Dual-Agent Reinforcement Learning for Automated Feature Generation

Wanfu Gao[1,2], **Zengyao Man**[1,2], **Hanlin Pan**[1,2*] and **Kunpeng Liu**[3]
[1]College of Computer Science and Technology, Jilin University, China
[2]Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, China
[3]Department of Computer Science, Portland State University, Portland, OR 97201 USA
gaowf@jlu.edu.cn, manzy23@mails.jlu.edu.cn, panhl23@mails.jlu.edu.cn, kunpeng@pdx.edu

Abstract

Feature generation involves creating new features from raw data to capture complex relationships among the original features, improving model robustness and machine learning performance. Current methods using reinforcement learning for feature generation have made feature exploration more flexible and efficient. However, several challenges remain: first, during feature expansion, a large number of redundant features are generated. When removing them, current methods only retain the best features each round, neglecting those that perform poorly initially but could improve later. Second, the state representation used by current methods fails to fully capture complex feature relationships. Third, there are significant differences between discrete and continuous features in tabular data, requiring different operations for each type. To address these challenges, we propose a novel dual-agent reinforcement learning method for feature generation. Two agents are designed: the first generates new features, and the second determines whether they should be preserved. A self-attention mechanism enhances state representation, and diverse operations distinguish interactions between discrete and continuous features. The experimental results on multiple datasets demonstrate that the proposed method is effective.

1 Introduction

Feature generation in machine learning is the process of combining original data features to create new ones. These new features can capture complex relationships between original features, enhance model robustness, improve data representation, and ultimately improve the performance of machine learning tasks. As Figure 1 shows, there are several features: gender, weight and height, and our goal is to predict whether a person is healthy based on these features. Feature generation methods can help create a new feature $\frac{\text{weight}}{\text{height}^2}$, known as the Body Mass Index (BMI) [Obese, 1998], which

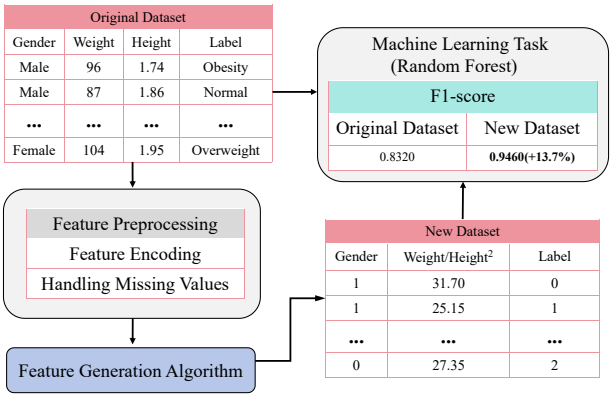


Figure 1: After data preprocessing, the dataset is transformed using a feature generation algorithm, resulting in a new dataset that significantly improved the *F1-score* in downstream machine learning tasks compared to the original dataset.

can lead to better predictive performance. However, manual feature generation by domain experts is labor-intensive and does not accurately capture the relationships among a large number of features. Therefore, automated methods for feature generation are naturally needed. Some traditional feature generation methods, such as FCTree [Fan *et al.*, 2010] and FICUS [Markovitch and Rosenstein, 2002], are still influenced by domain knowledge, lacking flexibility and adaptability. Current methods using reinforcement learning (RL) for feature generation have made the exploration of features more flexible and efficient [Khurana *et al.*, 2018; Wang *et al.*, 2022]. Despite these methods utilizing reinforcement learning for feature generation showing potential improvement, several challenges still exist. First, in the feature expansion phase of reinforcement learning, a large number of redundant features are generated that need to be removed. Current RL-based feature generation methods typically involve feature selection based on mutual information after generating features, selecting the top K features with the most information for subsequent feature generation. However, this method only considers the best features in each round, neglecting features that initially perform poorly but contribute better performance in later rounds. Second, RL

*Corresponding author

agents must take states as input and current state representation methods primarily focus on autoencoders and graph convolutional networks [Xiao *et al.*, 2022], but these methods fail to sufficiently capture the complex relationships between features. Third, there are significant differences between discrete and continuous features, and it requires different operations for different types of features. For example, mathematical operations like logarithmic or exponential operations are only suitable for continuous features while cross-product is only suitable for discrete features. Most existing methods ignore interactions between discrete and continuous features.

To deal with these challenges, in this paper, we propose a Dual-Agent Reinforcement Learning (DARL) method for automated feature generation. **To deal with the first challenge,** the concept of hierarchical reinforcement learning [Morimoto and Doya, 2001; Kulkarni *et al.*, 2016; Zhang *et al.*, 2023] is used to decompose complex tasks into smaller, simpler sub-tasks. Specifically, we designed two agents for the reinforcement learning framework: the first agent is responsible for feature generation, and the second agent is responsible for feature discrimination. Feature discrimination is used to determine whether the generated features are worth retaining. By integrating the feature discrimination process into the reinforcement learning framework, an optimal subset of features is obtained that performs well in the current decision step and offers better long-term rewards, as reinforcement learning aims to maximize cumulative rewards. The method effectively explores the feature space to generate a superior set of features. Compared to traditional mutual information methods, this method achieves better results. **To deal with the second challenge,** the representation of states in RL is enhanced using the Transformer model [Vaswani *et al.*, 2017]. Through self-attention mechanisms, the state representation in RL can reveal correlations between each feature and provide rich information for RL algorithms to better understand complex feature relationships. **As for the third challenge,** we use different arithmetic operations to capture three types of relationships, i.e., the relationship between discrete and discrete features, the relationship between continuous and continuous features, as well as the relationship between discrete and continuous features.

