# Neural Architecture Search Based on Bipartite Graphs for Text Classification

Xueming Yan, *Member, IEEE*, Han Huang, *Senior Member, IEEE*, Yaochu Jin, *Fellow, IEEE*, Zilong Wang, and Zhifeng Hao, *Senior Member, IEEE*

***Abstract*— Neural architecture search (NAS) is crucial for text representation in natural language processing (NLP); however, much less work on NAS for text classification has been proposed compared with NAS for computer vision. Similar to NAS for vision tasks, most existing work rely on a manually designed search space defined by a directed acyclic graph (DAG), resulting in limited generalization capability and high computational complexity. In text classification, the topological order of the NAS operators is essential for enhancing generalization, which cannot be accurately represented by a DAG. To address this issue, we propose a bipartite graph-based NAS (BGNAS) for text classification, which converts a DAG into a dual graph and then into a bipartite graph. This transformation makes it possible to accurately capture the topological order using multi-bigraph matching. In addition, we formulate NAS as a problem of identifying the lower bound of a submodular function, theoretically ensuring that optimal architectures in a bipartite graph-based search space can be identified using fewer search operators. Reduction of the search space is achieved by eliminating ineffective associated matching rules among search operators with a pruning strategy. As a result, the bipartite graph-based search space becomes more compact and less dependent on complex contextual semantics of text data. Experimental results on public benchmark problems demonstrate that BGNAS achieves better performance than the state-of-the-art NAS algorithms and is computationally more efficient. We also demonstrate that the bipartite graph search space can more effectively capture contextual semantics, thereby enhancing the generalization capability.**

***Index Terms*— Associated matching rules, bipartite graph, neural architecture search (NAS), text classification.**

Received 9 February 2024; revised 28 July 2024 and 4 October 2024; accepted 3 December 2024. This work was supported in part by the International Collaboration Fund for Creative Research Teams (ICFCRT) of NSFC under Grant W2441019; in part by the National Natural Science Foundation of China under Grant 62476163, Grant 62136003, and Grant 62276103; and in part by the Innovation Team Project of General Colleges and Universities in Guangdong Province under Grant 2023KCXTD002. *(Corresponding author: Han Huang.)*

Xueming Yan is with the School of Information Science and Technology, Guangdong University of Foreign Studies, Guangzhou 510006, China (e-mail: yanxm@gdufs.edu.cn).

Han Huang is with the School of Software Engineering, South China University of Technology, Guangzhou 510006, China, and also with the Key Laboratory of Big Data and Intelligent Robot (SCUT), MOE of China, Guangzhou 510006, China (e-mail: hhan@scut.edu.cn).

Yaochu Jin is with the School of Engineering, Westlake University, Hangzhou 310030, China (e-mail: jinyaochu@westlake.edu.cn).

Zilong Wang is with the School of Software Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: zilongwangi@163.com).

Zhifeng Hao is with the College of Mathematics and Computer Science, Shantou University, Guangdong 515063, China (e-mail: haozhifeng@stu.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TNNLS.2024.3514708, provided by the authors.

Digital Object Identifier 10.1109/TNNLS.2024.3514708

## I. Introduction

NEURAL architecture search (NAS) has demonstrated remarkable success in a variety of real-world applications, including computer vision [1], speech recognition [2], and natural language processing (NLP) [3]. The promising performance of NAS has been thoroughly documented due to its ability to discover highly efficient and effective neural network architectures [1], [4]. As a bilevel optimization problem, the mathematical formulation of NAS [5] can be defined with the weights $w_A$ of the potential neural architecture $A$ as follows:

$$\min_{A\in\Omega} \mathcal{L}_{\text{val}}(w^*, A) \tag{1}$$

$$\text{s.t. } w^* = \arg\min_{w_A} \mathcal{L}_{\text{train}}(w_A, A) \tag{2}$$

where $\mathcal{L}_{\text{val}}$ in (1) and $\mathcal{L}_{\text{train}}$ in (2) represent the training and validation losses, which are determined by both the architecture $A$ and the weights $w$ of the neural network, respectively. $w^*$ represents the weights corresponding to the optimal architecture on the training set. The primary objective of NAS is to enhance the performance of neural networks by identifying an optimal neural architecture that is well-suited to specific tasks.

In principle, NAS can also be considered a black-box optimization problem [6], [7], due to the lack of explicit knowledge about the relationship between the objectives and neural network architectures. Furthermore, NAS typically involves exploring a high-dimensional search space, especially when conducting the search using simplistic methods such as grid search or random search [8], [9]. Therefore, various search strategies are proposed for navigating through the large search space more efficiently to find the optimal neural network architecture, including gradient-based methods [4], [10], reinforcement learning (RL) [11], [12], and evolutionary algorithms (EAs) [13], [14]. However, the computational demands of exploring the search strategies across the entire search space become prohibitive as the network size and complexity increase [15]. Therefore, some existing NAS approaches concentrate on the manual design of a constrained search space to mitigate dimensionality and alleviate computational complexity [16].

The design of NAS for search space is crucial in finding the best architectures on specific tasks [17], [18]. Several

researchers [19], [20] have explored the design of different search spaces of NAS to improve search efficiency and performance, resulting in two main design approaches, namely, micro search and macro search. The micro search focuses on the design of individual building blocks or cells within a neural network architecture, while the macro search deals with the higher level architectural choices that define the overall structure of a neural network. The choice between macro search and micro search depends on specific features and characteristics of the related task [21].

More recently, it has been found that the NAS-based approaches have also demonstrated superiority in dealing with text classification [22], [23]. For example, Wang et al. [12] demonstrate that effective design of the macro search space for text representation can significantly enhance the efficiency and generalization. Nevertheless, most current work on the search space of NAS almost entirely depends on the incorporation of human prior knowledge for text representation, so it is necessary to explore the subset of possible architectures from the search space for lower computing resources and time consumption [24], [25]. Therefore, some attempts are made [20], [26], [27] to devise strategies for streamlining the search space in efficient NAS. These strategies often involve discarding unpromising candidate operators to improve the overall efficiency of the search process [28]. However, the current shrinking approaches are often intricately linked with the search algorithm, as noted by Hu et al. [29]. This tight coupling poses a challenge in isolating a smaller set of effective candidate operators tailed to a given task. One reason is that these approaches neglect the order of candidate operators and the combination of different candidate operators, which is crucial for the performance of text representation.

In this study, we propose a bipartite graph-based NAS (BGNAS) approach for text classification. Unlike the directed acyclic graph (DAG) used in many existing NAS approaches, the bipartite graph used in BGNAS represents each operator as a node, and the sequential relationships between the operators are represented using edges. Consequently, the proposed approach can efficiently explore the topological orders of the candidate operators for text classification. To ensure that NAS can theoretically find the optimal network architecture in the reduced bipartite graph-based search space, we view NAS as the problem of establishing a lower bound for a submodular function. Subsequently, we proceed to approximate this submodular function using the matching rules associated with the bipartite graph-based search space. It is worth noting that the reduction of the bipartite graph-based search space is adaptively evolved based on a set of learned bipartite graph matching associated rules.

