Comprehend, Divide, and Conquer: Feature Subspace Exploration via Multi-Agent Hierarchical Reinforcement Learning

WEILIANG ZHANG*, Computer Network Information Center, Chinese Academy of Sciences University of Chinese Academy of Sciences, China

XIAOHAN HUANG*, Computer Network Information Center, Chinese Academy of Sciences University of Chinese Academy of Sciences, China

YI DU, Computer Network Information Center, Chinese Academy of Sciences, China

ZIYUE QIAO, Great Bay University, China

QINGQING LONG, Computer Network Information Center, Chinese Academy of Sciences, China

ZHEN MENG, Computer Network Information Center, Chinese Academy of Sciences, China

YUANCHUN ZHOU, Computer Network Information Center, Chinese Academy of Sciences, China

MENG XIAO†, Computer Network Information Center, Chinese Academy of Sciences, China and Duke-NUS Medical School, National University of Singapore, Singapore

Feature selection aims to preprocess the target dataset, find an optimal and most streamlined feature subset, and enhance the downstream machine learning task. Among filter, wrapper, and embedded-based approaches, the reinforcement learning (RL)-based subspace exploration strategy provides a novel objective optimization-directed perspective and promising performance. Nevertheless, even with improved performance, current reinforcement learning approaches face challenges similar to conventional methods when dealing with complex datasets. These challenges stem from the inefficient paradigm of using one agent per feature and the inherent complexities present in the datasets. This observation motivates us to investigate and address the above issue and propose a novel approach, namely HRLFS. Our methodology initially employs a Large Language Model (LLM)-based hybrid state extractor to capture each feature's mathematical and semantic characteristics. Based on this information, features are clustered, facilitating the construction of hierarchical agents for each cluster and sub-cluster. Extensive experiments demonstrate the efficiency, scalability, and robustness of our approach. Compared to contemporary or the one-feature-one-agent RL-based approaches, HRLFS improves the downstream ML performance with iterative feature subspace exploration while accelerating total run time by reducing the number of agents involved. [1]

*Equal contribution
[1]Our codes and data are publicly accessible via Dropbox.

† Corresponding Author.

Authors' Contact Information: Weiliang Zhang, Computer Network Information Center, Chinese Academy of Sciences and University of Chinese Academy of Sciences, China, wlzhang@cnic.cn; Xiaohan Huang, Computer Network Information Center, Chinese Academy of Sciences and University of Chinese Academy of Sciences, China, xhhuang@cnic.cn; Yi Du, Computer Network Information Center, Chinese Academy of Sciences, China, duyi@cnic.cn; Ziyue Qiao, Great Bay University, China, zyqiao@gbu.edu.cn; Qingqing Long, Computer Network Information Center, Chinese Academy of Sciences, China, qqlong@cnic.cn; Zhen Meng, Computer Network Information Center, Chinese Academy of Sciences, China, zhenm99@cnic.cn; Yuanchun Zhou, Computer Network Information Center, Chinese Academy of Sciences, China, zyc@cnic.cn; Meng Xiao†, Computer Network Information Center, Chinese Academy of Sciences, China and Duke-NUS Medical School, National University of Singapore, Singapore, meng.xiao@nus.edu.sg.

CCS Concepts: • **Computing methodologies** → **Feature selection**.

Additional Key Words and Phrases: Automated Feature Engineering, Tabular Dataset, Reinforcement Learning.

ACM Reference Format:
Weiliang Zhang, Xiaohan Huang, Yi Du, Ziyue Qiao, Qingqing Long, Zhen Meng, Yuanchun Zhou, and Meng Xiao†. 2018. Comprehend, Divide, and Conquer: Feature Subspace Exploration via Multi-Agent Hierarchical Reinforcement Learning. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX).* ACM, New York, NY, USA, 23 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Feature selection (FS) plays a critical role in classical machine learning models by mitigating the curse of dimensionality, reducing training times, and alleviating feature redundancy, thereby substantially enhancing predictive performance [1–3]. The evolution of FS methodologies has progressed from early heuristic and statistical screening techniques to the development of sophisticated approaches that incorporate filter-based [4–6], wrapper-based [7, 8], and embedded strategies [9–11] tailored to manage increasingly complex data environments. Concurrently, the rapid advancement of modern artificial intelligence [12, 13] has broadened the application domain of FS to encompass challenging tasks such as data pattern discovery [14], biomarker identification [15, 16], and the construction of AI-driven data pipelines (AI4data) [17–20]. In the context of emerging demands in multi-modal data processing and real-time big data analytics, integrating efficient and intelligent feature selection is increasingly recognized as a pivotal element in enhancing data quality and advancing data-centric paradigms [21].

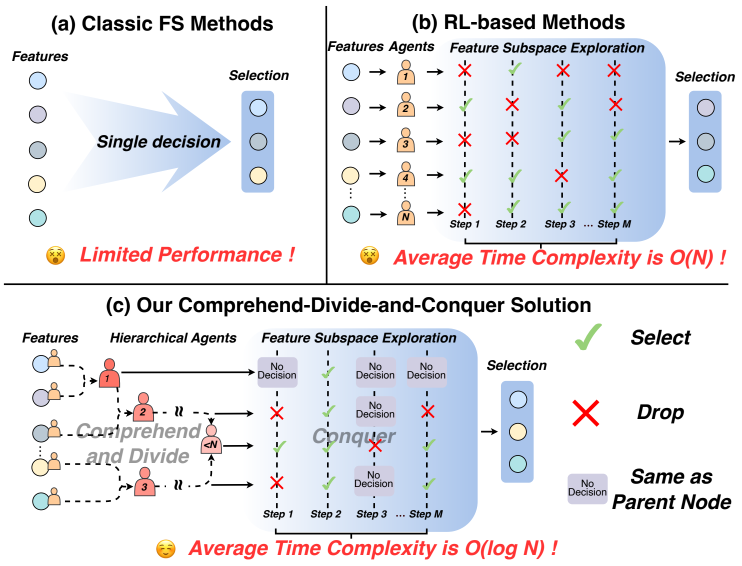


Fig. 1. Comparison between HRLFS with other feature selection approaches.

Among the various feature selection methodologies, reinforcement learning (RL)-based strategies [22] have received significant attention due to their ability to optimize feature subsets in an objective-directed manner with iterative feature subspace exploration. Despite these advances, RL-based feature selection methods encounter notable challenges when handling complex datasets. This difficulty primarily stems from the limited capability of a single-agent-all-feature framework [22]. Current RL-based research has made some progress in addressing the complexity of datasets. An insightful approach [23] involves adopting a multi-agent reinforcement learning [24] (MARL) architecture for feature sub-space exploration. However, the limitation of the one-agent-per-feature system becomes evident, as it necessitates an excessive number of agents when processing high-dimensional datasets.

In summary, as depicted in Figure 1(a), classic FS methods perform feature selection through a single decision, which is efficient but analyzes the nature of the features inadequately, resulting in lower performance. Present RL-based Methods as depicted in Figure 1(b). Those methods of subspace exploration come from wrapper feature selection ideas and have powerful feature selection capabilities. Still, they also have the inherent drawbacks of wrapper methods, i.e., they have $O(N)$ time complexity and require a lot of time when dealing with large datasets. To mitigate the challenge of managing an overwhelming number of agents, group-based [25] and interaction-wise [26] agent architectures have been proposed as promising solutions. Nevertheless, these approaches rely solely on superficial mathematical characteristics (e.g., standard variance of features) to organize features, resulting in inaccuracies due to neglecting semantic and contextual feature relationships [27, 28].

Those observations motivate us to develop novel hierarchical reinforcement learning-based architectures for feature sub-space exploration. The core concept of this research is that when a feature (e.g., age) is irrelevant to a task, its semantically equivalent feature (e.g., birthdate) may also be irrelevant. Additionally, two features exhibiting similar numerical patterns (e.g., highly related biological signals) could be redundant, necessitating fine-grained differentiation. For example, Mice-Protein [29] is utilized in research on mouse Down syndrome and comprises 77 key proteins linked to learning ability in the mouse brain. It contains two genes, 'pGSK3B_N' and 'GSK3B_N', that are highly correlated biologically, because they correspond to the same protein, 'GSK-3β', which represents different forms of the protein during its expression [29]. Conventional feature selection methods struggle with such intrinsic biological redundancy. A one-agent-one-feature approach, for example, makes independent decisions and often fails to remove both features due to a lack of agent collaboration. Meanwhile, group-based and interaction-wise methods based on purely mathematical correlations can detect their statistical link but lack the semantic context to understand the underlying biological reason. Our approach uses semantic information to identify latent relationships and employs a hierarchical structure for unified decision-making. This addresses the limitations of the aforementioned methods in terms of feature understanding, agent collaboration, and decision-making efficiency.

As depicted in Figure 1(c), our solution builds hierarchical agents to make decisions on feature subspace exploration tasks, reducing the average decision time complexity to $O(log\ N)$ while maintaining high performance. The core idea can be divided into three stages: (**Comprehend**) Our primary innovation lies in integrating Large Language Models to comprehend the semantic meanings of feature metadata, coupled with Gaussian Mixture Models (GMM) to capture the mathematical characteristics of the features. (**Divide**) Building upon these enhanced feature representations, we employ a clustering mechanism that groups similar features. (**Conquer**) Utilizing these clusters, we construct a multi-agent hierarchical reinforcement learning framework that mirrors the natural organization of the feature space. This hierarchical structure strategically divided decision-making responsibilities to cluster-specific agents and adaptively reduced the required activated agents, significantly reducing computational overhead and accelerating the exploration process. Our contributions can be summarized as:

- **Comprehensive Feature Understanding.** We harness the Large Language Models and Statistical Differential to extract meaningful information from dataset metadata and their mathematical characteristics, thereby enabling a deeper understanding of feature relationships beyond numerical statistics.

- **Beyond One-agent-one-feature Architecture.** The clustered features are managed through a hierarchical multi-agent reinforcement learning architecture, which reduces the total number of activated agents, enhances computational efficiency, and improves the feature selection process.
- **Theoretical and Empirical Efficiency.** We rigorously demonstrate the efficiency and effectiveness of our method from both theoretical and experimental perspectives, showing its superior predictive performance and computational scalability compared to existing feature selection approaches.