In summary, this paper presents a novel dual-agent reinforcement learning feature generation method. Through the dual-agent reinforcement learning process, features are generated with better performance and higher interpretability. Our main contributions include:

- The method Dual-Agent Reinforcement Learning (DARL) is used for feature generation, where the first agent is responsible for generating features, and the second agent is responsible for preserving useful features and removing redundant ones.
- We propose to use the self-attention mechanism for a reinforcement learning state, which can lead to better embedding representations.
- We propose to distinguish between discrete and continuous feature interactions, which enables the generation of more interpretable features.

2 Related Work

The goal of automated feature generation is to identify the optimal feature set to enhance the performance of machine learning models [Chen *et al.*, 2021]. Feature generation methods include expansion-reduction methods and search-based methods. Expansion-reduction methods first expand the feature space and then perform feature selection to reduce redundancy. Search-based methods are a class of techniques used in automated feature generation that explore potential feature transformations and combinations using search methods to find the optimal feature set.

Expansion-reduction methods such as ExploreKit [Katz *et al.*, 2016] include three stages in its workflow: candidate feature generation, candidate feature ranking, and candidate feature evaluation and selection. It executes all transformation functions on the complete dataset and selects the subset to be added based on the empirical performance of models trained with the candidate features. FEADIS [Dor and Reich, 2012]: it randomly selects original features and mathematical functions to generate new features. Autofeat [Horn *et al.*, 2020]: it first generates a large number of nonlinear features and then selects a small subset of meaningful features from them to create a new dataset. It is evident that expansion-reduction methods struggle with the problem of feature space explosion. Currently, a more effective method is to use search methods to explore potential feature transformations in order to identify the most valuable features for predictive models.

In the field of search methods, GRFG [Wang *et al.*, 2022] is based on group-wise feature transformations. Through reinforcement learning, the system selects an operator for each group and then performs operations between the groups. TransGraph [Khurana *et al.*, 2018] generates high-order features using transformation graphs and Q-learning algorithms. The Neural Feature Search (NFS) framework [Chen *et al.*, 2019] assigns a recursive neural network controller [Lipton *et al.*, 2015] to each feature. Each feature generates a new feature independently. However, this method requires *N* RNN controllers for *N* features. Bigfeat [Eldeeb *et al.*, 2022] proposed a dynamic feature generation technique based on computing tree logic. These methods show promising results in automating feature generation and improving machine learning performance, but they generate excessive redundant features. While there are methods to improve relevance and reduce redundancy [Zhang *et al.*, 2021; Zhang and Gao, 2021; Li *et al.*, 2024], they have not been integrated into existing approaches. Additionally, the embedded states do not adequately represent the relationships between features. At the same time, they do not distinguish between discrete and continuous features. Our proposed method aims to address these limitations by integrating feature discrimination into the search process and enhancing the representation of feature relationships.

3 The Proposed Method

3.1 Problem Settings

Given a dataset $\mathcal{D} = \{\mathcal{F}, y\}$, where $\mathcal{F}$ represents the original feature set and y represents the target label set, the feature set $\mathcal{F} = \{D_1, \ldots, D_N, C_1, \ldots, C_M\}$, consists of N

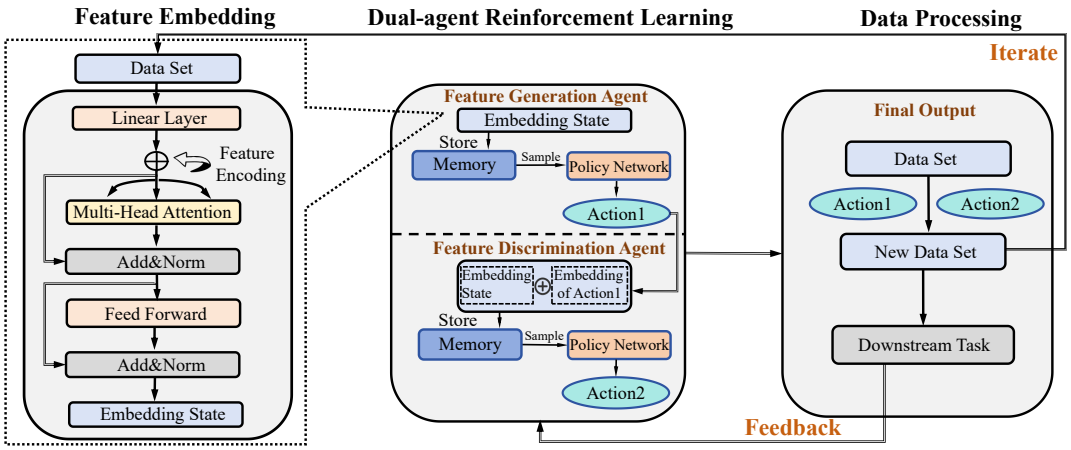


Figure 2: Overview of DARL. The dataset is transformed into feature embedding representations through a self-attention mechanism. Subsequently, a feature generation agent produces a sequence of operators, and a feature discrimination agent generates a discriminator sequence. These two sequences are combined with the original feature set to generate a new feature set. The updated feature set is then input into downstream tasks for evaluation, and the results are fed back to the two agents. This process iterates until the best feature set is discovered or the maximum number of iterations is reached.