The main contributions of this article are summarized below.

1) We propose a bipartite graph representation of the NAS search space. In contrast to the existing NAS approaches using a DAG, the bipartite graph representation can effectively capture the topological order in the candidate operators, which is beneficial for more effectively exploring the optimal architecture for text tasks.
2) We adopt an EA to search for optimal neural architectures in the reduced research space, whereas an approximated submodular function is introduced for an RL algorithm to gradually reduce the scale of bipartite graph-based search space.
3) Comprehensive experiments on public text classification datasets show that BGNAS consistently outperforms the state-of-the-art with lower computational costs. In addition, we show that the found architecture in the bipartite graph-based search space can be transferred to various text classification tasks.

The rest of this article is organized as follows. Section II briefly introduces the related work. Section III describes the bipartite graph representation of the search space, followed by a detailed presentation of the proposed BGNAS in Section IV. The experimental settings and results are given in Section V. Finally, Section VI concludes this article.

## II. Related Work

In this section, we first present a brief review of text representation and an introduction to main NAS approaches, followed by an account of NAS for text representation.

### A. Text Representation

Text representation plays a crucial role in NLP tasks, such as text classification [30] and natural language inference [31]. A range of text representation approaches [22] have been developed to learn the meaningful and context-aware representations of textual data. In an earlier study, the bag-of-words (BoW) model [32] is used for text representation, which represents a document as a collection of words or terms and their frequencies in the document. On the basis of BoW [32], Mikolov et al. [33] introduce the Word2Vec model to popularize the use of distributed word embeddings for text data, and Pennington et al. [34] present the GloVe model for text representation, which leverages global word co-occurrence statistics to learn word embeddings.

Besides, these are a series of neural network architectures used for text representation, aiming to capture semantic, syntactic, and contextual information present in textual data for NLP tasks. For example, the effective use of word order modeling with convolutional neural networks (CNNs) [35] and recurrent neural networks (RNNs) [36] has achieved competitive results in sentiment analysis and document classification tasks. Graph neural networks (GNNs) [37] can also enhance the representation of text data by incorporating external knowledge sources, such as knowledge graphs. Bidirectional encoder representations from transformers (BERT) [38] is introduced as a contextualized word embedding model, using a transformer architecture [39] and delivering state-of-the-art performance in text representation. black Although these models have made significant progress, manually designing and tuning neural architectures remains labor-intensive and time-consuming, especially when NLP tasks become more complex and demand more flexible architectures to capture the subtle nuances of text [3], [28].

### B. Neural Architecture Search

The emergence of NAS has created a new opportunity for customized architecture design in the AutoML domain, the aim of which is to search for optimal network architectures for specific tasks.

Recent work on NAS algorithms can be roughly categorized into three groups, gradient-based [4], [40], [41], EA-base [9], [14], and RL-based [12], [42], [43]. Gradient-based algorithms treat the architecture search problem as an optimization problem and use gradient-based optimization methods to find architectures, such as a differentiable architecture search (DARTS) [4], [44]. Several DARTS variants [10], [40], [41] address this problem by pruning operators on every edge except for the one with the largest architectural weight. Besides, Feng et al. [45] propose a differentiable search strategy based on the beneficial components to automatically search for adversarially robust lightweight neural architectures, maintaining the model size while deriving beneficial components and ensuring adversarial robustness. EA-based approaches [9], [14] rely mainly on stochastic search operators such as crossover and mutation and evolve a population of diverse architectures. For example, Lu et al. [9] introduce an automatic search strategy with a genetic algorithm for the optimal network architecture. However, EA-based methods usually require a large amount of computation cost for performance evaluations. Therefore, Wu et al. [28] use efficient Cartesian genetic programming with a crossover operator, a lightweight age mechanism, and two adaptive mutation operators to facilitate NAS with fewer evaluations. RL-based NAS approaches [2], [12] use an agent to guide different candidate architecture generation by optimizing a reward function. For example, a weight-sharing strategy between submodels is proposed in ENAS [42] to reduce the search time using a long short-term memory (LSTM) controller to select various subgraphs from the search space. These approaches need a huge number of attempts to get a positive reward for updating neural architectures, making them computationally expensive during training [46]. Although various search algorithms are used to improve search efficiency, NAS may obtain a neural architecture that overfits both training and validation data, especially when the search space is huge [47].

Therefore, human expertise is frequently necessary to define a search space that achieves a well-balanced tradeoff between exploration and exploitation for diverse applications [12]. There are two types of search space widely used for NAS, namely, the micro search space and the macro search space [5]. The macro search space is usually used for building the topological structure of a neural architecture, whereas the micro search space details the candidate operators between nodes (or cells) inside a neural network. Typically, the macro search space is represented by a supernet (direct acyclic graph) which is evolved automatically. In contrast, the micro search space is often referred to as a cell structure that is optimized by specific tasks. Due to huge computational resources and search costs, the existing NAS algorithms use either macro or micro search space instead of both of them at the same time [19]. In particular, previous research on NAS algorithms prefers the micro search space as it works well on different image-related tasks [48]. For example, several strategies to reduce the scale of search space in NAS for image-related tasks have recently been proposed, including accuracy-based approaches [26], [49], [50], magnitude-based approaches [51], and angle-based approaches [20]. However, these strategies for micro search space cannot be directly applied to macro search space, and it has been shown that the macro search space is better than the micro search space for text-related tasks [14].

Therefore, search space reduction in NAS for text classification tasks is still in its infancy. One pivotal issue is how to optimize the utilization of the search space representation for text-related tasks. Due to the importance of the order and combination of candidate operators, searching on DAG search space may be inefficient due to the lack of context of text representation.

### C. NAS for Text Presentation

NAS has gained significant attention in both text and computer vision domains, yet the underlying requirements and challenges of these tasks differ significantly [9]. In computer vision, NAS focuses on optimizing architectures to capture spatial features and local patterns, leveraging the structural regularity of images [47]. In contrast, NAS for text representation must tackle with the sequential nature of language, where contextual dependencies and semantic information play a crucial role [12]. Unlike tasks in computer vision, NAS needs to handle sequence data in text representation, which requires to capture more effectively the contextual dependency and semantic information [52] to improve NAS performance. For example, Fan et al. [53] present a hierarchical representation learning approach using NAS to capture the semantic and syntactic structures of source and target languages. Zoph et al. [11] reported an NAS with RL that effectively captures discriminative features in text classification. Chen et al. [54] propose a heterogeneous representation learning algorithm based on NAS to aggregate neighboring information and reduce unnecessary search for zero-shot multilabel text classification. However, most existing NASs for text representation face challenges in achieving high search efficiency due to the extremely large search space involved [55]. Therefore, this work is essentially motivated to design an effective and efficient NAS for text representation. To this end, we introduce a bipartite graph representation of the search space to deal with text-related tasks, which can also be a plug-in to improve the performance of any NAS algorithms.