2 Important Definitions

Feature Selection. Given the dataset $D = \{X \in \mathcal{R}^{m \times n}, y \in \mathcal{R}^{m \times 1}, F\}$, where X and y are the features and labels, respectively. m and n is the number of samples and features. We use a finite set $F = \{f_1, f_2, ..., f_n\}$ to indicate the feature column included in dataset D, where f_i is the $i-th$ column of X. The goal of feature selection is to find the optimal feature subset $F^* \subset F$ to enhance model performance while maintaining computational efficiency.

Combine Feature Selection with RL. In this paper, we frame the decision process of the feature selection method as a Markov Decision Process (MDP) [30]. Specifically, s_t represents the state of the selected feature F_t at time step t. The agent(s), with policy function(s), $\pi(\cdot)$, will select or drop each feature with an action $a_t \in \{0, 1\}^n$, where each action component corresponds to including (1) or excluding (0) a specific feature. With the action a_t, we could build a subset of the feature F_{t+1} and extract its related new state s_{t+1}. We can also evaluate the selection and obtain the reward r_t. With the collection of memory $m_t = (s_t, a_t, s_{t+1}, r_t)$, we could optimize the policy function(s) $\pi(\cdot)$ toward reinforcement learning and finally obtain the optimal selection F^*.

3 Methodology

3.1 Overview of HRLFS

HRLFS is a highly effective RL-based feature subspace exploration method that adopts a comprehend-divide-and-conquer paradigm. HRLFS will first extract the hybrid state of each feature, cluster each feature, and then initialize the hierarchical agent architecture. By using the state of each feature, the hierarchical agents will explore the feature combination and optimize its selection policy. After that, HRLFS will output the final optimal selection to enhance the downstream machine-learning task.

3.2 Hybrid Feature State Extraction

As illustrated in Figure 2, we develop a hybrid-faceted feature state extraction method to help both the clustering component and each RL agent comprehend the given dataset.

Leveraging Feature Distribution. We first consider the mathematical characteristics of each feature. To achieve that, we model each feature f using a Gaussian Mixture Model with k Gaussian component to accurately capture the distribution of the feature values. The probability density function (pdf) of the feature f is defined as:

$$P(f|\theta) = \sum_{i=1}^{k} z_i N_i \left(f \mid \mu_i, \sigma_i\right), \text{ where } \sum_{i=1}^{k} z_i = 1. \quad (1)$$

Here, $N_i(f \mid \mu_i, \sigma_i)$ denote the i-th Gaussian distribution, z_i is a non-negative constant that controls the weight of each component. $\theta = (z_1, ..., z_k, \mu_1, ..., \mu_k, \sigma_1, ..., \sigma_k)$ are the overall parameters of the GMM model. We then use the following

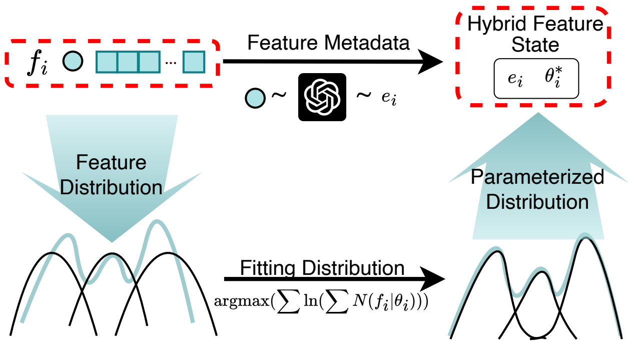


Fig. 2. Hybrid feature state extraction.

log-likelihood function to maximize the likelihood between PDF to real distribution of F, given as:

$$l(\Theta \mid F) = \sum_{f \in F} \ln \left(\sum_{i=1}^{k} z_i N \left(f \mid \mu_i, \sigma_i \right) \right), \tag{2}$$

where $\Theta = \{\theta_i\}_{i=1}^n$ denotes all the parameters for n features. By that, we apply the expectation maximization (EM) algorithm to optimize the loss function and obtain the final parameter θ_i^* as the distribution state of the feature f_i.

Leveraging Feature Metadata. Another key source to address feature state inaccuracy is to obtain semantic information from feature descriptions (i.e., metadata). We employ a straightforward but efficient approach: input the feature name-description pair into a large language model and subsequently utilize PCA [31] for dimensionality reduction to align with the Gaussian component number k. This process results in sentence embeddings that serve as the semantic feature state. For incomplete datasets, we use the following prompt to generate its feature description. As shown in the figure 3, we feed the description of the dataset, the available feature names and their descriptions, and the names of the features without descriptions to the large language model to complete the missing feature description. Further, we use an all-zero

```
Prompt: I will provide a dataset description along with examples of features and their
descriptions. Please interpret the other features that do not have descriptions.
Dataset description: This dataset represents ...
Examples: word_freq_WORD = percentage of words in the e-mail that match WORD, i.e. 100 *
(number of times the WORD appears in the e-mail) / total number of words in e-mail...
'''
[feature name]
'''
using the following format:
[feature name]: [DESCRIPTION HERE]
```

Fig. 3. The prompt to generate feature description using dataset metadata.

embedding as its semantic feature state for the dataset with both no dataset description and feature description (e.g., a synthetic dataset). By that, we can obtain the semantic state $e_i \in \mathbf{R}^k$ for each feature f_i.

Finally, we concatenate the distribution and semantic states to generate the hybrid state, given as $h_i = e_i \oplus \theta_i^*$.

3.3 Divide-and-Conquer Agent Architecture

We then introduce how we build the agent hierarchy (divide) and organize the agent decision (conquer) to overcome the challenge of the dataset complexity.

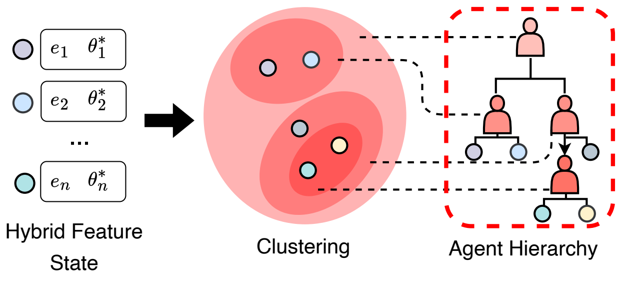


Fig. 4. Construction of agent hierarchy by incremental clustering.

3.3.1 Agent Hierarchy Construction. As illustrated in Figure 4, we propose H-clustering algorithm that aggregates the features by the similarity of their hybrid state and uses this group information to build the hierarchy of the agents. Specifically, we employ an incremental clustering algorithm to merge two clusters that cause the smallest increase in variance [32]. Algorithm 1 illustrates that starting with all feature states $H = \{h_i\}_{i=1}^n$, where every feature initially forms a cluster, and new clusters are sequentially merged through similarity grouping until just one remaining cluster is achieved. We then derive the cluster set C_{agent} where each cluster $C_i \in C_{\text{agnet}}$ signifies an Agent $\mathcal{A}_i$. Inclusion relationships establish the agents' hierarchy; a lower-level agent $\mathcal{A}_i$ is part of a higher-level agent $\mathcal{A}_j$ if $C_i \subset C_j$ (with two lower-level agents combining to form a higher-level agent).

Algorithm 1 Agent Hierarchy Construction

```
Input: Feature number $n$, feature embeddings $H = \{h_1, ..., h_n\}$
Output: A set of clusters $C_{\text{agent}}$
1:  # Initialize each feature as a cluster.
2:  $C_i \leftarrow \{f_i\}$
3:  $C \leftarrow \{C_1, ..., C_n\}$
4:  $C_{\text{agent}} \leftarrow \{\}$
5:  while $|C| > 1$ do
6:      # Calculate the distance between each two clusters by Ward's method.
7:      $d_{ij} = \frac{|C_i||C_j|}{|C_i|+|C_j|}||\overline{h_{ci}} - \overline{h_{cj}}||^2$, where $|\cdot|$ represents the number of items
        and $\overline{h_{ci}}, \overline{h_{cj}}$ is the mean vector of each feature's state within $C_i$ and $C_j$.
8:      # Find the two most similar clusters. Then aggregate them into new cluster.
9:      $C_{\text{new}} \leftarrow C_a \cup C_b$
10:     # add $C_{new}$ to $C$ and $C_{\text{agent}}$.
11:     $C \leftarrow C \setminus C_a$
12:     $C \leftarrow C \setminus C_b$
13:     $C \leftarrow C \cup \{C_{\text{new}}\}$
14:     $C_{\text{agent}} \leftarrow C_{\text{agent}} \cup \{C_{\text{new}}\}$
15: end while
```

Hierarchical Agent for Cluster-specific Decision. As shown in Figure 5, agents work together to make decisions across various granular levels within a hierarchical framework, extending from higher to lower-level agents. Each agent shares states and rewards while developing policies. The leaf nodes of the tree correspond to individual features and are responsible for making fine-grained decisions on whether to select or discard each specific feature. In contrast, the internal (non-leaf) nodes represent higher-level feature clusters and make broader decisions by partitioning the feature space, determining whether entire groups of features—represented by their respective subtrees—should be further considered or pruned from the selection process.

◇ *Action:* At the t-th iteration, the action a_t^i associated with the i-th agent (corresponding to a node in the tree) is a binary choice: $a_t^i \in$ select, drop. If an agent (tree node) chooses the 'select' action, it recursively delegates the feature selection task to its child nodes (i.e., the subtrees rooted at its children are further explored). Conversely, if the 'drop' action is selected by a node, the entire subtree rooted at this node becomes inactive, and none of its descendant agents (or their associated feature clusters) are activated. In this way, a 'drop' action at any internal node results in a pruning of the corresponding subtree from the selection process.