discrete features $\{D_1, \ldots, D_N\}$ and M continuous features $\{C_1, \ldots, C_M\}$. We aim to find the optimal feature set $\mathcal{F}^*$ that maximizes:

$$\mathcal{F}^* = argmax_{\mathcal{F}}(V_A(\mathcal{F}, y)), \tag{1}$$

where A represents downstream machine learning tasks such as random forest, SVM, etc., and V denotes evaluation metrics like *F1-score*. Therefore, our objective is to identify a feature set that maximizes the performance metric of downstream tasks.

3.2 Overall Framework

In this section, we describe the overall framework idea of our feature generation algorithm. The framework adopts a hierarchical reinforcement learning strategy, which interacts with the environment through two Markov decision processes to achieve better overall effectiveness, as illustrated in Figure 2. Specifically, for a dataset $\mathcal{D}$, after feature embedding, it is input into the feature generation agent. The feature generation agent generates a new feature for each feature in the original feature set using predefined operations. The goal of this stage is to expand the feature set by creatively transforming existing features to increase the information content. Obviously, continuously generating a new feature for each feature would lead to a feature explosion, resulting in an excessive number of irrelevant features. Therefore, another agent is designed to eliminate these irrelevant features. Hence, we propose to design a feature discrimination agent. The feature discrimination agent aims to streamline the feature set by selecting the most useful features. Due to the hierarchical nature of the framework, the output of the feature generation agent serves as the input for the feature discrimination agent. The feature generation agent generates an operator (e.g., "+", "-") for each feature in the feature set $\mathcal{F}$, resulting in an operator sequence $\mathcal{T}_1 = \{p_1, \ldots, p_{M+N}\}$, while the feature discrimination agent generates a discriminator for each operator, resulting in a discriminator sequence $\mathcal{T}_2 = \{q_1, \ldots, q_{M+N}\}$. Combining the two action sequences together, $\mathcal{T} = \{p_1 q_1, \ldots, p_{M+N} q_{M+N}\}$, generates a new action sequence. This new action sequence is then applied to the original feature set $\mathcal{F}$, resulting in a new, optimized feature set $\hat{\mathcal{F}}$. The new feature set $\hat{\mathcal{F}}$ is input into downstream tasks (e.g., random forest, SVM) for feature set evaluation and reward calculation. The evaluation results of downstream tasks are used as feedback rewards for the feature generation agent and the feature discrimination agent to guide them in obtaining better policy networks. To obtain higher-order feature representations, the original feature set $\mathcal{F}$ is updated to the newly generated feature set $\hat{\mathcal{F}}$, and the process is repeated for K rounds until reaching the maximum number of iterations. Unlike traditional feature generation methods, our framework primarily aims to maximize cumulative rewards by treating feature discrimination as a Markov Decision Process (MDP). The framework automates the feature generation process and reduces the reliance on domain expert knowledge. It also considers computational efficiency and manages computational resources by limiting the number of iterations and optimizing algorithms.

3.3 Feature Embedding

Reinforcement learning, as a framework for solving the MDP, requires the description of the MDP state. The feature generation agent must take a state as the input, and this state is supposed to be constructed from the original feature set. Thus, the state representation from the original feature set to an embedding vector is mandatory. This section focuses on introducing the state of the feature generation agent. Previous methods primarily utilized autoencoders, graph convolutional autoencoders, etc., as the embedding representations of states [Xiao *et al.*, 2022]. These methods, with their shorter context lengths and limited ability to capture intricate dependencies, have not effectively learned the underlying relationships between features. Therefore, in this paper, we adopt the idea

from the Transformer model [Vaswani *et al.*, 2017] which has demonstrated excellent performance in embedding tabular data [Zhang *et al.*, 2024; Huang *et al.*, 2020]. It utilizes a multi-head self-attention mechanism to learn more effective embedding representations. Additionally, when processing tabular data, it is important to distinguish between discrete and continuous features. The embedding representation of the state is shown in the feature embedding section of Figure 2. Similar to the architecture of a Transformer, the dataset is initially passed through a linear layer to transform the tabular data into a fixed dimension. Next, feature encoding is performed on the tabular data. Since tabular data does not convey positional information, feature encoding in this context only focuses on the distinctions between discrete and continuous features. The equation for feature encoding is shown as:

$$F_{enc} = \gamma \sin\left(\frac{p_f}{10^{\left(\frac{i}{d_{\text{model}}-1}\right)}}\right), \tag{2}$$

where $p_f = -1$ for discrete features, $p_f = 1$ for continuous features, and $p_f = 0$ for labels. Considering that $sin(\omega t)$ is an odd function and has origin symmetry, it is chosen for feature encoding. γ is a coefficient used to amplify or diminish the influence of feature encoding. Adding positional encoding to feature embedding enables the distinction of different types of features.