## III. Bipartite Graph Representation

In this section, we describe the search space represented by a bipartite graph. First, we introduce the NAS search space for text representation. Subsequently, we provide details on transforming the DAG search space into a bipartite graph presentation. Finally, we propose associated matching rules for a bipartite graph-based search space.

### A. Search Space of NAS for Text Representation

Most previous work on NAS primarily uses a macro search space for text representation in NLP tasks, which is typically defined by a general DAG. The DAG search space for text representation is constructed following the approach in [12]. As shown in Fig. 1 (bottom left), each node in the search space represents a layer in the neural network, and the total number

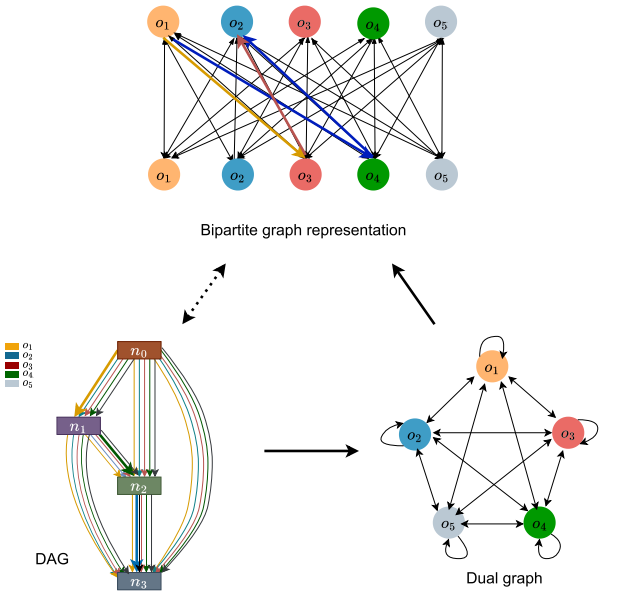



Fig. 1. Bipartite graph presentation for DAG search space. On the bottom left, the search space is depicted using a DAG, where $o_1$, $o_2$, $o_3$, $o_4$, and $o_5$ represent different candidate operators. On the bottom right, we transform the DAG into a dual graph, focusing on contextual edge operators rather than relationships between nodes in the DAG. On the top, a bipartite graph represents the dual graph, where the sequential relationship of operators is indicated by the edges in the bipartite graph. The directed bold lines, marked with different colors, form a series of bipartite matching. The bipartite matching $(\{\langle o_1, o_4\rangle, \langle o_4, o_2\rangle\})$ with the blue color corresponding to a path $(\{\langle n_0, n_1\rangle, \langle n_1, n_2\rangle, \langle n_2, n_3\rangle\})$ in the DAG, denoted by bold lines.

of nodes, denoted as $N$, is chosen based on factors such as the complexity of the networks to be designed and the available computational resources, such as GPU memory.

In a DAG search space, layers (nodes) are defined in a topological order, enabling the sampling of a child network architecture by traversing the layers according to this topological order. For each layer $n_i$, we initially choose a unique input layer from the preceding layers $n_0, n_1, \ldots, n_{i-1}$. Subsequently, we make multiple choices from the previous layers to establish skip connections, which are summed up as the output of layer $n_i$. Layer $n_0$ represents the original input layer, and layer $n_i$ must select its input from the preceding five layers. Since the operator on each node does not modify the dimension of the input data, we can freely add more layers as long as the input dimension remains unchanged. To limit the search space, a common practice is to set a fixed value for $N$, such as $N = 25$, as demonstrated in [12].

Each edge $\langle i, j\rangle$ from node $i$ to node $j$ indicates that node $i$ serves as an input or skip connection to node $j$, where an edge $\langle i, j\rangle$ exists if $i < j$. The corresponding operator $o_k$ on the edge belongs to the set of candidate operators $\mathcal{O}$, typically determined by manual expertise and prior knowledge. The candidate operators consist of a total of eight options, grouped into four categories: 1-D convolution with filter sizes of 1, 3, 5, and 7; max pooling; average pooling; gated recurrent units (GRUs); and multihead self-attention with the number of attention heads being set to 8. The probabilities of selecting these options are determined using the softmax function.

### B. Bipartite Graph Representation

To capture the context of edge operators in the DAG search space, our focus transitions from the candidate operator between nodes to the available options of context candidate operators. We transform the DAG into the dual graph first, which is further converted into a bipartite graph, where the topological order and combination of operators can be clearly illustrated in the bipartite graph representation. Each path in the original DAG can be regarded as a bigraph matching.

*Definition 1 (Dual Graph):* Dual graph $G^* = (V^*, E^*)$ is a graph created based on the original DAG $G = (V, E)$, with the roles of nodes and edges exchanged, where we introduce a corresponding vertex $v_e$ in the dual graph $G^*$ for each edge $e$ in the original graph $G$ where $e \in E$. In addition, an edge is introduced to connect all the vertices corresponding to edges $e$ adjacent to $v$ in the dual graph $G^*$ for each vertex $v$ in the original graph $G$ where $v \in V$.

*Definition 2 (Bipartite Graph):* A bipartite graph, denoted as $\hat{G} = (\mathcal{O}_1, \mathcal{O}_2, E)$, is a special graph representation of the search space in which vertices are divided into two independent candidate operator sets, $\mathcal{O}_1$ and $\mathcal{O}_2$ $(\mathcal{O}_1, \mathcal{O}_2 \subseteq \mathcal{O})$, i.e., every edge connects a vertex in $\mathcal{O}_1$ to one in $\mathcal{O}_2$. Vertex sets $\mathcal{O}_1$ and $\mathcal{O}_2$ are usually called the parts of the bipartite graph.

*Definition 3 (Bigraph Matching):* In a bipartite graph $\hat{G}$, a bigraph matching is a set of edges chosen in such a way that no two edges share a common endpoint.

*Definition 4 (Reduced-Scale Bipartite Graph-Based Search Space):* A reduced-scale bipartite graph-based search space involves representing potential neural network architectures using a bipartite graph, where there are the reduced-scale search operators to efficiently explore and identify optimal architectures.