◇ *State* The state for each agent is a vectorized representation derived from the selected feature subset. We adopt the hybrid feature state as the state representation for each agent. We first acquire the parameter $\theta_i^* = \{z_1^i, \cdots, z_k^i, \mu_1^i, \cdots, \mu_k^i, \sigma_1^i, \cdots, \sigma_k^i\}$ from GMM for the feature f_i. Due to the z adjustment of the weight of each Gaussian component, we weighted-sum each μ and σ toward the corresponding z to form a state vector s_i, formally:

$$s_i = e_i \oplus \sum_{j=1}^{k} z_j^i * (\mu_j^i \oplus \sigma_j^i), \tag{3}$$

To obtain the state of step t, we concatenate all feature states to form a fixed-size vector $\mathbf{s}_t \in \mathbf{R}^{n\times(k+2)}$:

$$\mathbf{s}_t = \bigoplus_{i=1}^{n} (\mathbb{1}_t(f_i) * s_i) \tag{4}$$

where $\mathbb{1}_t(\cdot)$ is an indicator function denoting that the feature f_i is selected or not in step t. If f_i is not chosen, the function yields 0; if selected, it yields 1.

◇ *Policy* The agent's policy network is a feed-forward neural network with a binary classification head. Formally, for feature f_i, its action in t-th iteration is then derived by $a_t^i = \pi_i(\mathbf{s}_{t-1})$.

◇ *Reward* As illustrated in Figure 5, we design the reward function regarding downstream task performance and quantity suppression of the selected feature numbers.

⋆ *Performance*: the first aspect of the reward function evaluates performance through downstream tasks, such as classification and regression. We train a downstream task model using a selected feature subset and define r_t^p using the model evaluation metrics.

⋆ *Quantity Suppression*: The second perspective focuses on ensuring a compact number of features through:

$$r_t^q = \frac{|F| - |F_t|}{|F| + \lambda \cdot |F_t|} \tag{5}$$

where F is the entire feature set, F_t is the selected feature subset in step-t, λ is a hyperparameter, and $|\cdot|$ denoted the size of given set. As λ increases, the penalty for keeping too many features (large $|\mathcal{F}_t|$) becomes more severe, thus encouraging more substantial gene reduction. Conversely, a lower value of λ relaxes the penalty against the size of $|\mathcal{F}_t|$,

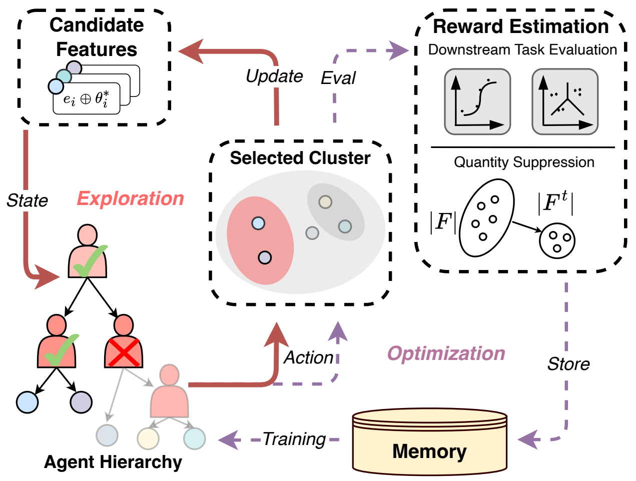


Fig. 5. Detail of the iteration and optimization with the hierarchical agents.

suitable when minimal reduction is sufficient. This metric ensures that the selection process strategically reduces the number of features.

⋆ *Reward Assignment*: Then we combine two perspectives and obtain the reward in step-t:

$$r_t = \alpha \cdot r_t^p + (1 - \alpha) \cdot r_t^q, \tag{6}$$

where r_t is the total reward in this step. α is a hyperparameter to adjust the weight of two perspectives. After obtaining the reward, the framework will assign the reward equally to each activated agent.

3.4 Exploration and Opitmization

Figure 5 demonstrates our approach of using hierarchical reinforced iteration to discover the best subsets of characteristics, partitioning model training into two distinct phases: exploration and optimization.

Exploration Phase. In the exploration phase, hierarchical agents randomly explore the feature subspace. They take the current state s_t as input and select or drop features hierarchically, constructing a new set of selected features. The reward r_t and the next state s_{t+1} are computed from the newly selected features subset. Each agent $\mathcal{A}_i$ taking action a_t in this step stores the experience $m_t = \{\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1}\}$ in its memory M_i.

Optimization Phase. Agents will autonomously train their individual policies through the memory mini-batches obtained from prioritized experience replay [33]. We refined the policy utilizing the Actor-Critic approach [34], where the policy network $\pi(\cdot)$ assumes the role of the actor, while $V(\cdot)$ serves as its associated critic. The optimization objective for agent $\mathcal{A}_i$ is defined by the expected cumulative reward, expressed as:

$$\max_{\pi} \mathbb{E}_{m_t \sim \mathcal{B}} \left[\sum_{t=0}^{T} \gamma_t r_t \right], \tag{7}$$

where $\mathcal{B}$ refers to the distribution of experiences within the prioritized replay buffer, γ is the discount factor, and T denotes the time horizon of an episode. Additionally, we incorporate the Q-function, $Q(\mathbf{s}, a)$, which signifies the expected return for taking action a in state s and adhering to policy π subsequently:

$$Q(\mathbf{s}, a) = \mathbb{E}\left[r + \gamma \max_{a'} Q(\mathbf{s}', a') \mid \mathbf{s}, a\right]. \tag{8}$$

The training adjustments for the actor networks are determined by the policy gradient [35] defined as

$$\nabla_\theta J(\pi) = \mathbb{E}_{m_t \sim \mathcal{B}}\left[\nabla_\theta \log \pi(a_t|\mathbf{s}_t) A(\mathbf{s}_t, a_t)\right]. \tag{9}$$

Here, $A(\mathbf{s}, a) = Q(\mathbf{s}, a) - V(\mathbf{s})$ is the advantage function, which aids in the gradient estimation for policy improvement.

Analysis of the Advancement of HRL. The hierarchical agent structure activates fewer agents in each step, leading to superior performance in the iteration. The single-agent approach uses one agent to make N decisions on N features. Meanwhile, the multi-agent approach uses N agents to make N decisions on N features, and the time complexity of both approaches is $O(N)$. In contrast, we apply a hierarchical structure, which reduces the number of activated agents making decisions, totally using $2N - 1$ agents to make $log(N)$ decisions on N features, with time complexity $O(log(N))$. In the following section, we will provide detailed proof of the time complexity advantage.

3.5 The Proof of Time Complexity of Comprehend-Divide-and-Conquer Solution

In this section, we provide formal proofs of the time complexity of the comprehend-divide-and-conquer feature selection solution, analyzing the best, worst, and average cases. Consider an architecture with N agents organized as a complete binary tree, where each non-leaf agent delegates decision-making to its children with probability p.

Theorem 3.1 (Best-Case Time Complexity). *The best-case time complexity satisfies $O(1)$.*

Proof. The best-case scenario occurs when the root agent terminates the decision process immediately without delegating to its children. This requires only a constant number of operations at the root node:

$$T_{\text{best}}(N) = c_1 \tag{10}$$

where c_1 is a constant. Therefore,

$$T_{\text{best}}(N) \in O(1). \tag{11}$$

□

Theorem 3.2 (Worst-Case Time Complexity). *The worst-case time complexity satisfies $O(N)$.*

Proof. The worst-case scenario occurs when every agent recursively delegates decisions to its children until the leaf nodes are reached, thus activating all N nodes in the tree. For a complete binary tree with $N = 2^h - 1$ nodes (height h),

$$T_{\text{worst}}(N) = \sum_{k=0}^{h-1} 2^k = 2^h - 1 = N. \tag{12}$$

Therefore,

$$T_{\text{worst}}(N) \in O(N). \tag{13}$$

□

Theorem 3.3 (Average-Case Time Complexity). *Under the assumptions below, the average-case time complexity satisfies $O(\log N)$ when $p = 1/2$.*

Assumption 1 (Balanced Tree Structure). *The hierarchical architecture is a perfect binary tree with:*

- *$N = 2^h - 1$ nodes, where h is the tree height;*
- *All leaf nodes at the same depth level;*
- *Each non-leaf node has exactly two subtrees of size $\lfloor N/2 \rfloor$.*

While this assumption is theoretically convenient, in practice our hierarchy closely approximates such a structure. We provide an empirical validation of this assumption in Section 5.9, showing that the balance factors and tree heights observed across 21 datasets are strongly aligned with those of a perfect binary tree.

Assumption 2 (Independent Delegation). *Each non-leaf agent independently delegates decision-making to its children with probability p, where*

$$p \in [0, 1]. \tag{14}$$

Let $E(N)$ denote the expected number of active agents in such a tree. We formalize the analysis as follows:

Lemma 3.4 (Expectation Decomposition). *For $N > 1$, the expected number of active agents satisfies*

$$E(N) = (1 - p) \cdot 1 + p \cdot \left[1 + E\left(\left\lfloor \frac{N}{2} \right\rfloor\right) + E\left(\left\lceil \frac{N}{2} \right\rceil\right)\right], \tag{15}$$

with boundary condition $E(1) = 1$.

Proof. By the law of total expectation:

$$E(N) = \mathbb{E}[\text{Nodes} \mid \text{root terminates}] \cdot (1 - p) \tag{16}$$

$$+ \mathbb{E}[\text{Nodes} \mid \text{root delegates}] \cdot p. \tag{17}$$

Specifically:

- If the root terminates: $\mathbb{E}[\text{Nodes}] = 1$;
- If the root delegates: $\mathbb{E}[\text{Nodes}] = 1 + E\left(\left\lfloor \frac{N}{2} \right\rfloor\right) + E\left(\left\lceil \frac{N}{2} \right\rceil\right)$.