After applying the multi-head self-attention mechanism as shown in Formula (3), followed by a residual connection with layer normalization, the output is fed into a feed-forward neural network. After that, another residual connection with layer normalization is applied to obtain the final embedding representation.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{3}$$

By utilizing the self-attention mechanism, the original tabular data can learn the distinct relationships between features. This type of state representation can lead to better results in reinforcement learning tasks.

3.4 Dual-agent Reinforcement Learning

We illustrate the proposed dual-agent reinforcement learning process in Figure 2 in this section. The goal of the reinforcement learning agent is to find the optimal action by utilizing a policy network to maximize cumulative rewards. In a single training iteration, the agent receives the current state and generates actions through the policy network. The environment evaluates the actions to obtain rewards based on the machine learning algorithm *A* and evaluation metric *V*, and updates the policy network. Next, we introduce the states, actions, and rewards for the two agents.

Feature generation agent

Feature generation agent learning system includes *i) state* The state of the feature generation agent is defined as the embedded representation from the original tabular data through the self-attention mechanism, $s_1 = emb(\mathcal{D})$. *ii) action* The action space of the feature generation agent consists of operations on the original features. To distinguish discrete and continuous features, we utilize different action spaces. For continuous features, the action space includes operations such as "absolute value", "none", "square", "inverse", "logarithm", "square root", "cube", "addition", "subtraction", "multiplication" and "division". For discrete features, the action space includes "cross" [Luo *et al.*, 2019] and "add". Feature cross means that if two original features are $f_1 = \{A, B\}$ and $f_2 = \{C, D\}$, then a new feature with four categorical values will be generated: $f_{new} = \{AC, AD, BC, BD\}$. By performing feature cross, the model can learn the combined effects of discrete features. To address the issue of cross-operation between discrete and continuous features, we convert continuous features into discrete features by binning with a decision tree [Ying *et al.*, 2023]. The binning process converts continuous features into discrete features so that they can interact with originally discrete features. This method allows for learning richer feature representations. *iii) reward* The reward is calculated as the difference between the score achieved by the newly generated feature set $Score_{new}$ and the score of the original feature set $Score_{ori}$ in the downstream learning task. It can be calculated by:

$$r_1 = Score_{new} - Score_{ori}. \tag{4}$$

Feature discrimination agent

Feature discrimination agent learning system includes *i) state* Considering the hierarchical relationship between the feature generation agent and the feature discrimination agent, in order to provide a better state representation for the feature discrimination agent, the action sequence generated by the feature generation agent is transformed into word embedding vectors, which are then concatenated with the state of the feature generation agent to form the state of the feature discrimination agent. This hierarchical representation enables the feature discrimination agent to consider the historical behavior of the feature generation agent, which helps capture the collaboration and dependency between agents. This hierarchical information enhances the contextual awareness of the feature discrimination agent's decision-making process, $s_2 = emb(\mathcal{D}) \oplus emb(\mathcal{T}_1)$. *ii) action* The action space of the feature discrimination agent consists of "delete", "replace", and "add". "Delete" means that the newly generated feature should be removed. "Replace" means that the newly generated feature is superior to the original feature and should substitute the original one. "Add" means that the newly generated feature should be added to the original feature set. Through continuous evaluation and optimization, the feature discrimination agent will develop a long-term strategy to determine which features should be deleted, replaced, or added at each step, thereby creating the most valuable feature set. The feature discrimination agent can finely manage the feature set and offer a more refined state representation for the entire system through its decision-making process. The feature discrimination agent enhances the prediction accuracy and efficiency of the entire system. *iii) reward* We design three reward functions r_{del}, r_{rep}, and r_{add} corresponding to the three actions:

As Equation (5) shows, to determine whether the newly generated feature f_{new} should be deleted, the reward is formally represented as the difference between the mutual infor-

mation of the original feature and the label $I(f_{ori}, y)$, and the mutual information of the newly generated feature and the label $I(f_{new}, y)$. A larger difference indicates that the original feature contains more information about the label y, therefore, the newly generated feature is meaningless and should be deleted.

$$r_{del} = I(f_{ori}, y) - I(f_{new}, y). \tag{5}$$

To determine whether the newly generated feature f_{new} should replace the original feature f_{ori}, we design a reward function r_{rep} using mutual information difference as Equation (6) shows. Opposite to r_{del}, a larger difference indicates that the new feature contains more information about the label y, therefore, the newly generated feature is meaningful and should replace the original feature.

$$r_{rep} = I(f_{new}, y) - I(f_{ori}, y). \tag{6}$$

To determine whether the newly generated feature should be added, it is necessary to measure the redundancy between the original feature and the new one. We use mutual information between the original feature and the new feature $I(f_{ori}, f_{new})$ to quantify their redundancy. A smaller mutual information value indicates a weaker correlation between the two features, suggesting lower redundancy. Therefore, the new feature should be added.

$$r_{add} = I(f_{ori}, f_{new}). \tag{7}$$

The utility reward is defined as the difference between the score of the downstream learning task achieved using the newly generated feature set and the score achieved using the original feature set, as shown in Equation (8).

$$r_{imp} = Score_{new} - Score_{ori}. \tag{8}$$

Combining the above equations, we obtain the reward for the feature discrimination agent as follows:

$$r_2 = \alpha * r_{del} + \beta * r_{rep} + \gamma * r_{imp} - \delta * r_{add}, \tag{9}$$

where α, β, γ and δ are positive weights.