As depicted in Fig. 1, the bottom left shows the original search space DAG, which includes four architectural nodes and five candidate edge operator options ($o_1$, $o_2$, $o_3$, $o_4$, and $o_5$), marked with different colors on each edge. On the bottom right, the dual graph of the original DAG is provided. At the top, a bipartite graph $\hat{G}$ presents the context of the operators in the dual graph. The directed edges in the bipartite graph represent the sequential relationship of operators, and the combination of different candidate operators can be defined with bigraph matching in a reduced-scale bipartite graph-based search space, where there are only four operators ($o_1$, $o_2$, $o_3$m and $o_4$). The set of sequential edges $\{\langle o_1, o_4\rangle, \langle o_4, o_2\rangle\}$ marked with the blue color forms a bigraph matching, corresponding to a path $\{\langle n_0, n_1\rangle, \langle n_1, n_2\rangle, \langle n_2, n_3\rangle\}$ marked with bold lines in the original DAG. In other words, operator $o_5$ does not form an effective contextual combination with other operators in the bipartite graph representation, and thus it is considered redundant. Removing $o_5$ can not only simplify the search space of NAS but also facilitate the effective exploration and utilization of the potential advantages offered by other operators.

The transformation of the NAS search space represented by DAG into that by bipartite graph aims to effectively identify and remove redundant operators. The transformation initially involves obtaining the dual graph of the DAG search space, which is essentially a spatial mapping issue. Using the bipartite graph representation for the DAG search space, the same candidate operators are distributed across two separate sets. All kinds of combinations of operators can be represented by a series of bigraph matching in a bipartite graph without redundant operators, which has the potential to offer a more

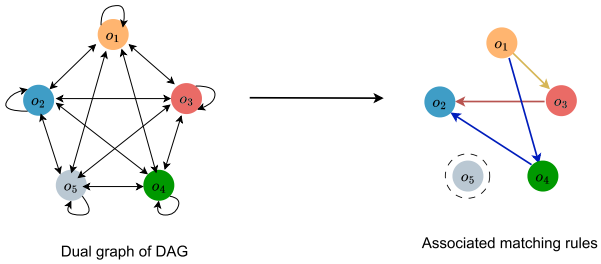



Fig. 2. Example of three associated matching rules from the dual graph of DAG, where the rules are associated with bigraph matching in the bipartite graph shown in Fig. 1.

effective text representation and enhance the efficiency of the NAS search process. The reason is that it emphasizes the matching and sequential relationships between operators, making the proposed BGNAS more intuitive and effective for exploring potential network architectures with the reduced bipartite graph search space compared with the traditional DAG search space.

### C. Associated Matching Rules for Bipartite Graph-Based Search Space

In the proposed BGNAS, we assume that the associated matching rules in bigraph matching are adapted to identify the most effective candidate operators for text representation, where the associated matching rules influence how the bigraph matching is formed.‘

*Theorem 1:* If the search space facilitates multimatching ensembles in a bipartite graph, the optimal architecture exists within the reduced search space generated from the associated matching rules.

*Proof of Theorem 1:* Given that the DAG search space supports multiple paths’ ensemble for representing text sequences [12], incorporating multimatching ensembles in a bipartite graph is achievable. The bipartite graph-based search space is represented using a series of bipartite matching ensembles. These ensembles are guided by the associated matching rules that have the corresponding mappings in the original DAG search space. As illustrated in Fig. 2, three associated matching rules with different colors are derived from the dual graph of the DAG. These rule sets, denoted as $\{\langle o_1, o_4\rangle, \langle o_4, o_2\rangle\}$, $\{\langle o_3, o_2\rangle\}$, and $\{\langle o_1, o_3\rangle\}$, are associated with bigraph matching in the bipartite graph representation of the DAG, as presented at the top of Fig. 1. It is worth noting that $o_5$ has no sequential relationship with the other four operators ($o_1, o_2, o_3$, and $o_4$). Therefore, there is a high probability that the associated matching involving $o_5$ does not exist in the bipartite graph search space.

Based on the generated associated matching rules in [56], we consider the reduction of the bipartite graph-based search space as finding the equivalence subset $\mathcal{O}'$ of the candidate operator set $\mathcal{O}$

$$\begin{aligned} &\arg\max_{\mathcal{O}'\subseteq\mathcal{O}} f(M_{\mathcal{O}'}) = \mathrm{Acc}(A^*) \\ &\text{s.t. } c(\mathcal{O}') \leq \Theta \end{aligned} \tag{3}$$

where the function $f(.) : 2^{\mathcal{O}} \longrightarrow \mathbb{R}$ is defined as the accuracy of the optimal architecture $A^*$ found through NAS search, and the constraint function $c(\mathcal{O}') = |\mathcal{O}'|$ is used to compute the cost of NAS search. $\Theta$ represents the cost constraint value, which can be used to limit the scale of the reduced bipartite graph-based search space. The bipartite graph-based search space related to the operator subset $\mathcal{O}'$ is $M_{\mathcal{O}'}$.

Let the operator subset $\mathcal{O}'$ correspond to a reduced bipartite graph search space generated based on the associated matching rules. It follows that given any operator subset $\mathcal{O}_1 \subseteq \mathcal{O}'$, we have $f(M_{\mathcal{O}_1}) \leq f(M_{\mathcal{O}'})$. This means that the performance in bipartite graph search space $M_{\mathcal{O}_1}$ will not exceed the performance in bipartite graph search space $M_{\mathcal{O}'}$ that is generated based on the associated matching rules. However, adding the candidate operators to a bipartite graph-based search space $M_{\mathcal{O}_1}$ does not guarantee a higher accuracy for the optimal architecture, because we cannot guarantee that the function $f(.)$ is submodular. That is, for $\forall \mathcal{O}_1 \subseteq \mathcal{O}_2 \subseteq \mathcal{O}$, $o_k \notin \mathcal{O}_2$, it is not necessarily true that $f(M_{\mathcal{O}_1} \cup \{o_k\}) - f(M_{\mathcal{O}_1}) > f(M_{\mathcal{O}_2} \cup \{o_k\}) - f(M_{\mathcal{O}_2})$. To select the proper subset $\mathcal{O}'$ to obtain the optimal architecture in (3), we define the submodular ratio $\alpha_f$ to measure how close function $f(.)$ is to a submodular function [57], which is calculated as follows:

$$\alpha_f = \frac{f(M_{\mathcal{O}_1} \cup \{o_k\}) - f(M_{\mathcal{O}_1})}{f(M_{\mathcal{O}_2} \cup \{o_k\}) - f(M_{\mathcal{O}_2})} \tag{4}$$

where $\alpha_f \in (0, 1]$. $\alpha_f = 1$ indicates that function $f(.)$ is submodular. When $f(\cdot)$ is not submodular, a lower bound estimation for $\alpha_f$ is relied upon, which is computed using approximated algorithms [58], [59]. In BGNAS, we use an EA to estimate the lower bound $\alpha_f$ of the submodular function $f(.)$, which provides a conservative estimate of NAS performance. By means of bipartite graph-based search space reduction (Section IV-D) in EAs, the properties of the submodular function can be effectively explored and evaluated, thereby identifying its lower bound, $\alpha_f$, as defined in (4). This process improves search efficiency and ensures that an acceptable optimal architecture can be identified.