For a perfect binary tree, $\left\lfloor \frac{N}{2} \right\rfloor = \left\lceil \frac{N}{2} \right\rceil = \frac{N}{2}$ when $N = 2^h - 1$, so the recurrence simplifies to

$$E(N) = 1 + 2pE(N/2). \tag{18}$$

□

Theorem 3.5 (Closed-Form Solution). *For $N = 2^h - 1$, the recurrence resolves to*

$$E(N) = \frac{(2p)^{\log_2(N+1)} - 1}{2p - 1}. \tag{19}$$

In particular, when $p = 1/2$,

$$E(N) = 2\log_2(N + 1) + O(1). \tag{20}$$

Proof. Unrolling the recurrence for $N = 2^h - 1$:

$$\begin{aligned} E(N) &= 1 + 2pE(N/2) \\ &= 1 + 2p[1 + 2pE(N/4)] \\ &= \sum_{k=0}^{h-1}(2p)^k + (2p)^h E(1) \\ &= \frac{(2p)^h - 1}{2p - 1} + (2p)^h \cdot 1 \\ &= \frac{(2p)^{h+1} - 1}{2p - 1} \end{aligned} \tag{21}$$

Substituting $h = \log_2(N + 1)$ yields:

$$E(N) = \frac{(2p)^{\log_2(N+1)} - 1}{2p - 1}. \tag{22}$$

For $p = 1/2$, $2p = 1$, so:

$$E(N) = \log_2(N + 1) + 1 = 2\log_2(N + 1) + O(1). \tag{23}$$

□

Therefore, the average-case time complexity satisfies:

$$T_{\text{avg}}(N) \in O(\log N). \tag{24}$$

In Section 5.2, we conducted experiments and the results show that the active agents decreased by 70.61% to 82.30%, and time consumption reduced by 35.49% to 55.26%. These observations are aligned with the conclusion.

4 Experimental Settings

4.1 Dataset Description

We conducted experiments on 21 datasets, which are available for public use from the Feature Selection Benchmark [36],the National Center for Biotechnology Information (NCBI) 's Gene Expression Omnibus (GEO) [37],UCI [38], Kaggle [39], OpenML [40], libSVM [41], etc. These datasets vary among three tasks (classification, multi-label classification, and regression), sample sizes, and number of feature samples. The datasets come from various fields, including biology, finance, and synthetic data. The datasets contain different sample sizes from 253 to 83773. These datasets include a wide range of feature number, from 21 to 20670. To avoid overfitting, we followed existing research [14] for splitting the datasets into training and validation folds. Specifically, we preserved 20% of the samples for validating the efficiency and capabilities of the selected feature. Dataset descriptions, including the number of samples, features, and task type, are listed in Table 1.

4.2 Evaluation Metrics

We used F1-Score (Micro-F1), accuracy, and recall to evaluate the performance of classification tasks (multi-classification tasks). For regression tasks, we utilized the 1-Relative Absolute Error (1-RAE) from studies [42, 43] to evaluate the performance. The higher values indicate better performance for all metrics.

4.3 Baseline Algorithms

We conduct extensive comparisons against 8 methods, encompassing both classical and recent approaches, covering three categories of feature selection: filter, embedded, and RL-based (latest warpper methods). For filter methods, KBest [44] selects the K-best features based on statistical measures, and MCDM [45] uses predefined rankers to assign scores to features and selects them accordingly. for embedded methods, LASSONet [11] utilizes a residual component that integrates feature selection with the model training process, enhancing its generalizability, and GAINS [14] embeds features into a vector space and identifies the optimal subset using a gradient ascent search algorithm; for wrapper methods (including reinforcement learning-based ones), CompFS [46] provides an efficient, single-pass implementation of greedy forward selection, utilizing attention weights at each step to approximate feature importance, SAFS [47] ensembles feature selection models within a neural network to identify feature groups and measures the similarity between these groups to select the optimal feature subset. For RL-based methods, MARLFS [23] is a multi-agent system in which the number of agents matches the number of features, and RLAS [48] synergizes median-initialized filters, Q-learning-based module selection, and importance-aware random grouping.

4.4 Hyperparameter Settings and Reproducibility

We performed 200 epochs for HRLFS to explore the feature space and an additional 200 epochs to optimize the agents, with the memory size set to 400. Follow the hyperparameter and ablation experiment results, The policy model of all agents are Actor-Critic, where the actor and critic models are both implemented as two-layer neural networks, with (64, 8) as hidden size. We employed Adam to optimize actor and critic models with a minibatch size of 32. Following the existing research [49], to ensure convergence, the critic's learning rate should be around ten times greater than that for the actor. So the learning rates of the actor and critic models were 0.001 and 0.01, respectively. The hyperparameter γ in Equation 8 as 0.9. We adopted Random Forest as the downstream machine learning model to evaluate the performance of the selected feature subset in each step. We set all features' Gaussian component number k as the maximum number of their BIC search result. For feature metadata embedding, we apply `text-embedding-3-large` [50] based on GPT-4 [51]. We set the hyperparameters α in Equation 6 and λ in Equation 5 as 0.4 and 0.6 based on the results from the hyperparameter study in section 5.8. The hyperparameter settings for the baselines follow their original articles.

4.5 Environmental Settings

The experiments were conducted on an Ubuntu 18.04.6 LTS operating system, equipped with an AMD EPYC 7742 CPU and 4 NVIDIA V100 GPUs, within the Python 3.11.0 environment and using PyTorch 2.1.1.

5 Experimental Results

5.1 Main Comparison

This experiment aims to answer the question: *Is HRLFS capable of effectively refining a superior feature subspace across all domains?* We compared our method with classical and recent approaches across various datasets. Table 1 shows the overall results of HRLFS. We observed that HRLFS outperforms the classical, deep learning-based, and RL-based methods on all tasks. The underlying driver is that, compared to the classical methods (KBest, LASSONet, and MCDM), HRLFS discovers fine-grained differentiation between features through LLMs to construct a more accurate feature subspace beyond numerical statistics, resulting in better performance. Compared to deep learning-based methods (SAFS, CompFS, GAINS), HRLFS manages clustered features using a hierarchical architecture that considers the correlation

Table 1. Overall performance comparison. The best result is highlighted in **bold** for each dataset, and the second-best result is highlighted underlined. #Samp and #Feat denote the number of samples and features.

Dataset	Task	#Samp.	#Feat.	KBest	LASSONet	MCDM	RLAS	MARLFS	SAFS	GAINS	CompFS	Ours
SpectF	C	267	44	79.16	82.11	79.44	80.38	77.01	80.78	80.55	83.69	**87.56**$^{\pm 0.22}$
SVMGuide3	C	1243	21	76.55	77.61	75.68	76.13	75.92	75.53	77.53	72.83	**78.90**$^{\pm 0.45}$
German_Credit	C	1001	24	65.75	58.85	70.49	64.91	66.89	66.89	69.40	68.27	**73.07**$^{\pm 1.13}$
Credit_Default	C	30000	25	80.40	79.86	74.87	80.41	80.11	77.49	80.05	78.15	**80.56**$^{\pm 0.09}$
SpamBase	C	4601	57	90.36	89.16	89.10	91.40	90.04	90.93	91.71	89.70	**92.60**$^{\pm 0.31}$
Megawatt1	C	253	38	84.02	89.59	89.59	83.22	87.64	86.46	89.42	91.83	**92.70**$^{\pm 0.73}$
Ionosphere	C	351	34	91.48	90.17	88.85	94.11	90.18	90.18	92.24	91.08	**94.27**$^{\pm 1.04}$
Mice-Protein	MC	1080	77	82.86	82.71	81.95	77.78	80.56	77.81	81.04	78.96	**83.79**$^{\pm 0.91}$
Coil-20	MC	1440	400	93.92	88.19	96.53	90.97	96.19	94.41	97.22	95.16	**97.56**$^{\pm 0.06}$
MNIST	MC	10000	784	89.67	86.40	91.25	88.30	90.75	86.65	91.40	87.65	**91.75**$^{\pm 0.24}$
Otto	MC	61878	93	72.31	71.65	73.24	72.86	73.74	72.13	72.40	71.97	**74.24**$^{\pm 0.07}$
Jannis	MC	83733	54	65.48	65.78	64.20	57.88	66.70	64.42	65.15	66.61	**67.91**$^{\pm 0.22}$
Cao	MC	4186	13488	82.81	83.65	82.37	90.57	83.71	84.67	82.47	85.57	**89.37**$^{\pm 0.13}$
Han	MC	2746	20670	70.18	72.57	71.33	74.73	74.69	75.79	74.03	73.20	**81.45**$^{\pm 0.21}$
Openml_586	R	1000	25	56.35	56.03	55.82	51.04	55.02	53.15	59.36	55.49	**61.75**$^{\pm 0.71}$
Openml_589	R	1000	25	53.34	52.99	54.12	49.36	52.68	45.51	59.33	48.46	**61.60**$^{\pm 1.01}$
Openml_607	R	1000	50	53.63	53.70	55.37	57.42	54.82	51.82	60.44	53.18	**61.70**$^{\pm 0.69}$
Openml_616	R	500	50	31.16	21.69	28.27	50.38	31.36	46.16	49.54	49.16	**53.90**$^{\pm 0.86}$
Openml_618	R	1000	50	49.21	48.97	48.69	58.57	49.31	45.16	55.68	48.55	**56.40**$^{\pm 0.62}$
Openml_620	R	1000	25	53.00	54.63	55.00	41.63	54.57	50.16	61.51	55.49	**62.78**$^{\pm 0.82}$
Openml_637	R	500	50	23.09	25.42	23.75	42.04	25.96	34.18	39.46	36.17	**40.54**$^{\pm 0.37}$

* We report F1-Score for classification and multi-class classification, 1-RAE for regression.

** The standard deviation is computed based on the results of 5 independent runs

between features from coarse to fine grain, enables better exploration of feature subspace. Compared to RL-based methods (RLAS, MARLFS), agent collaboration is viewed through the hierarchical reinforcement learning architecture in HRLFS, leading to an improvement in performance. Overall, this experiment illustrates that HRLFS is prominent and robust across diverse datasets, emphasizing various tasks, sizes, and fields, underlining its universal applicability for feature subspace exploration.

