data from the previous GNN model and additional features detailing the current cutting state. Furthermore, DQN's learning process is enhanced by an experience replay mechanism, which stores past transitions in a replay buffer and randomly samples mini batches during training. Afterwards, to mitigate the instability caused by rapid changes in the Q-values, the experiment employed a target network. It is updated periodically, allowing the DQN to progressively converge towards an optimal policy for maximizing material utilization and minimizing waste.

3 Results and Analysis

3.1 Results Display

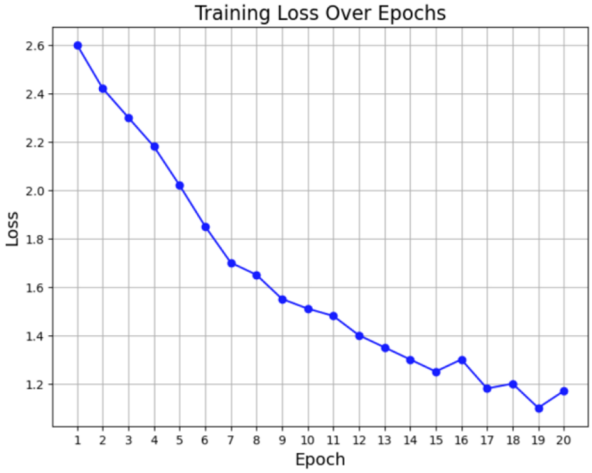

Fig 3. Loss Curve of the GNN model on training set.

Figure 3 shows the loss change of the GNN model on the training set. The monotonic decrease in loss across epochs indicates effective learning and gradual optimization of the model's fit to the training data. This trend suggests that the GNN component is successfully capturing the structural and textural features of the input data.

Fig 4. Loss Curve of the GNN model on test set.

Figure 4 shows the loss change of the GNN model on the test set. The decreasing trend of the model's loss on the test set mirrors that of the training set, suggesting that the model can maintain strong generalization ability when dealing with the unseen data and gradually converge to a stable error level.

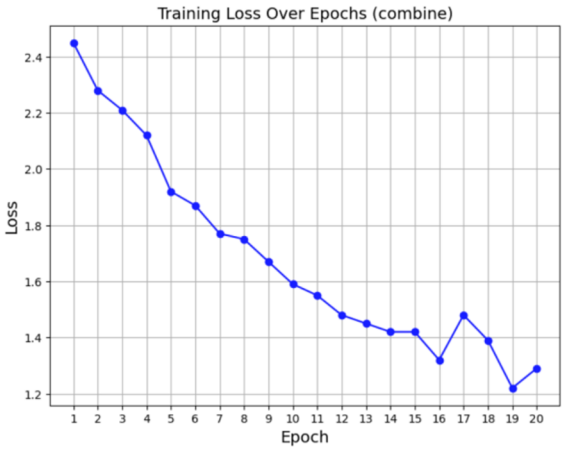

Fig 5. Combined Loss Curve of GNN and DRL.

Figure 5 illustrates the comprehensive loss function changes of the combined GNN and DQN model. The overall downward trend in the loss function demonstrates the model's ability to progressively reduce prediction errors. Notably, the rapid decline in loss during the initial iterations reflects the model's efficient adaptation to the data distribution and effective weight adjustments.

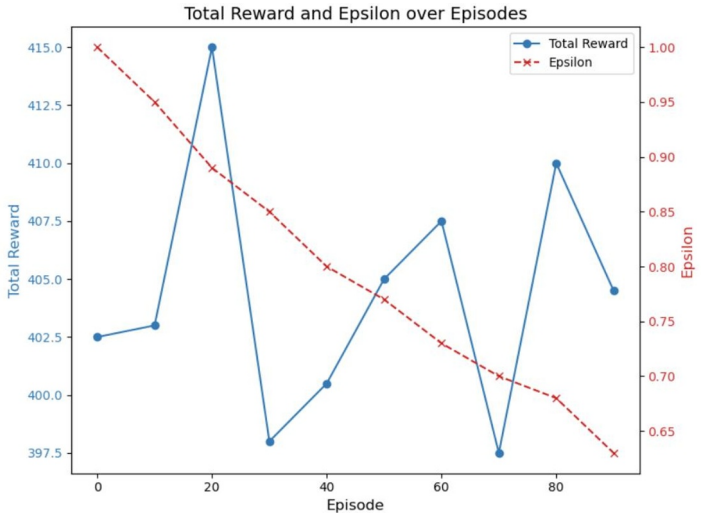

Fig 6. Combined Total Reward of the GNN and DRL.

Figure 6 presents the total reward and Epsilon value changes over the training rounds. As the training progresses, the Epsilon value gradually decreases, indicating a transition from exploration to exploitation in the model's learning strategy. Concurrently, the total reward, despite fluctuations, maintains a high overall level and stabilizes in the later stages of training. This stabilization suggests the model's convergence towards an optimal strategy for handling diverse cutting tasks while maximizing material utilization.

3.2 Analysis and Discussion

The performance of the GNN model was evaluated by training and testing on the KTH-TIPS dataset. The consistent performance of the GNN model on both training and test sets, as

evidenced by the similar loss trends, underscores its strong generalization capabilities. This consistency is crucial for the model's practical applicability, as it suggests reliable performance on unseen data in real-world scenarios.