The use of the submodular function is due to its property of marginal diminishing returns [60]. This helps us more effectively select and add edges in the search process of BGNAS, thereby identifying the optimal neural architecture more quickly. In the proposed BGNAS, marginal diminishing returns in bipartite graph matching refer to the fact that as you gradually add more edges (connections) to the bipartite graph-based search space, the additional gain (the improvement in accuracy) from each new edge decreases. If the function $f(.)$ is estimated to closely resemble a submodular function, a reduced-scale bipartite graph-based search space containing the optimal network architecture can be identified. Thus, we complete the proof of Theorem 1.

Therefore, Theorem 1 confirms that the reduced-scale bipartite graph-based search space still contains the optimal architecture that can be found by NAS. Based on Theorem 1, the proposed BGNAS adopts an EA to identify a reduced bipartite graph-based search space by locating an approximated lower bound $\alpha_f$. Considering the associated matching rules with multimatching ensembles in a bipartite graph, BGNAS aims to explore comparatively relevant architectures for text classification.

## IV. NAS Based on Bipartite Graphs

As discussed above, the bipartite graph representation enables the realization of a neural network architecture through a multimatching relationship within the bipartite graph. Different multimatching relationships between candidate operators will result in a series of neural networks with diverse neural architectures. In this study, the candidate operators and their implementation order are represented in a bipartite graph to obtain optimal neural architectures. In BGNAS, each individual can be viewed as a bipartite graph-based search space, and the association rule sets in the bipartite graph guide the reduction of the bipartite graph search space for each individual. Moreover, we do not directly evaluate the performance of bipartite graph-based search space; instead, we use an RL-based NAS algorithm to search optimal neural architectures for text classification. Consequently, BGNAS can avoid redundant training in the complete bipartite graph-based search space, thereby effectively reducing the high computational costs usually required by traditional NAS approaches in text classification.

### A. Overall Framework

The pseudocode of BGNAS is presented in Algorithm 1, which primarily comprises three components, namely, multimatching encoding of neural architecture (lines 1 and 2), RL-based NAS (lines 7–24) with an approximated submodular function, and bipartite graph-based search space reduction based on associated matching rules (line 28). It begins with an individual (a neural architecture) from population $P$ in the bipartite graph representation (line 2), and new individuals are generated through bitwise mutation (line 6). The variables $u_i$ and $q_i$ are used to explore the reduced-scale bipartite graph-based search space. The bipartite graph-based search space can be reduced within iter generations, which in turn enhances the search efficiency while minimizing computational costs. In particular, the associated matching rules can be mined, and infrequent matches in the bipartite graph-based search space are pruned for the current individuals, requiring continuous updates. The associated matching rules for each generation can offer a certain level of interpretability regarding the context of operators in text representation. The value of the maximum number of matching edges, denoted by $\Theta$, can be updated, and the bipartite graph-based search space can be reduced as specified in Algorithm 2. More details about Algorithm 2 can be illustrated in Section IV-D.

The overall framework of BGNAS is illustrated in Fig. 3. It is worth noting that the architecture of the proposed BGNAS is distinct from that of most existing NAS methods for text classification. We primarily use an EA to reduce the bipartite graph-based search space iteratively, not as the search strategy in our BGNAS. By applying EAs, we can effectively identify and eliminate architectures that underperform or lack competitiveness. In BGNAS, we use a bipartite graph to represent the search space, which enables effective acquisition of context for candidate operators used in text representation. This proposed BGNAS not only reduces the bipartite graph-based search space to mitigate computation costs but also allows us to focus on more promising network architectures.

**Algorithm 1** Proposed BGNAS

**Input:** Bipartite graph-based search space $U_{\mathcal{O}}$, iteration times $T$, and rule iteration interval Interval.
**Output:** The optimal architecture $A^*$

1: Initialize the population $P$ with an multimatching individual from the bipartite graph-based search space.
2: Initialize the bipartite graph-based search spaces $u_0, q_0 = 0^{|U_{\mathcal{O}}|}$, respectively.
3: $t = 0, NumPop = 1, temp = 0; \Theta = |U_{\mathcal{O}}|$.
4: **while** $t < T$ **do**
5: Choose the individual $M_b(p_i)(1 \le i \le NumPop)$ from $P$ uniformly at random.
6: Generate a new individual $\hat{M}_b(p_i)$ using bitwise mutation to $M_b(p_i)$.
7: **if** $|\hat{M}_b(p_i)| < \Theta$ **then**
8: $l = |\hat{M}_b(p_i)|$
9: **if** $bin(l) = \emptyset$ **then**
10: $P = P \cup \{\hat{M}_b(p_i)\}$
11: $u_l = q_l = \hat{M}_b(p_i)$
12: **else**
13: **if** $g(\hat{M}_b(p_i)) \le g(u_l)$ **then**
14: $u_l = \hat{M}_b(p_i)$
15: $NumPop = NumPop + 1$
16: **end if**
17: **if** $f(\hat{M}_b(p_i)) \le f(q_l)$ **then**
18: $q_l = \hat{M}_b(p_i)$
19: $NumPop = NumPop + 1$
20: **end if**
21: $P = (P - bin(l)) \cup \{u_l\} \cup \{q_l\}$
22: Applying the RL-based search algorithm [42] to search the optimal architecture $A^*$ based on $P$. $/*SectionIV-C*/$
23: **end if**
24: **end if**
25: $t = t + 1, temp = temp + 1$
26: **if** $temp =$ Interval **then**
27: $temp = 0$
28: Reduction of the bipartite graph-based search space for each individual according to Algorithm 2. $/*SectionIV-D*/$
29: Update the maximum number of the matching edges $\Theta$ for the current population $P$.
30: **end if**
31: **end while**

### B. Multimatching Encoding of Neural Architectures

We assess the performance of neural architectures for text classification through a multimatching encoding approach within the bipartite graph representation of search space. To this end, bipartite matching is used to substitute a sequence of path ensembles in which each edge, connecting preceding and succeeding sequences in a bipartite graph, is treated as a unit. The context of operators is maintained through the combination of these units in a bipartite graph-based