Mutual information is utilized as part of the reinforcement learning framework and works in conjunction with downstream task performance. This enables our method to more effectively balance short-term feature relevance and long-term task utility, overcoming the limitations of using mutual information alone for feature selection.

Model Training

We employ Deep Q-Networks (DQN) [Mnih *et al.*, 2013] as the reinforcement learning model. DQN marries deep learning with Q-learning, using a neural network to approximate the Q-function, predicting the expected return of an action in a given state. It balances exploration and exploitation using the ϵ-greedy strategy, where the agent randomly selects actions with a probability of ϵ to explore the environment and with a probability of 1-ϵ, selects the optimal action according to the current policy. Additionally, this method also employs the technique of experience replay, which involves maintaining a replay buffer to store tuples of data (state, action, reward and next state) sampled from the environment. During the training of the Q network, a random sample of data is drawn from the replay buffer for training purposes. Both agents need to minimize the Bellman formula of the action-value function and reduce the temporal difference error:

$$\mathcal{L} = Q(s_t, a_t) - (R(s_t, a_t) + \gamma max_{a_{t+1}} Q(s_{t+1}, a_{t+1})). \tag{10}$$

Here, γ represents the discount factor, and Q is the estimated Q function by the deep neural network. As training progresses, the agent's policy gradually converges to the optimal policy π^*. This means that for a given state, the agent can choose the action that maximizes the expected cumulative return. It can be expressed as:

$$\pi^* = argmax_a Q(s_t, a). \tag{11}$$

Through this training method, both agents learn optimal actions in their state spaces to maximize cumulative rewards. Information is exchanged between the two agents, and their decisions are guided by reward signals generated by the environment. These signals can be fed back to both agents to help them adjust their strategies. Through effective information exchange, the two agents form a more powerful and flexible system to better cope with complex and dynamic environments, thereby significantly improving the effectiveness of reinforcement learning.

4 Experiments

4.1 Experimental Setup

The experimental objectives include validating the improvement of our method compared to other feature generation methods, verifying the effectiveness of each part of the method through ablation experiments, demonstrating the stability of the algorithm on different downstream learning tasks, and comparing the runtime to validate algorithm time superiority. We conduct experiments on 21 datasets from UCI [Public, 2024b], Kaggle [Howard, 2024], and OpenML [Public, 2024a], LibSVM [Lin, 2024], comprising 12 classification tasks and 9 regression tasks. The evaluation metrics are as follows: for classification tasks, we used the *F1-score* to evaluate recall and precision. For regression tasks, we use the 1-relative absolute error (1-RAE) to evaluate accuracy. The code is available at https://github.com/extess0/DARL.

4.2 Environmental Settings

All experiments are conducted on the Ubuntu operating system, Intel(R) Core(TM) i9-10900X CPU@ 3.70GHz, and V100, with the framework of Python 3.10.12 and PyTorch 1.13.1.

4.3 Evaluation Metrics

The equation for 1-RAE is as follows:

$$1 - RAE = 1 - \frac{\sum_{i=1}^{n} |y_i - y_i^*|}{\sum_{i=1}^{n} |y_i - y_m|}, \tag{12}$$

y_i^* is the actual target value of the i-th observation, y_i is the predicted target value of the i-th observation, and y_m is the mean of all actual target values.