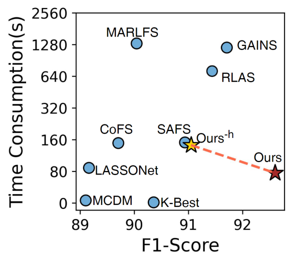
5.2 Space/Time Complexity Analysis

Space/Time Complexity Comparison with OAPF. This experiment aims to answer the question: *Does HRLFS have an advantage in space/time complexity over One-Agent-Per-Feature (OAPF) approaches?* Table 2 compares the number of active agents as well as the overall runtime between the OAPF architecture and HRLFS on four datasets. We observed that the number of active agents for HRLFS is reduced by at least 70.61% across all four datasets. Furthermore, this reduction in active agents directly translates to substantial time savings: on every reported dataset, the overall runtime of HRLFS decreases by more than 30% to 50% compared to OAPF. These empirical results corroborate our complexity proof in Section 3.5, which shows that the expected number of active agents for HRLFS is $O(\log N)$, in contrast to the $O(N)$ active agents required by OAPF. The underlying reason is that HRLFS's hierarchical decision structure, built upon the divide-and-conquer paradigm, achieves superior decision efficiency, significantly reducing the number of active agents and thus both space and time complexity. In summary, by decreasing the number of active agents, the hierarchical architecture not only achieves better space efficiency but also leads to a dramatic reduction in runtime compared to OAPF approaches.

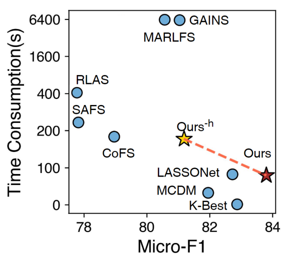
Space/Time Complexity Comparison with Baselines. This experiment aims to answer the question: *Does HRLFS excel in both time and performance?* Figure 6 shows the performance and time consumption of HRLFS compared to other baseline methods and our ablation model HRLFS^{-h} which replaces the multi-agent hierarchical reinforcement learning architecture with One-Agent-Per-Feature (OAPF) architecture. We found that HRLFS outperforms all baselines and

Table 2. Comparison of the average number of active agents between OAPF (one agent per feature) and our model.

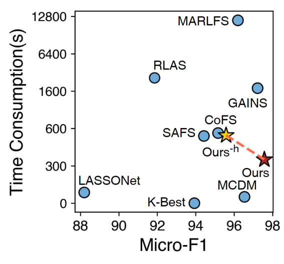
Dataset	Spam Base		Mice-Protein		Coli-20		MNIST	
Method	Ours^{-h}	Ours	Ours^{-h}	Ours	Ours^{-h}	Ours	Ours^{-h}	Ours
Active Agents (Avg.)	56	$16.46^{-70.61\%}$	76	$13.45^{-82.30\%}$	400	$104.39^{-73.90\%}$	784	$156.73^{-80.01\%}$
Time Consumption (s)	146.56	$76.28^{-47.95\%}$	177.77	$79.53^{-55.26\%}$	547.05	$352.86^{-35.49\%}$	2291.25	$1317.28^{-42.50\%}$



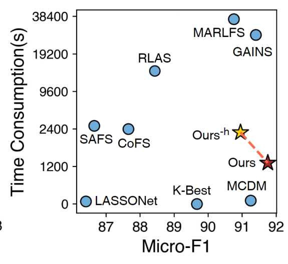
(a) Spam Base



(b) Mice-Protein



(c) Coil-20



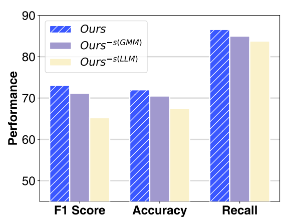
(d) MNIST

Fig. 6. Comparison of HRLFS and baseline methods in downstream task performance and time consumption.

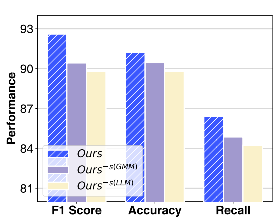
HRLFS^{-h}, with time consumption comparable to traditional statistical methods (KBest, MCDM, and LASSONet) and much superior to methods based on RL (RLAS and MARLFS), deep learning-based methods (SAFS, CoFS, and GAINS), and HRLFS^{-h}. Compared to the deep learning-based methods, our agents only consists of two-layer neural networks, which are less time-consuming compared to numerous-parameters models used by these methods. Compared to RL-based methods, HRLFS applies the hierarchical decision architecture to manage clustered features, greatly reducing the number of active agents, where active agents will make decisions when the rest of the agents remain inactive, enabling more efficient iteration and less time consumption. Further, HRLFS far outperforms HRLFS^{-h} in both performance and time, underlining the fact that the hierarchical decision architecture can capture correlations between features, facilitate agents' collaboration, and reduce the number of active agents. In conclusion, with effective state extraction and hierarchical architecture design, HRLFS offers great advantages in terms of time and performance.

5.3 Ablation Study

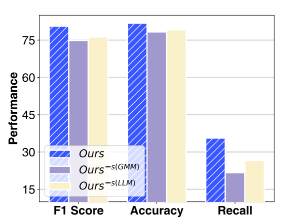
We designed three variants: **HRLFS**$^{-s(GMM)}$ which only utilizes the Gaussian Mixture Model for feature representation; and **HRLFS**$^{-s(LLM)}$, which only employs the Large Language Model for feature representation; **HRLFS**$^{-h}$, which replaces the multi-agent hierarchical reinforcement learning architecture with OAPF architecture.



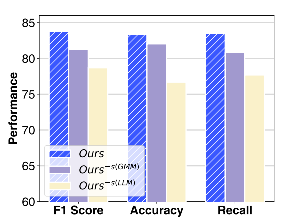
(a) German Credit



(b) Spam Base



(c) Credit_Default



(d) Mice-Protein

Fig. 7. The impact of different state representation methods.

Impact of Hybrid State Representation. Figure 7 shows the comparison results of feature representation variants on two classification tasks and two regression tasks. We find that HRLFS is overall better than HRLFS$^{-s(GMM)}$ and HRLFS$^{-s(LLM)}$. The underlying driver for this observation is that, compared to HRLFS$^{-s(GMM)}$, the hybrid state representation utilizes the power of LLMs to interpret and extract information from feature metadata, providing valuable insights for hierarchical agents to optimize and explore feature subspace, leading to a better feature selection. Compared to HRLFS$^{-s(LLM)}$, the hybrid state representation component integrates GMM statistical data, providing a better description of the distribution of each feature, resulting in a performance improvement. In conclusion, this experiment illustrates that the hybrid state representation component incorporates meaningful information from feature names and statistical data, playing an important role in HRLFS.

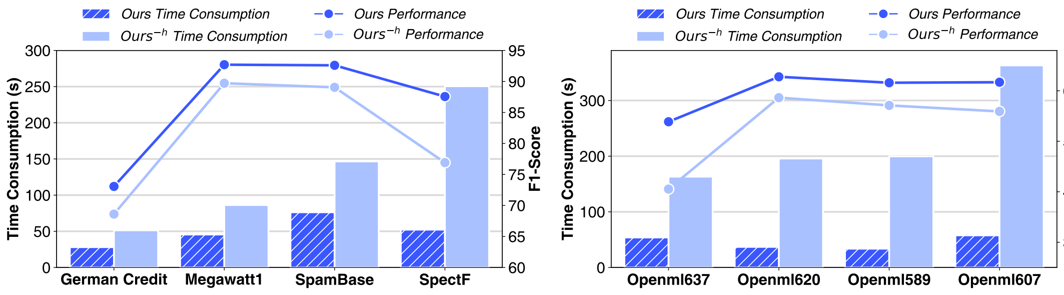
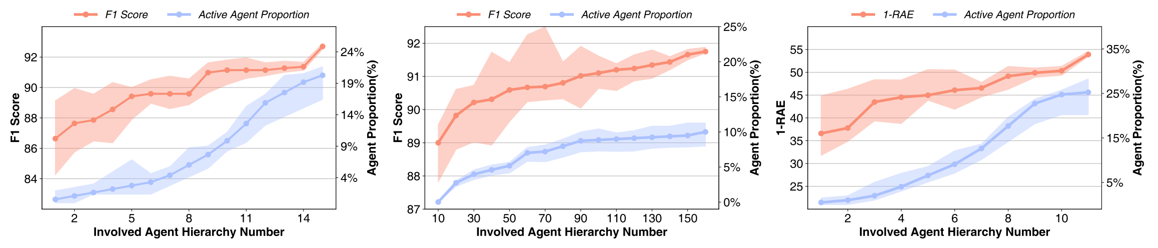


Fig. 8. The impact of multi-agent hierarchical reinforcement learning architecture.

Impact of HRL Architecture. Figure 8 compares HRL architecture variants on two classification tasks and two regression tasks. We can observe that HRLFS reduces time consumption while outperforming HRLFS^{-h} across performance. This implies that the hierarchical agent structure greatly reduces the number of active agents, thus reducing the consumed time. Meanwhile, this hierarchical architecture provides a variable granularity toward feature subspace exploration, bringing about improved performance.



(a) Megawatt1 (b) MNIST (c) OpenML_616

Fig. 9. The performance on downstream tasks and the proportion of active agents with changes in the granularity of the agent hierarchy involved.

5.4 Analysis of the Multi-Agent Hierarchy Architecture Setting

This experiment aims to answer the question: *How does a finer or coarser-grained Agent Hierarchy setting impact the overall system?* In Algorithm 1, features with similar semantic or mathematical distributions are controlled by the same agent, forming a hierarchical tree structure. In this experiment, we selected one dataset from each of the three tasks—Megawatt1, MNIST, and OpenML_616—and varied the involved agent hierarchy number. A smaller value indicates that fewer agent layers participate in feature selection, leading to a coarser-grained decision-making process.

From the result in Figure 9, we can observe that increasing the number of hierarchy levels leads to consistent improvement of performance and stabilization, as evidenced by higher scores and reduced standard deviation. The underlying driver of this behavior is the finer exploration of the feature space, facilitated by the increased number of agents in more granular hierarchies. Further, we note that coarser hierarchies, which activate fewer agents, result in lower performance due to a less detailed exploration of relevant features. This suggests that a more granular hierarchy enables better agent coordination and decision-making, improving feature selection efficiency. However, this improvement comes at the cost of higher computational complexity and more active agents, indicating a trade-off between agent involvement and system efficiency.