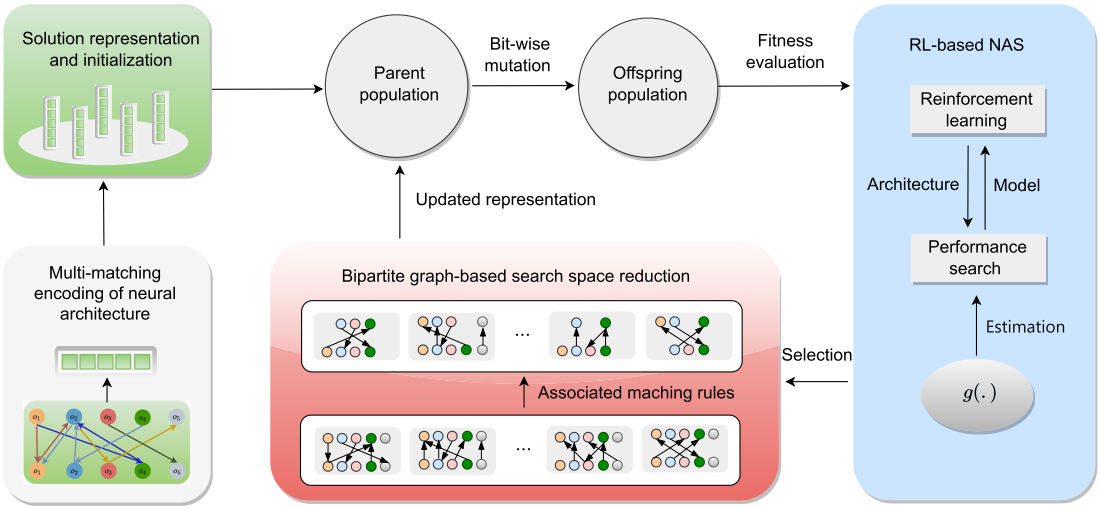



Fig. 3. Overall framework of the proposed BGNAS includes multimatching encoding of the neural architecture, RL-based NAS, and bipartite graph-based search space reduction. Each individual in the population represents a bipartite graph-based search space, and $g(.)$ is the approximated submodular function for performance search in RL-based NAS.

search space. We define that Sets $U_{\mathcal{O}}$ and $U_{\mathcal{O}'}$ represent the units of operator pairs (edges) in the complete and reduced bipartite graph-based search space, respectively. The multimatching encoding in the bipartite graph for each individual is represented by a binary vector $M_b = [bm(U_{\mathcal{O}'}, 1), bm(U_{\mathcal{O}'}, 2), \ldots, bm(U_{\mathcal{O}'}, k), \ldots, bm(U_{\mathcal{O}'}, |U_{\mathcal{O}}|)]$, of length $|U_{\mathcal{O}}|$. Each element $bm(U_{\mathcal{O}'}, k)$ indicates the inclusion or exclusion of the $k$th unit in the reduced search space. $bm(U_{\mathcal{O}'}, k) = 1$ indicates the unit belongs to $U_{\mathcal{O}'}$, and $bm(U_{\mathcal{O}'}, k) = 0$ means that the unit does not belong to $U_{\mathcal{O}'}$.

The proposed multimatching encoding scheme represents the search space as a bipartite graph. In Fig. 4, two paths from a neural architecture within a DAG search space (left) are represented as two bipartite matchings in the bipartite graph. The bottom right shows the bipartite matchings, while the top right details the bipartite graph representation. Unlike the path-based encoding used for DAG search spaces [12], the multimatching encoding emphasizes matching relationships within the bipartite graph. This focus on the flexible combinations of different operator pairs is particularly crucial for text-related tasks, primarily because it effectively addresses complex dependencies among operators and boosts the capture of contextual information.

Most importantly, by leveraging the connectivity properties of bipartite graphs, it is possible to remove operators from the operation space that are less relevant to the task, thereby reducing the complexity of the structural search. Fig. 4 provides an illustrative example of the multimatching encoding process. There are four nodes ($n_1, n_2, n_3$, and $n_4$) and five candidate operators ($o_1, o_2, o_3, o_4$, and $o_5$) in the original DAG search space. With four operators ($o_1, o_2, o_3$, and $o_4$) in the reduced bipartite graph-based search space, there are $A_4^2 + 4 = 16$ kinds of units in set $U_{\mathcal{O}'}$, where $U_{\mathcal{O}'} \subseteq U_{\mathcal{O}}$. Meanwhile, in set $U_{\mathcal{O}}$ with five operators ($o_1, o_2, o_3, o_4$, and $o_5$), there are $A_5^2 + 5 = 25$ kinds of units. The reduction in the number of candidate operators leads to a decrease in matching relationships within the bipartite graph-based search space.

This allows us to directly concatenate multiple binary vectors for aggregation. This proposed approach enables the decoding of multipaths in the DAG search space into the corresponding bipartite matchings. Notably, the units in the bipartite matching have variable lengths, providing flexibility to capture combinations of operator units. This variability enhances the ability of the NAS algorithm to generalize from diverse architectures.

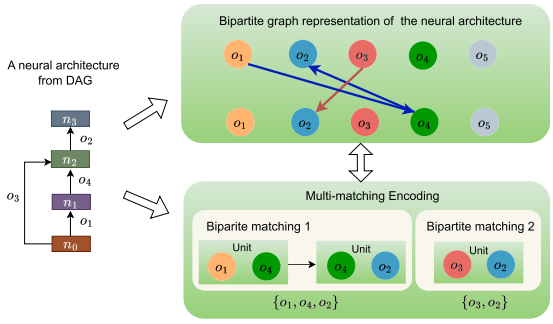



Fig. 4. Example of multimatching encoding, where $\{o_1, o_4, o_2\}$ and $\{o_3, o_2\}$ represent two bigraph matching in a neural architecture. The neural architecture in the DAG search space (DAG) can be represented in the bipartite graph, and the proposed BGNAS leverages multimatching for the context of operators with the bipartite graph representation.

### C. RL-Based NAS With the Approximated Submodular Function

In this work, we assess the performance of candidate architectures from the bipartite graph-based search space within the population. Each individual with bipartite graph representation, denoted as $M_b(p_i)$, represents a general bipartite graph search space. We use the RL-based search algorithm for weight sharing NAS [42] to evaluate the individual's fitness. Within the framework of a bipartite graph-based search space, we leverage a single LSTM to sample child neural architectures. Subsequently, these child architectures undergo training on the training set and evaluation on the validation set. Simultaneously, the child neural architectures share a common set of parameters within the ongoing bipartite graph search space, facilitating an accelerated evaluation procedure. Once the performance of the child neural architectures is assessed,

the accuracy feedback is incorporated into the LSTM layer. Consequently, the parameters of the entire bipartite graph search space undergo updating through RL.