Datasets	Source	C/R	Samples/Features	Base	Random	PCA	DFS	Autofeat	Bigfeat	NFS	GRFG	DARL
PimaIndian	UCIrvine	C	768/8	0.7566	0.7670	0.6444	0.7579	0.7566	0.7461	0.7806	0.7857	**0.7904**
German Credit	UCIrvine	C	1001/24	0.7390	0.7620	0.5910	0.7610	0.7540	0.7370	0.7720	0.7740	**0.7770**
SPECTF	UCIrvine	C	267/44	0.7751	0.8462	0.8051	0.7515	0.7856	0.8238	0.8500	0.8568	**0.8688**
Ionosphere	UCIrvine	C	351/34	0.9233	0.9260	0.6893	0.9401	0.9381	0.9203	0.9516	0.9554	**0.9601**
Wine Quality Red	UCIrvine	C	999/12	0.5395	0.5485	0.5135	0.5085	0.5275	0.5425	0.5736	0.5774	**0.5856**
Wine Quality White	UCIrvine	C	4900/12	0.4976	0.5086	0.4710	0.4798	0.5021	0.4955	0.5117	0.5142	**0.5202**
Ilpd	OpenML	C	583/10	0.6878	0.7342	0.6123	0.6929	0.6843	0.6980	0.7428	0.7387	**0.7512**
Svmguide3	LibSVM	C	1243/21	0.7989	0.8303	0.6678	0.8206	0.8230	0.8166	0.8339	0.8359	**0.8391**
Messidor Features	UCIrvine	C	1150/19	0.6594	0.7411	0.5543	0.7524	0.7324	0.6498	0.7462	**0.7482**	0.7433
Lymphography	UCIrvine	C	148/18	0.8170	0.8655	0.8577	0.8673	0.8444	0.8037	0.8717	**0.8746**	0.8715
Airfoil	UCIrvine	R	1503/5	0.5118	0.6106	0.4586	0.6074	0.5927	0.4980	0.6134	0.6197	**0.6273**
Housing Boston	UCIrvine	R	506/13	0.4378	0.4597	0.2020	0.4760	0.4295	0.4203	0.4937	0.5012	**0.5123**
Openml_586	OpenML	R	1000/25	0.6635	0.6321	0.3994	0.7187	0.7109	0.6635	0.7210	0.7310	**0.7430**
Openml_607	OpenML	R	1000/50	0.6498	0.6367	0.2484	0.6814	0.6624	0.6363	0.6573	0.7005	**0.7168**
Openml_618	OpenML	R	1000/50	0.6448	0.6194	0.2744	0.6848	0.6797	0.6351	0.6563	0.7071	**0.7144**
Openml_592	OpenML	R	1000/25	0.6633	0.6578	0.2831	0.6939	0.6960	0.6633	0.6782	0.6926	**0.7290**
Openml_584	OpenML	R	500/25	0.5826	0.5827	0.2153	0.5977	0.6356	0.5826	0.6020	0.6873	**0.6929**
Openml_599	OpenML	R	1000/5	0.7199	0.7011	0.6344	0.7802	0.7233	0.7199	0.7819	0.7642	**0.7937**
Bikeshare DC	Kaggle	R	10886/11	0.9880	0.9920	0.9862	0.9993	0.9909	0.9995	0.9991	0.9994	**0.9996**
Ap-omentum-ovary	OpenML	C	275/10936	0.7818	0.4550	0.5927	0.3787	0.4570	0.8072	0.8509	0.8691	**0.8764**
Adult Income	UCIrvine	C	48842/14	0.8498	0.8494	0.5916	0.8483	0.8463	8.8295	0.8501	0.8505	**0.8514**

Table 1: Overall performance comparison.

F1-score is the harmonic mean of Precision and Recall.

$$F_1 = 2 * \frac{Precision * Recall}{Precision + Recall}. \tag{13}$$

Precision measures the proportion of positive predictions that are correctly labeled, defined as $Precision = TP/(TP + FP)$. *TP* stands for true positives, while *FP* stands for false positives. Recall measures the proportion of actual positive samples that are correctly identified by the model, given by: $Recall = TP/(TP + FN)$. *FN* represents false negatives.

4.4 Hyperparameter Settings

The number of epochs is limited to 200. By using 6 exploration steps per epoch, we further control the number of features generated. We adopt random forest as the downstream machine learning model and performed 5-fold stratified cross-validation in all experiments, instead of a simple 70%-30% split. We used the Adam [Kingma and Ba, 2015] optimizer with a learning rate of 0.0001 to optimize DQN, and set the memory limit of experience replay to 24, and the DQN batch size to 8. The model incorporated 8 attention heads, with a word embedding vector dimension of 8 and a model hidden layer dimension of 128. The discrimination agent's reward weights α, β, γ, and δ are set to 0.1, 0.1, 1, and 0.01.

4.5 Baseline Methods

We compare our method with 8 widely used feature generation methods, as well as random generation and feature dimension reduction methods: (1) Base: using the original dataset without feature generation. (2) Random: randomly generating features for each feature. (3) DFS [Kanter and Veeramachaneni, 2015]: an expansion-reduction method that first expands and then selects feature, automatically generated features for the dataset. (4) PCA [Candès *et al.*, 2011]: a feature reduction method that compresses the original feature set. (5) Autofeat [Horn *et al.*, 2020] is an expansion-reduction algorithm that generates nonlinear features and selects a subset of relevant features to form a new dataset. (6) Bigfeat [Eldeeb *et al.*, 2022]: proposes a dynamic feature generation technique based on computational tree logic. (7) NFS [Chen *et al.*, 2019]: uses the RNN as a controller for each original feature, which outputs actions to generate new features. (8) GRFG [Wang *et al.*, 2022]: iteratively generates new features and reconstructs an interpretable feature space through group-group interactions. We conduct experiments on the open-source code provided by these methods.

4.6 Overall Comparison

Table 1 shows the comparison of our method with eight baseline models on the 21 datasets in terms of *F1-score* or 1-RAE. Our method outperformed the others on most datasets. Additionally, to validate the performance of the proposed method on large datasets where both the feature and sample sizes exceed 10,000, we conducted experiments on three large datasets. The results confirmed the effectiveness of our proposed method. It can be observed that our dual-agent feature generation method outperforms other feature generation methods by maximizing cumulative rewards through reinforcement learning.