5.5 Analysis of the Hybrid State Extraction Method

This experiment aims to answer the question: *Is combining GMM and GPT-4 for feature state extraction optimal?* We

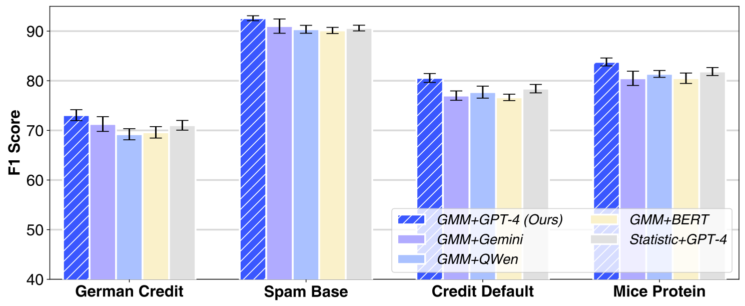


Fig. 10. Comparison of different feature state extraction models.

compared various feature distribution extraction methods and semantic extraction models with different parameter scales in several datasets, shown in Figure 10. **Statistic+GPT-4** applies the mean, maximum, minimum, variance, the first quartile, the second quartile, and the third quartile to extract feature distribution and GPT-4 to leverage metadata. **GMM+BERT, GMM+QWen, GMM+Gemini** use GMM to pick up statistic and leverage semantic information using BERT [52], `gte-Qwen2-7B-instruct` [53], `gemini-embedding-exp` [54], respectively. By comparing **Statistic+GPT-4** and **GMM+GPT-4**, we can observe that the feature distribution obtained using GMM has better results compared to statistics. The underlying driver of this behavior is that statistics such as mean, maximum, minimum, etc., can only provide a simple description of the data and cannot capture the inner structure of the data, whereas GMM can approximate an arbitrarily shaped probability distribution by a linear combination of multiple Gaussian distributions, which better captures this complexity and leads to better results. By comparing **GMM+BERT, GMM+QWen, GMM+Gemini, GMM+GPT-4**, we find that GPT-4 extracts semantic information with better performance than other models. This implies that compared to BERT and QWen, the model architecture and scale of GPT-4 are larger and capable of capturing more complex semantic relationships and contextual information. Compared to Gemini, GPT-4's data screening and cleaning criteria reduce errors and biases in training data, further improving the accuracy and reliability of semantic information extraction. In summary, as GMM and GPT-4 can accurately extract feature distribution and semantic information from data, they are the best choice for feature state extraction.

5.6 Analysis of the Feature Clustering Algorithm Selection

This experiment aims to answer the question: *Whether the H-clustering algorithm has superior performance compared to other clustering methods in the feature subspace exploration task?* We compared the H-clustering (Ours) with KMeans [55], DBSCAN [56] and SpectralClustering [57] in four datasets with different quantitative numbers of features in Figure 11.

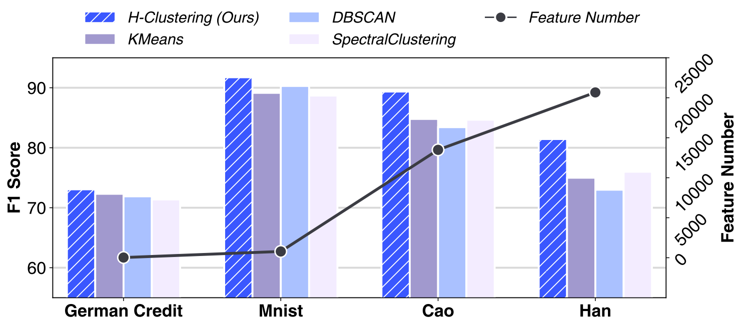
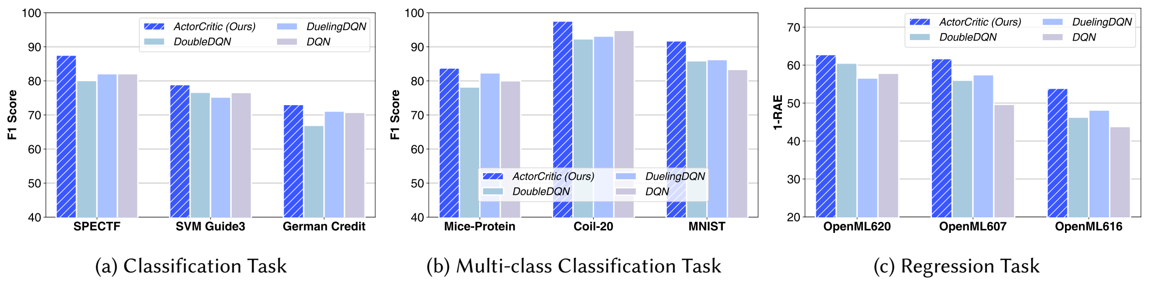


Fig. 11. The impact of different clustering algorithms.

We can observe that the H-clustering outperforms all clustering algorithms, and the advantages of H-clustering become more apparent as the number of features increases. The underlying driver of this behavior is that, compared to ordinary clustering algorithms, H-clustering aggregates features from fine-grained to coarse-grained, constructs a hierarchical decision architecture, and considers the feature selection problem from more granularities, leading to better performance. At the same time, this coarse-to-fine decision-making approach decomposes an elaborate task into several relatively simple tasks, which naturally has the advantage of dealing with complex problems, so H-clustering is more advantageous for tasks with a larger number of features. In summary, H-clustering decides feature selection problems at different granularities, has better performance, can handle more complex problems, thus outperforming other methods.



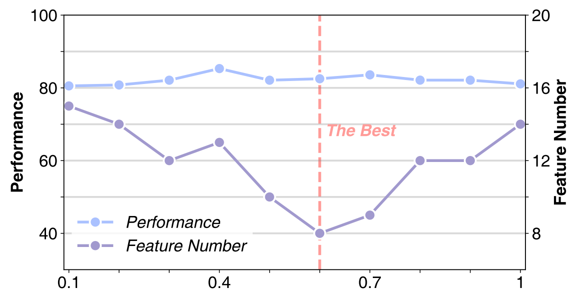
(a) Classification Task (b) Multi-class Classification Task (c) Regression Task

Fig. 12. The impact of different policy model.

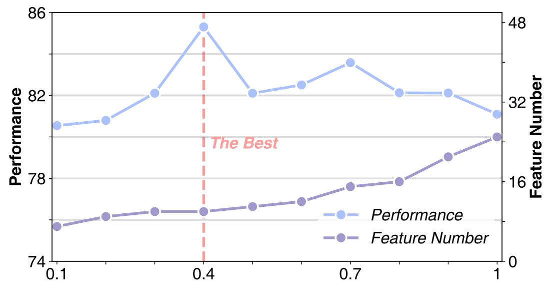
5.7 Analysis of Reinforcement Learning Backend Selection

This experiment aims to answer the question: *Whether the policy model used in HRLFS is superior to other existing models?* We compared ActorCritic [34] (Ours) with DQN [58], Double DQN [59] and Dueling DQN [60] on binary classification, multi classification and regression tasks in Figure 12. We find that ActorCritic outperforms other methods in all tasks. This implies that, compared to methods such as DQN, which rely on indirect updating of Q-values, ActorCritic combines policy gradient and value function estimation and is able to directly optimize the policy, reducing variance and improving learning efficiency, resulting in better performance on all tasks. In conclusion, ActorCritic is a suitable choice for the policy model due to its higher efficiency and stability of strategy optimization.

5.8 Hyperparameter Study



(a) α keeps 0.5, λ varies from 0.1 to 1.



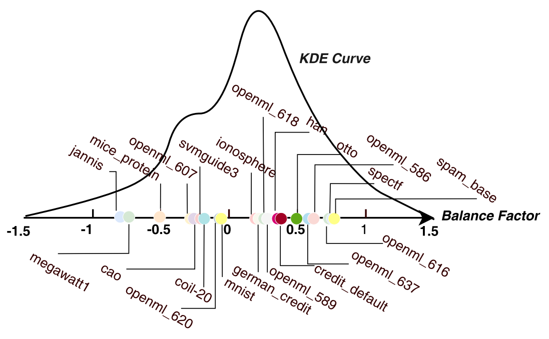
(b) λ keeps 0.5, α varies from 0.1 to 1.

Fig. 13. The impact of hyperparameter λ and α.

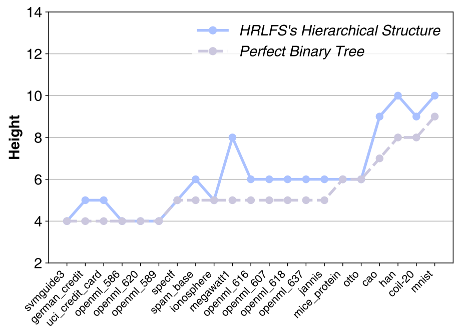
In this section, we analysis two essential hyperparameter of HRLFS, i.e., λ (Equation 5) and α (Equation 6). A higher λ encourages the selection of fewer features, while an increase α prioritizes performance over feature compactness. We performed experiments adjusting λ and α within the range of 0.1 to 1.0 on the SpectF dataset and report the results in Figure 13. Analysis reveals that changes in λ have minimal impact on model performance but influence the number of selected features, which initially decreases and then increases as λ rises. This pattern indicates that higher λ first suppresses selected feature size, but its further increases reduce fluctuations in r_t^q, allowing more features to be included. Regarding α, increasing its value initially enhances model performance by emphasizing performance objectives. However, beyond a certain point, performance declines as excessive feature selection introduces redundancy. The number of selected features rises with α, reflecting the decrease in suppression in the reward function. These findings demonstrate that λ and α are crucial in balancing feature selection and model performance. Consequently, we selected $\lambda = 0.6$ and $\alpha = 0.4$ to balance performance and feature compactness. The experimental results confirm that λ and α modulate the reward function in alignment with our objectives.