In particular, we introduce an approximated submodular function $g(\cdot)$ to estimate a lower bound for $\alpha_f$ in the reduced bipartite graph search space as described in (4). The performance of an individual $M_b(p_i) \in U_{\mathcal{O}'}$ in the population is defined as follows:

$$g(M_b(p_i)) = f(M_b(p_i)) \Big/ \left(1 - \exp\left(\frac{-\alpha_f c(U_{\mathcal{O}'})}{\Theta}\right)\right). \quad (5)$$

In (5), $\Theta$ is the maximum number of matching edges for the current associated matching rules, and the initial value is set to $n-1$, where $n$ is the number of the candidate operators in the complete bipartite graph-based search space. Similar to $c(.)$ in (3), $c(.)$ is leveraged to compute the cost of NAS on the bipartite graph search space. For the initial reduced bipartite graph-based search space, $|U_{\mathcal{O}'}| = 0$, we set $g(\cdot) = f(\cdot)$ outlined in (3), where $f(.)$ is the accuracy of individual $M_b(p_i)$. According to the theoretical analysis in [58], the proposed BGNAS gives an upper bound of $2en^2(n+1)$ iterations when a lower bound $\alpha_f$ is applied to the approximated submodular function. In this case, BGNAS can find a reduced bipartite graph-based search space $U_{\mathcal{O}'} \subseteq U_{\mathcal{O}}$ with

$$f(M_b(p_i)) \geq \left(\frac{\alpha_f}{2}\right)(1 - e^{-\alpha_f}) \cdot f(M_b(p^*)) \quad (6)$$

where $M_b(p^*) \in U_{\mathcal{O}'}$ is the optimal solution for the current generation, and $f(M_b(p_i))$ is defined in (3). In particular, the lower bound $\alpha_f$ and $\Theta$ can be estimated by mining the associated matching rule with the bipartite graph representation, as illustrated in Section IV-D.

During the performance search process, we randomly select an individual $M_b(p_i)$ from population $P$ and use bitwise mutation to generate a new multimatching vector $M'_b(p_i)$ in the bipartite graph representation. The proposed BGNAS maintains the solutions in population $P$, and the new solution $M'_b(p_i)$ will only be compared with the solutions in $\text{bin}(|M'_b(p_i)|)$, which can be formulated as follows:

$$\text{bin}(|M'_b(p_i)|) = \{M_b(p_j) \in P || M_b(p_j)| = |M'_b(p_i)|, i \neq j\}. \quad (7)$$

Moreover, only if individual $M'_b(p_i)$ has a number of matching edges (after removing duplicated operator units) smaller than the constraint $\Theta$ of the current generation, it can be added to the population $P$. If the population $P$ does not have an individual of the same matching edges as the individual $M'_b(p_i)$, individual $M'_b(p_j)$ will be directly added to population $P$.

### D. Bipartite Graph-Based Search Space Reduction

By analyzing potential preceding and succeeding matching relationships among these individuals, we use a pruning strategy [56] in each generation to explore the associated rules between candidate operators, thereby reducing infrequent matching in the bipartite graph-based search space. Algorithm 2 demonstrates the procedure of reducing the bipartite graph-based search space based on associated matching rules. In Algorithm 2, the associated rule in a bipartite graph indicates a specific multimatching relationship, where any subset of the set of operator units is denoted as an itemset. An associated matching rule is considered supported if the percentage of items in all individuals of $P$ exceeds a support threshold $\text{Sup}_{\min}$. A set of items containing $k$ matching edges (items) is referred to as a $k$-itemset. During the generation process of $k$-itemsets, we perform a self-joined operator on the items in the current $k$-itemset to obtain the $k+1$-itemset. Notably, infrequent itemsets are pruned from the candidate itemsets. This exploration results in the generation of infrequent rules set $\text{IR} = \bigcup_{k=1}^{\text{Maxlen}}\{C_k - L_k\}$, as well as the associated edge rules set $\text{AR} = \bigcup_{k=2}^{\text{Maxlen}} L_k$ for matching in the bipartite graph. With the above procedure, we can reduce the scale of bipartite graph-based search space by removing the related bigraph matchings whenever there exists any subset of the infrequent rules set IR. Moreover, we calculate the maximum number of matching edges with an overlapping strategy to update the constraint $\Theta = \sum_{i=2}^{|C_1|-1} |L_k|$ based on the associated edge rules AR in (6). The lower bound $\alpha_f = \min_k(|C_k - L_k|/|C_k|)$ is updated and estimated as the submodular ratio in (6).

Fig. 5 provides an illustrative example with ten individuals and five candidate operators. It demonstrates the procedure of generating associate rules for reducing the bipartite graph-based search space. The minimum support threshold $\text{Sup}_{\min}$ is set to 2. As presented in the table on the top left of Fig. 5, each row corresponds to an individual, illustrating the multimatching encoding representation. At first, all the

**Algorithm 2** Procedure of Reducing the Bipartite Graph-Based Search Space Based on Associated Matching Rules

**Input:** Population $P$, constraint cost $\Theta$, candidate itemset $C_1$, and support threshold $\text{Sup}_{\min}$.
**Output:** The maximum number of matching edges $\Theta$.

```
Generate frequent 1-itemsets L_1, by eliminating the 1-itemsets from the C_1 whose support level is less than Sup_min.
Calculate the maximum length Maxlen of the bigraph matchings from all individuals.
C_k = ∅, L_k = ∅, U_O' = U_O, k ∈ (1, |C_1| − 1].
for (k = 1; L_k ≠ ∅; k + +) do
    The itemsets in L_k are listed from left to right in ascending order based on the sequence number of the operators.
    Perform a self-join on L_k with L_1 to derive the candidate itemsets C_{k+1}.
    Prune the infrequent itemsets from C_{k+1} if the support level of itemsets are less than the support threshold Sup_min, and generate the frequent itemsets L_{k+1}
    for each individual in population P do
        increment count of candidates in C_{k+1} that are contained in the individual.
    end for
    L_{k+1} = candidates in C_{k+1}, where the support level of candidates are greater than or equal to Sup_min.
end for
Obtain the infrequent rules set IR = ⋃_{k=1}^{Maxlen} {C_k − L_k}.
Reduce the scale of the bipartite graph-based search space by removing the corresponding bigraph matching for each individual based on the infrequent rules set IR.
Obtain the associated edge rules set FR = ⋃_{k=2}^{Maxlen} L_k.
Calculate the maximum number of matching edges Θ = FR with the associated edge rules.
```