4.7 Ablation Study

This experiment aims to verify whether each component of our method indeed has a positive impact on the final results. Therefore, we have developed three variants: (1) The feature discrimination agent was replaced with mutual information feature selection. This change was made to verify whether constraining feature quantity through the feature discrimination agent is more effective than using mutual information. This variant is referred to as "DARL-k". (2) The state of

Feature Types and Proportions	Openml_584	Openml_599	Housing Boston	Airfoil	Ilpd	Ionosphere	German Credit	PimaIndian
Low-order features	33	11	9	11	11	0	1	2
High-order features	147	26	47	175	10	13	38	4
Proportion of high-order features	81.7%	70.3%	83.9%	94.1%	47.6%	100.0%	97.4%	66.7%

Table 2: Proportion of high-order features generated by different datasets.

the feature generation agent does not utilize self-attention encoding. This is to verify if self-attention encoding is more effective. This variant is referred to as "DARL-t". (3) For discrete features, no cross-operation and feature encoding are performed to examine the necessity of distinguishing discrete features from continuous features. This variant is referred to as "DARL-c". We validate these variants on four datasets. Tabel 3 shows that the best performance is achieved when all components are present. This validates the effectiveness of our method: dual-agent reinforcement learning is more effective than traditional feature selection methods, as reinforcement learning can maximize cumulative rewards. Self-attention encoding is more effective than not using it, as it considers deep connections between features. Distinguishing between feature types is more effective than not distinguishing them, as there are inherent differences between discrete and continuous features. It is noteworthy that not distinguishing between discrete and continuous features yields similar results on regression datasets. This is because regression task datasets often lack discrete features, only containing continuous features, making the differentiation between discrete and continuous features irrelevant.

Datasets	DARL	DARL-k	DARL-t	DARL-c
Airfoil	**0.6273**	0.5244	0.6241	**0.6273**
German Credit	**0.7770**	0.7750	0.7650	0.7680
Housing Boston	**0.5123**	0.4811	0.4982	**0.5123**
PimaIndian	**0.7904**	0.7891	0.7892	0.7800

Table 3: Comparison of different DARL variants in terms of *F1-score* or 1-RAE.

4.8 Robustness Analysis

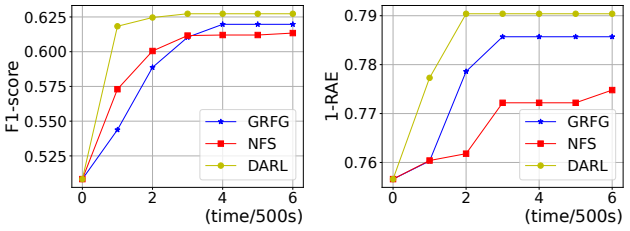
Datasets	XGB		SVM		Logistic	
	Base	DARL	Base	DARL	Base	DARL
Airfoil	0.4692	**0.6301**	0.0340	**0.0430**	0.2792	**0.3406**
SPECTF	0.8015	**0.8313**	0.7901	**0.8050**	0.8014	**0.8238**
Ilpd	0.6827	**0.7067**	0.7086	**0.7153**	0.7187	**0.7221**
PimaIndian	0.7513	**0.7566**	0.7693	**0.7826**	0.7696	**0.7800**

Table 4: Comparison of different machine learning models in terms of *F1-score* or 1-RAE.

This experiment aims to compare the performance improvement achieved by using various machine learning algorithms to process the new dataset against the original dataset. This will help verify the robustness of the method. We employed XGBoost (XGB), Support Vector Machine (SVM), and Logistic Regression (Log) on the new dataset to validate if the newly generated feature set still enhances performance compared to the original dataset. As Table 4 shows, we conducted experiments on 4 datasets and observed that the scores of the new datasets were consistently higher than those of the original datasets, indicating the high robustness of our method.

4.9 Efficiency Improvement

This experiment aims to verify the optimality of our method in terms of efficiency. By comparing the running time of our method with NFS and GRFG on two datasets, Figure 3 shows that our method consumes the least amount of time and achieves the best scores.



(a) PimaIndian (b) Airfoil

Figure 3: Time comparison of different algorithms.

4.10 Feature Order Analysis

High-order features can capture more complex patterns and relationships in the data. The experiments in Table 2 aim to determine whether the newly generated dataset contains high-order features. We define features with an order greater than or equal to 2 as high-order features. Experiments conducted on 8 datasets show that on most of them, more than 70% of the features are high-order features.

5 Conclusion

In this paper, we propose a novel dual-agent reinforcement learning (DARL) method for feature generation. This method aims to capture complex relationships between features to enhance the performance of machine learning tasks. In DARL, two agents are designed for the reinforcement learning framework, where the first agent generates new features, and the second agent determines whether the generated features are worth preserving. We propose to use a self-attention mechanism to enhance state representation, effectively capturing the complex relationships between features. We propose to distinguish between discrete and continuous feature interactions to generate more interpretable features. Extensive experiments have demonstrated that DARL exhibits significant effectiveness in feature generation when compared to other baseline methods. Pseudocode of the DARL, experimental settings, comparison of different downstream task and convergence analysis are presented in Appendix.

Acknowledgments

This work was supported by the Science Foundation of Jilin Province of China under Grant YDZJ202501ZYTS286, and in part by Changchun Science and Technology Bureau Project under Grant 23YQ05.

References

[Candès *et al.*, 2011] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM (JACM)*, 58(3):1–37, 2011.

[Chen *et al.*, 2019] Xiangning Chen, Qingwei Lin, Chuan Luo, Xudong Li, Hongyu Zhang, Yong Xu, Yingnong Dang, Kaixin Sui, Xu Zhang, Bo Qiao, et al. Neural feature search: A neural architecture for automated feature engineering. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 71–80. IEEE, 2019.