5.9 Empirical Validation of the Hierarchical Assumption

Our theoretical analysis assumes that the constructed hierarchical structure approximates a perfect binary tree. To assess this assumption empirically, we evaluated the balance factor and height of the learned hierarchies across all 21 datasets. The balance factor is defined as the average difference in height between the left and right subtrees for all nodes [61]. In a perfect binary tree, this difference never exceeds 1, and the tree height is uniquely determined by the number of nodes. As shown in Figure 14, the absolute balance factor remains below 1 for all datasets, and over 80% of them fall within 0.5. In addition, for 18 datasets, the height of our constructed hierarchy differs from that of the corresponding perfect binary tree by no more than 2 levels. This strong similarity arises because our Agent Hierarchy Construction algorithm clusters agent sets with comparable sizes at earlier stages, naturally promoting balanced splits. In summary, both balance and height analyses confirm that the constructed hierarchies empirically resemble perfect binary trees, thereby supporting the validity of our assumption.



(a) Balance Factor.



(b) Total Tree Height.

Fig. 14. Empirical study of HRLFS's hierarchical structure compared with the perfect binary tree.

6 Related Work

Feature Selection. Feature selection methods are broadly categorized into three approaches: filter-based, wrapper-based, and embedded-based methods [1]. Filter-based methods [4, 5] evaluate the statistical relationship between each feature and the prediction target or among features themselves. Features with measurements below a certain threshold are excluded. These methods are computationally efficient but do not account for interactions between features, which can result in redundant or suboptimal feature subsets. Wrapper-based methods[7, 8, 62, 63] assess feature subsets by iteratively training and evaluating a specific machine learning model. However, as the number of features grows, the search space becomes increasingly large, rendering these methods impractical for large-scale datasets [26, 64]. Embedded methods[9–11] integrate feature selection directly into the model training process. This approach allows for the simultaneous optimization of feature relevance and model performance, facilitating the efficient identification of relevant features[65–67]. However, embedded methods are often tied to specific types of models, limiting their flexibility and transferability to different downstream tasks. Additionally, some embedded techniques can still be computationally demanding when applied to large and complex datasets.

Reinforcement Learning. Reinforcement Learning is a machine learning paradigm in which an agent learns to make decisions by interacting with an environment based on its policy [68]. RL has been successfully applied to various research areas [69–72]. However, classical RL algorithms often struggle with long-range decision-making tasks due to limitations in scalability and efficiency [73]. To overcome these challenges, Hierarchical Reinforcement Learning (HRL) has been developed to decomposes the decision-making process into multiple levels of abstraction, allowing for more efficient learning and planning in complex tasks [74, 75]. Building on HRL, Multi-Agent Hierarchical Reinforcement Learning focuses on the coordination and collaboration among multiple HRL agents [73, 76–79]. For example, Chakravorty et al.[80] investigated the learning of temporal abstractions in cooperative multi-agent systems, while Yang et al.[81] demonstrated the integration of cooperative MARL algorithms for training high-level policies alongside single-agent RL for low-level skill acquisition. These advancements in MAHRL inspire our approach, where we employ a divide-and-conquer strategy within a multi-agent hierarchical reinforcement learning framework.

RL-based Feature Selection. Reinforcement learning has been widely employed to recast wrapper-based feature selection as an objective-directed sequential decision problem, enabling iterative subspace exploration. In the early stage, Liu et al. [22] proposed a single-agent-all-feature framework; however, it suffers from poor scalability and high time complexity ($O(N)$ on high-dimensional data). Liu et al. [23] introduced a multi-agent reinforcement learning (MARL) architectures that assign one agent per feature, but the resulting quadratic growth in agent count quickly becomes prohibitive. Group-based and interaction-wise agents[25, 26] reduce the decision horizon by clustering features with simple statistical criteria, yet they ignore semantic and contextual relationships, leading to redundant or sub-optimal selections [27, 28]. Inspired by hierarchical RL's success in long-horizon tasks [74, 75], we adopt a divide-and-conquer multi-agent hierarchical RL framework that first semantically clusters features and then assigns cooperative hierarchical agents to each cluster, simultaneously improving scalability, efficiency, and downstream performance.

7 Conclusion and Remark

This paper studies the challenges of exploring feature subspaces from complex datasets. Specifically, we reformulated the problem of feature selection as a multi-agent hierarchical reinforcement learning framework through the hybrid feature state extraction (comprehend), the hierarchical agent initialization (divide), and the exploration and optimization (conquer). We carried out comprehensive evaluations of HRLFS, illustrating its efficient, superior, and robust performance in various data sets from different tasks and fields.

8 Limitation and Future Work

HRLFS relies on metadata and feature names to generate feature descriptions, which makes it vulnerable to adversarial or maliciously crafted inputs that could corrupt the generated descriptions. Moreover, the current design and evaluation are limited to tabular data. Extending the framework to high-dimensional modalities such as time series, images, or audio remains an open challenge. Finally, the present formulation focuses on standard supervised learning tasks (classification and regression). Adapting HRLFS to more diverse downstream objectives—including survival analysis, anomaly detection, and causal inference—represents a promising direction for future research.

References

[1] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM computing surveys (CSUR)*, vol. 50, no. 6, pp. 1–45, 2017.

[2] D. Wang, Y. Huang, W. Ying, H. Bai, N. Gong, X. Wang, S. Dong, T. Zhe, K. Liu, M. Xiao *et al.*, "Towards data-centric ai: A comprehensive survey of traditional, reinforcement, and generative approaches for tabular data transformation," *arXiv preprint arXiv:2501.10555*, 2025.

[3] Y. Li, T. Li, and H. Liu, "Recent advances in feature selection and its applications," *Knowledge and Information Systems*, vol. 53, pp. 551–577, 2017.

[4] J. Biesiada and W. Duch, "Feature selection for high-dimensional data—a pearson redundancy based filter," in *Computer recognition systems 2*. Springer, 2008, pp. 242–249.

[5] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, 2003, pp. 856–863.

[6] X.-F. Song, Y. Zhang, D.-W. Gong, and X.-Z. Gao, "A fast hybrid feature selection based on correlation-guided clustering and particle swarm optimization for high-dimensional data," *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 9573–9586, 2021.

[7] G. Stein, B. Chen, A. S. Wu, and K. A. Hua, "Decision tree classifier for network intrusion detection with ga-based feature selection," in *Proceedings of the 43rd annual Southeast regional conference-Volume 2*, 2005, pp. 136–141.

[8] I. A. Gheyas and L. S. Smith, "Feature subset selection in large dimensionality domains," *Pattern recognition*, vol. 43, no. 1, pp. 5–13, 2010.

[9] V. C. Dinh and L. S. Ho, "Consistent feature selection for analytic deep neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2420–2431, 2020.

[10] Y. Li, C.-Y. Chen, and W. W. Wasserman, "Deep feature selection: theory and application to identify enhancers and promoters," *Journal of Computational Biology*, vol. 23, no. 5, pp. 322–336, 2016.

[11] I. Lemhadri, F. Ruan, and R. Tibshirani, "Lassonet: Neural networks with feature sparsity," in *International conference on artificial intelligence and statistics*. PMLR, 2021, pp. 10–18.

[12] C. Qin, X. Chen, C. Wang, P. Wu, X. Chen, Y. Cheng, J. Zhao, M. Xiao, X. Dong, Q. Long *et al.*, "Scihorizon: Benchmarking ai-for-science readiness from scientific data to large language models," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 2025, pp. 5754–5765.

[13] Y. Liu, L. Yang, M. Meskini, A. Goel, M. Opperman, S. S. Shyamal, A. Manaithiya, M. Xiao, R. Ni, Y. An *et al.*, "Gut microbiota and tuberculosis," *iMeta*, p. e70054.

[14] M. Xiao, D. Wang, M. Wu, P. Wang, Y. Zhou, and Y. Fu, "Beyond discrete selection: Continuous embedding space optimization for generative feature selection," in *2023 IEEE International Conference on Data Mining (ICDM).* IEEE, 2023, pp. 688–697.

[15] M. Xiao, W. Zhang, X. Huang, H. Zhu, M. Wu, X. Li, and Y. Zhou, "Knowledge-guided gene panel selection for label-free single-cell rna-seq data: A reinforcement learning perspective," *IEEE Transactions on Computational Biology and Bioinformatics*, pp. 1–14, 2025.

[16] W. Ying, D. Wang, X. Hu, J. Qiu, J. Park, and Y. Fu, "Revolutionizing biomarker discovery: Leveraging generative ai for bio-knowledge-embedded continuous space exploration," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024, pp. 5046–5053.

[17] W. Liang, G. A. Tadesse, D. Ho, L. Fei-Fei, M. Zaharia, C. Zhang, and J. Zou, "Advances, challenges and opportunities in creating data for trustworthy ai," *Nature Machine Intelligence*, vol. 4, no. 8, pp. 669–677, 2022.

[18] X. Cai, C. Wang, Q. Long, Y. Zhou, and M. Xiao, "Knowledge hierarchy guided biological-medical dataset distillation for domain llm training," *arXiv preprint arXiv:2501.15108*, 2025.

[19] X. Huang, D. Wang, Z. Ning, Z. Qiao, Q. Long, H. Zhu, Y. Du, M. Wu, Y. Zhou, and M. Xiao, "Collaborative multi-agent reinforcement learning for automated feature transformation with graph-driven path optimization," *arXiv preprint arXiv:2504.17355*, 2025.

[20] M. Xiao, X. Cai, Q. Long, C. Wang, Y. Zhou, and H. Zhu, "m-kailin: Knowledge-driven agentic scientific corpus distillation framework for biomedical large language models training," *arXiv preprint arXiv:2504.19565*, 2025.

[21] Y. Fu, D. Wang, H. Xiong, and K. Liu, "Tabular data-centric ai: Challenges, techniques and future perspectives," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024, pp. 5522–5525.