The decreasing trend of the comprehensive loss function indicates the model's refinement in feature representation and prediction accuracy. The initial rapid decline in loss demonstrates the model's ability to quickly adapt to the data distribution, while the subsequent steady decline and convergence highlight the model's stability and robustness. As the Epsilon value gradually decreases, the model reduces randomness exploration and relies more on the learned effective strategies. Therefore, the fluctuation of the total reward mainly reflects the tentative optimization of the model in the process of strategy adjustment. At the later stage of training, the total reward stabilized, indicating that the model had moved toward an optimal strategy that could effectively cope with different cutting tasks, maximize material utilization and reduce waste.

GNN captures important characteristic information in the cutting process through a structured representation of the material, which provides a more accurate state input for DRL. As expected by the theoretical framework, the rich structural information provided by GNN enables DRL to make more accurate decisions when evaluating different cutting strategies, thus improving the overall optimization effect. The epsilon value's gradual decrease, coupled with the stabilization of total reward, illustrates the model's effective balance between exploration and exploitation. This balance is critical in reinforcement learning contexts, allowing the model to discover and refine optimal strategies while avoiding local optima.

In a word, the results, aligning well with my theoretical framework, confirms that the rich structural information provided by the GNN enhances the DRL's decision-making capabilities. My experimental outcomes demonstrate the significant potential of the GNN-enhanced DQN model in addressing complex industrial optimization problems. The model's ability to improve material utilization rates, reduce waste, and adapt to varying cutting tasks and external constraints positions it as a promising solution for enhancing resource efficiency and cost control in manufacturing processes.

4 Conclusion and Future Work

The current research has solved the CSP in a highly innovative way by integrating GNNs and DRL. The GNN component effectively captured the structural and textural information of the cutting materials, as evidenced by the consistent performance on both training and test sets of the KTH-TIPS dataset. The DQN, enhanced by the rich representational output of the GNN, showed a stable learning trajectory, with decreasing loss and increasing total rewards over the training period. Finally, the integrated GNN-DQN model demonstrated an ability to balance exploration and exploitation, converging towards optimal cutting strategies that maximize material utilization. These results suggest that our approach can potentially overcome limitations of traditional CSP solutions, offering a more adaptable and efficient method for resource optimization in manufacturing processes. In a word, the model's ability to learn from complex, graph-structured data and make sequential decisions aligns well with the dynamic nature of real-world cutting stock scenarios.

The implications of this research extend beyond the immediate context of material cutting. It contributes to the broader field of AI-driven optimization in manufacturing industry, which highlights the potential of hybrid models that leverage the structural learning capabilities of GNNs and the sequential decision-making strength of DRL.

Although the results are promising, it is also significant to acknowledge the limitations of this study. The model's performance has been validated on the KTH-TIPS dataset, which, while diverse, may not fully represent the complexity encountered in real-world industrial

cutting process. Future work should focus on testing the model with industry-specific datasets and in real-world manufacturing environments.

Additionally, further research could explore the model's sensitivity to hyperparameter choices and investigate the impact of different GNN architectures on the overall performance. Comparative studies with other optimization techniques would also provide valuable insights into the relative strengths and weaknesses of this model.

References

1. P. C. Gilmore, R. E. Gomory, A linear programming approach to the cutting-stock problem. Operations research, **9**, 849 (1961)
2. K. Grzybowska, A. Awasthi, Literature review on sustainable logistics and sustainable production for Industry 4.0. Sustainable Logistics and Production in Industry 4.0: New Opportunities and Challenges, **1**,18 (2020)
3. H. Dyckhoff, A typology of cutting and packing problems. European journal of operational research, **44**, 145 (1990)
4. J. A. Bennell, J. F. Oliveira, The geometry of nesting problems: A tutorial. European journal of operational research, **184**, 397 (2008)
5. A. C.Cherri, M. N. Arenales, H. H. Yanasse, K. C. Poldi, A. C. G. Vianna, The one-dimensional cutting stock problem with usable leftovers–A survey. European Journal of Operational Research, **236**, 395 (2014)
6. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, D.Hassabis, Human-level control through deep reinforcement learning. Nature,**518,** 529, (2015)
7. Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S. Y. Philip, A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems,**32**, 4 (2020)
8. E. Hayman, B. Caputo, M. Fritz, J. O. Eklundh, On the significance of real-world conditions for material classification. In Computer Vision-ECCV 2004: 8th European Conference on Computer Vision, Prague, Czech Republic, Springer Berlin Heidelberg, May 11-14, May 11-14 (2004), pp. 253-266
9. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, D. Hassabis, Mastering the game of Go with deep neural networks and tree search. nature, **529**, 484 (2016)
10. F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model. IEEE Transactions on Neural Networks, **20**, 61-80. (2009)
11. T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks. International Conference on Learning Representations (ICLR), (2017) 1-14
12. I. Bello, H. Pham, Q. V. Le, M. Norouzi, S. Bengio, Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:**1611**.09940 (2016)
13. G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, B. Coppin, Deep reinforcement learning in large discrete action spaces， arXiv preprint arXiv:1512.07679 (2015)