[Chen *et al.*, 2021] Yi-Wei Chen, Qingquan Song, and Xia Hu. Techniques for automated machine learning. *ACM SIGKDD Explorations Newsletter*, 22(2):35–50, 2021.

[Dor and Reich, 2012] Ofer Dor and Yoram Reich. Strengthening learning algorithms by feature discovery. *Information Sciences*, 189:176–190, 2012.

[Eldeeb *et al.*, 2022] Hassan Eldeeb, Shota Amashukeli, and Radwa ElShawi. Bigfeat: Scalable and interpretable automated feature engineering framework. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 515–524. IEEE, 2022.

[Fan *et al.*, 2010] Wei Fan, Erheng Zhong, Jing Peng, Olivier Verscheure, Kun Zhang, Jiangtao Ren, Rong Yan, and Qiang Yang. Generalized and heuristic-free feature construction for improved accuracy. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA*, pages 629–640. SIAM, 2010.

[Horn *et al.*, 2020] Franziska Horn, Robert Pack, and Michael Rieger. The autofeat python library for automated feature engineering and selection. In *Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, pages 111–120. Springer, 2020.

[Howard, 2024] Jeremy Howard. Kaggle dataset download. https://www.kaggle.com/datasets, 2024. Accessed: 2024-05-01.

[Huang *et al.*, 2020] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar S. Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *CoRR*, abs/2012.06678, 2020.

[Kanter and Veeramachaneni, 2015] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international conference on data science and advanced analytics (DSAA)*, pages 1–10. IEEE, 2015.

[Katz *et al.*, 2016] Gilad Katz, Eui Chul Richard Shin, and Dawn Song. Explorekit: Automatic feature generation and selection. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 979–984. IEEE, 2016.

[Khurana *et al.*, 2018] Udayan Khurana, Horst Samulowitz, and Deepak S. Turaga. Feature engineering for predictive modeling using reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 3407–3414. AAAI Press, 2018.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[Kulkarni *et al.*, 2016] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

[Li *et al.*, 2024] Yonghao Li, Liang Hu, and Wanfu Gao. Multi-label feature selection with high-sparse personalized and low-redundancy shared common features. *Information Processing & Management*, 61(3):103633, 2024.

[Lin, 2024] Chih-Jen Lin. Libsvm dataset download. https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/, 2024. Accessed: 2024-05-01.

[Lipton *et al.*, 2015] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning, 2015.

[Luo *et al.*, 2019] Yuanfei Luo, Mengshuo Wang, Hao Zhou, Quanming Yao, Wei-Wei Tu, Yuqiang Chen, Wenyuan Dai, and Qiang Yang. Autocross: Automatic feature crossing for tabular data in real-world applications. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1936–1945, 2019.

[Markovitch and Rosenstein, 2002] Shaul Markovitch and Dan Rosenstein. Feature generation using general constructor functions. *Mach. Learn.*, 49(1):59–98, 2002.

[Mnih *et al.*, 2013] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[Morimoto and Doya, 2001] Jun Morimoto and Kenji Doya. Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. *Robotics Auton. Syst.*, 36(1):37–51, 2001.

[Obese, 1998] HJOR Obese. Body mass index (bmi). *Obes Res*, 6(2):51S–209S, 1998.

[Public, 2024a] Public. Openml dataset download. https://www.openml.org, 2024. Accessed: 2024-05-01.

[Public, 2024b] Public. UCI Dataset Download. https://archive.ics.uci.edu/, 2024. Accessed: 2024-05-01.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Wang *et al.*, 2022] Dongjie Wang, Yanjie Fu, Kunpeng Liu, Xiaolin Li, and Yan Solihin. Group-wise reinforcement feature generation for optimal and explainable representation space reconstruction. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1826–1834, 2022.

[Xiao *et al.*, 2022] Meng Xiao, Dongjie Wang, Min Wu, Kunpeng Liu, Hui Xiong, Yuanchun Zhou, and Yanjie Fu. Self-optimizing feature transformation. *arXiv preprint arXiv:2209.08044*, 2022.

[Ying *et al.*, 2023] Wangyang Ying, Dongjie Wang, Kunpeng Liu, Leilei Sun, and Yanjie Fu. Self-optimizing feature generation via categorical hashing representation and hierarchical reinforcement crossing. In *2023 IEEE International Conference on Data Mining (ICDM)*, pages 748–757. IEEE, 2023.

[Zhang and Gao, 2021] Ping Zhang and Wanfu Gao. Feature relevance term variation for multi-label feature selection. *Applied Intelligence*, 51:5095–5110, 2021.

[Zhang *et al.*, 2021] Ping Zhang, Wanfu Gao, Juncheng Hu, and Yonghao Li. A conditional-weight joint relevance metric for feature relevancy term. *Engineering Applications of Artificial Intelligence*, 106:104481, 2021.

[Zhang *et al.*, 2023] Tianru Zhang, Andreas Hellander, and Salman Toor. Efficient hierarchical storage management empowered by reinforcement learning. *IEEE Trans. Knowl. Data Eng.*, 35(6):5780–5793, 2023.

[Zhang *et al.*, 2024] Xinhao Zhang, Zaitian Wang, Lu Jiang, Wanfu Gao, Pengfei Wang, and Kunpeng Liu. Tfwt: Tabular feature weighting with transformer, 2024.