[22] K. Liu, P. Wang, D. Wang, W. Du, D. O. Wu, and Y. Fu, "Efficient reinforced feature selection via early stopping traverse strategy," in *2021 IEEE International Conference on Data Mining (ICDM).* IEEE, 2021, pp. 399–408.

[23] K. Liu, Y. Fu, P. Wang, L. Wu, R. Bo, and X. Li, "Automating feature subspace exploration via multi-agent reinforcement learning," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 207–215.

[24] K. Zhang, Z. Yang, and T. Başar, *Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms.* Cham: Springer International Publishing, 2021, pp. 321–384. [Online]. Available: https://doi.org/10.1007/978-3-030-60990-0_12

[25] W. Fan, K. Liu, H. Liu, A. Hariri, D. Dou, and Y. Fu, "Autogfs: Automated group-based feature selection via interactive reinforcement learning," in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM).* SIAM, 2021, pp. 342–350.

[26] W. Fan, K. Liu, H. Liu, P. Wang, Y. Ge, and Y. Fu, "Autofs: Automated feature selection via diversity-aware interactive reinforcement learning," in *2020 IEEE International Conference on Data Mining (ICDM).* IEEE, 2020, pp. 1008–1013.

[27] D. Li, Z. Tan, and H. Liu, "Exploring large language models for feature selection: A data-centric perspective," *arXiv preprint arXiv:2408.12025*, 2024.

[28] R. Moraffah, P. Sheth, S. Vishnubhatla, and H. Liu, "Causal feature selection for responsible machine learning," *arXiv preprint arXiv:2402.02696*, 2024.

[29] C. Higuera, K. J. Gardiner, and K. J. Cios, "Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome," *PloS one*, vol. 10, no. 6, p. e0129126, 2015.

[30] E. A. Feinberg and A. Shwartz, *Handbook of Markov decision processes: methods and applications.* Springer Science & Business Media, 2012, vol. 40.

[31] G. H. Dunteman, *Principal components analysis.* Sage, 1989, vol. 69.

[32] J. H. Ward Jr, "Hierarchical grouping to optimize an objective function," *Journal of the American statistical association*, vol. 58, no. 301, pp. 236–244, 1963.

[33] T. Schaul, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[34] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[35] S. M. Kakade, "A natural policy gradient," *Advances in neural information processing systems*, vol. 14, 2001.

[36] V. Cherepanova, R. Levin, G. Somepalli, J. Geiping, C. B. Bruss, A. G. Wilson, T. Goldstein, and M. Goldblum, "A performance-driven benchmark for feature selection in tabular deep learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 41 956–41 979, 2023.

[37] R. Edgar, M. Domrachev, and A. E. Lash, "Gene expression omnibus: Ncbi gene expression and hybridization array data repository," *Nucleic acids research*, vol. 30, no. 1, pp. 207–210, 2002.

[38] Public, "Uci dataset download," [EB/OL], 2022, https://archive.ics.uci.edu/.

[39] J. Howard, "Kaggle dataset download," [EB/OL], 2022, https://www.kaggle.com/datasets.

[40] Public, "Openml dataset download," [EB/OL], 2022, https://www.openml.org.

[41] L. Chih-Jen, "Libsvm dataset download," [EB/OL], 2022, https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

[42] D. Wang, Y. Fu, K. Liu, X. Li, and Y. 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*, 2022, pp. 1826–1834.

[43] M. Xiao, D. Wang, M. Wu, K. Liu, H. Xiong, Y. Zhou, and Y. Fu, "Traceable group-wise self-optimizing feature transformation learning: A dual optimization perspective," *ACM Transactions on Knowledge Discovery from Data*, vol. 18, no. 4, pp. 1–22, 2024.

[44] Y. Yang, J. O. Pedersen *et al.*, "A comparative study on feature selection in text categorization," in *icml*, vol. 97. Citeseer, 1997, p. 35.

[45] A. Hashemi, M. B. Dowlatshahi, and H. Nezamabadi-pour, "Ensemble of feature selection algorithms: a multi-criteria decision-making approach," *International Journal of Machine Learning and Cybernetics*, vol. 13, no. 1, pp. 49–69, 2022.

[46] T. Yasuda, M. Bateni, L. Chen, M. Fahrbach, G. Fu, and V. Mirrokni, "Sequential attention for feature selection," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=TTLLGx3eet

[47] F. Imrie, A. Norcliffe, P. Liò, and M. van der Schaar, "Composite feature selection using deep ensembles," *Advances in Neural Information Processing Systems*, vol. 35, pp. 36 142–36 160, 2022.

[48] H. Zhang, X. Yue, and X. Gao, "Reinforcement learning guided auto-select optimization algorithm for feature selection," *Expert Systems with Applications*, vol. 268, p. 126320, 2025.

[49] Y. F. Wu, W. Zhang, P. Xu, and Q. Gu, "A finite-time analysis of two time-scale actor-critic methods," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 617–17 628, 2020.

[50] OpenAI, "New embedding models and api updates," https://openai.com/index/new-embedding-models-and-api-updates/, accessed: 2025-04-17.

[51] ——, "Gpt-4 is openai's most advanced system, producing safer and more useful responses," https://openai.com/blog/gpt-4/, accessed: 2025-04-17.

[52] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[53] Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, and M. Zhang, "Towards general text embeddings with multi-stage contrastive learning," *arXiv preprint arXiv:2308.03281*, 2023.

[54] J. Lee, F. Chen, S. Dua, D. Cer, M. Shanbhogue, I. Naim, G. H. Ábrego, Z. Li, K. Chen, H. S. Vera *et al.*, "Gemini embedding: Generalizable embeddings from gemini," *arXiv preprint arXiv:2503.07891*, 2025.

[55] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, vol. 5. University of California press, 1967, pp. 281–298.

[56] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[57] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, pp. 395–416, 2007.

[58] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[59] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[60] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[61] K. S. Larsen, "Avl trees with relaxed balance," in *Proceedings of 8th International Parallel Processing Symposium*. IEEE, 1994, pp. 888–893.

[62] X. Huang, L. Zhang, B. Wang, F. Li, and Z. Zhang, "Feature clustering based support vector machine recursive feature elimination for gene selection," *Applied Intelligence*, vol. 48, pp. 594–607, 2018.

[63] J. Chen, M. Stern, M. J. Wainwright, and M. I. Jordan, "Kernel feature selection via conditional covariance minimization," *Advances in neural information processing systems*, vol. 30, 2017.

[64] M. Xiao, D. Wang, M. Wu, Z. Qiao, P. Wang, K. Liu, Y. Zhou, and Y. Fu, "Traceable automatic feature transformation via cascading actor-critic agents," in *Proceedings of the 2023 SIAM International Conference on Data Mining (SDM)*. SIAM, 2023, pp. 775–783.

[65] Y. Lu, Y. Fan, J. Lv, and W. Stafford Noble, "Deeppink: reproducible feature selection in deep neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[66] K. Koyama, K. Kiritoshi, T. Okawachi, and T. Izumitani, "Effective nonlinear feature selection method based on hsic lasso and with variational inference," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022, pp. 10 407–10 421.

[67] A. Kumagai, T. Iwata, Y. Ida, and Y. Fujiwara, "Few-shot learning for feature selection with hilbert-schmidt independence criterion," *Advances in Neural Information Processing Systems*, vol. 35, pp. 9577–9590, 2022.

[68] R. S. Sutton, "Reinforcement learning: An introduction," *A Bradford Book*, 2018.

[69] A. Sarkar, N. Jacobs, and Y. Vorobeychik, "A partially-supervised reinforcement learning framework for visual active search," *Advances in Neural Information Processing Systems*, vol. 36, pp. 12 245–12 270, 2023.

[70] O. et al., "Gpt-4 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2303.08774

[71] D. Zhang, L. Chen, S. Zhang, H. Xu, Z. Zhao, and K. Yu, "Large language models are semi-parametric reinforcement learning agents," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[72] T. He, X. Huang, Y. Du, Q. Long, Z. Qiao, M. Wu, Y. Fu, Y. Zhou, and M. Xiao, "Fastft: Accelerating reinforced feature transformation via advanced exploration strategies," in *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE Computer Society, 2025, pp. 4120–4133.

[73] S. Pateria, B. Subagdja, A.-h. Tan, and C. Quek, "Hierarchical reinforcement learning: A comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–35, 2021.

[74] B. Hengst, "Hierarchical reinforcement learning," *Encyclopedia of machine learning*, pp. 495–502, 2011.

[75] V. H. Wang, T. Wang, W. Yang, J. Kämäräinen, and J. Pajarinen, "Probabilistic subgoal representations for hierarchical reinforcement learning," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024.* OpenReview.net, 2024. [Online]. Available: https://openreview.net/forum?id=b6AwZauZPV

[76] M. Liu, C. Amato, E. Anesta, J. Griffith, and J. How, "Learning for decentralized control of multiagent systems in large, partially-observable stochastic environments," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, 2016.

[77] M. Ghavamzadeh, S. Mahadevan, and R. Makar, "Hierarchical multi-agent reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 13, pp. 197–229, 2006.

[78] A. Sivagnanam, A. Pettet, H. Lee, A. Mukhopadhyay, A. Dubey, and A. Laszka, "Multi-agent reinforcement learning with hierarchical coordination for emergency responder stationing," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024.* OpenReview.net, 2024. [Online]. Available: https://openreview.net/forum?id=TTZXl9WYFF

[79] Y. Ma, J. Hao, H. Liang, and C. Xiao, "Rethinking decision transformer via hierarchical reinforcement learning," in *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024.* OpenReview.net, 2024. [Online]. Available: https://openreview.net/forum?id=WsM4TVsZpJ

[80] J. Chakravorty, N. Ward, J. Roy, M. Chevalier-Boisvert, S. Basu, A. Lupu, and D. Precup, "Option-critic in cooperative multi-agent systems," *arXiv preprint arXiv:1911.12825*, 2019.

[81] J. Yang, I. Borovikov, and H. Zha, "Hierarchical cooperative multi-agent reinforcement learning with skill discovery," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 1566–1574.