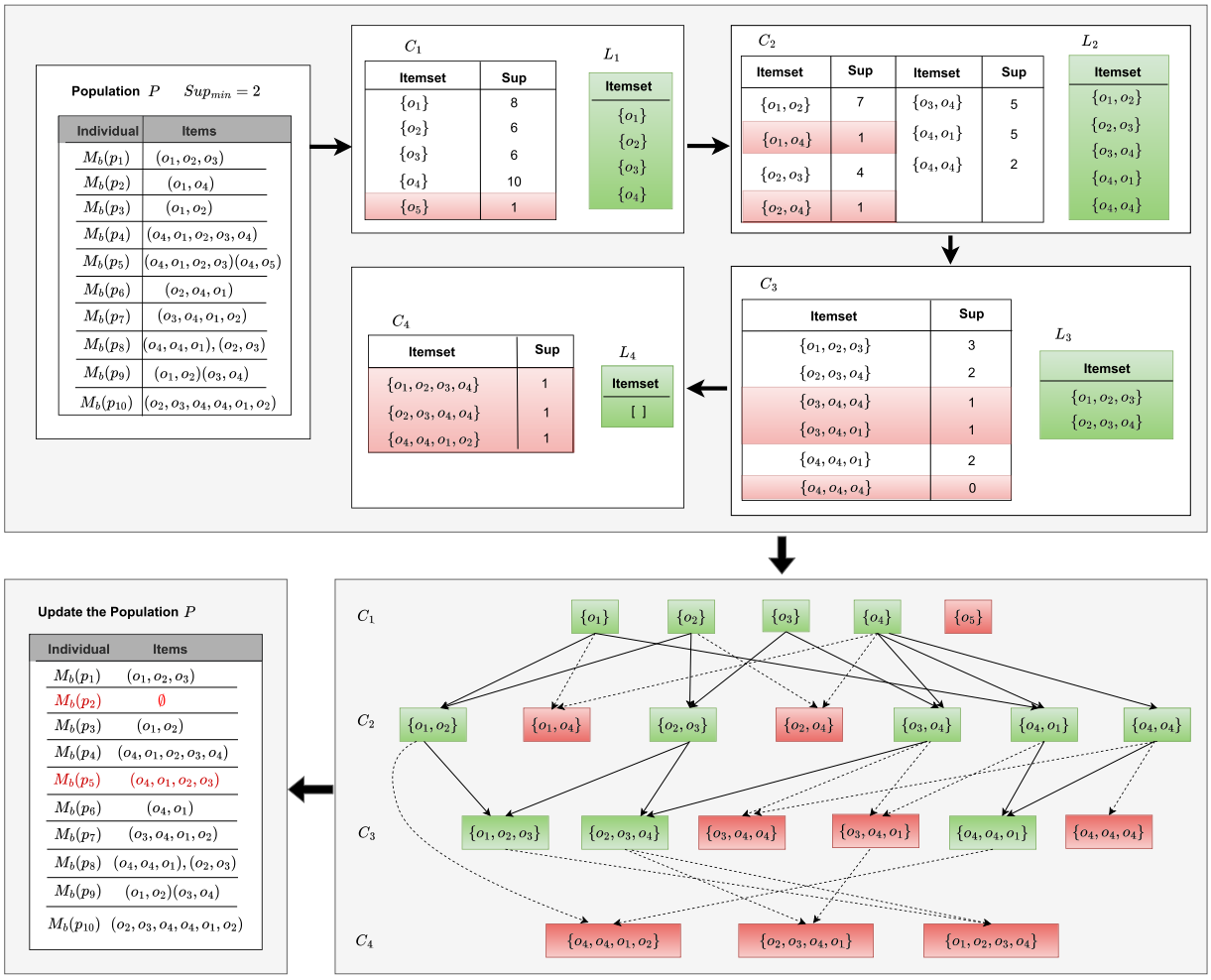



Fig. 5. Illustrative example of the generated associated matching rules in the bipartite graph-based search space. (Top) The process of generating frequent $i$-itemsets $L_i$, $i = 1, 2, 3, 4$, is illustrated based on the population $P$ with ten individuals and five candidate operators. (Bottom right) Potential association rules with multilevel items, where the red box highlights the corresponding pruned association infrequent matching rules. (Bottom left) The update of the population with the individuals that reduce the scale of the bipartite graph-based search space.

individuals are scanned through to identify, for each 1-itemset, the corresponding list of individuals and to calculate the number of individuals for each operator in the candidate first-level itemset $C_1$. Next, eliminating all 1-itemsets with a support level less than 2, such as $o_5$, results in $L_1 = \{o_1, o_2, o_3, o_4\}$. Second, self-joining the items to $L_1$ generates candidate 2-itemsets through an overlapping strategy. The support of candidate second-level itemsets $C_2$ is calculated, and all 2-itemsets with a support level less than 2 are eliminated. This process results in $L_2 = \{o_1, o_2, o_2, o_3, o_3, o_4, o_4, o_1, o_4, o_4\}$. Similarly, frequent $k$-itemsets ($k = 2, 3$) are generated by combining $L_{k-1}$ with $L_1$, using an overlapping strategy to count the support of $k$-candidate itemsets. After the intersection, the $k$-level itemsets in $C_k$ are pruned if their support level is less than 2. Therefore, frequent 3-itemsets $L_3 = \{o_1, o_2, o_3, o_2, o_3, o_4, o_4, o_4, o_1\}$ and frequent 4-itemsets $L_4 = \{o_1, o_2, o_3, o_4\}$ are obtained. Finally, the pruning strategy terminates when the 4-itemset $L_4$ is null, and the individuals in the current generation are updated in the bottom left of Fig. 5. The pruned association infrequent matching rules in a certain level of itemsets are highlighted in the red box in the bottom right of Fig. 5.

When an itemset (whether first level, second level, or higher level) is no longer frequent, all bigraph matchings containing this itemset need to be excluded from the corresponding bipartite graph search space. In Fig. 5, as $\{o_5\}$ has already been excluded from first-level itemset $C_1$, a bigraph matching $\{o_4, o_5\}$ containing $\{o_5\}$ as a subset also needs to be removed from the corresponding bipartite graph search space $M_b(p_5)$. Similarly, as $\{o_1, o_4\}$ has been removed from the second-level itemset $C_2$, and $\{o_1, o_4\}$ appears as a true subset in a bipartite graph matching $\{o_1, o_4\}$ of the corresponding bipartite graph search space $M_b(p_2)$, therefore the bipartite graph matching $\{o_1, o_4\}$ also needs to be removed. Deleting infrequent rules enables the reduction of the bipartite graph-based search space, and the scale of bipartite graph-based search space can be effectively reduced, thereby saving substantial computation costs in the proposed BGNAS.

## V. Experimental Results

In this section, we begin by empirically conducting basic experiments to validate the performance of BGNAS on the SST dataset. We also explore the transferability of the derived

TABLE I
STATISTICS OF EIGHT TEXT CLASSIFICATION DATASETS

| Dataset | Classes | Training | Validation | Test |
|---|---|---|---|---|
| SST | 5 | 8,544 | 1,101 | 2210 |
| SST-B | 2 | 6,920 | 872 | 1821 |
| AG | 4 | 12,000 | - | 7600 |
| DBP | 14 | 560,000 | - | 70,000 |
| YELP-B | 2 | 560,000 | - | 38,000 |
| YELP | 5 | 650,000 | - | 50,000 |
| YAHOO | 10 | 1,400,000 | - | 60,000 |
| AMZ-B | 2 | 3,600,000 | - | 400,000 |

architectures to text classification tasks on additional benchmark datasets. Subsequently, we delve into comprehensive experiments designed to analyze the effectiveness and efficiency of the extended bipartite graph-based search space.

### A. Datasets and Experimental Setting

In this study, we use eight publicly available datasets widely used for text classification to evaluate the BGNAS, including include SST [61], SST-B,[1] AG,[2] DBP [62], YELP-B [63], YELP [63], YAHOO, and AMZ-B [64]. The datasets cover diverse domains, encompassing sentiment analysis, Wikipedia article categorization, news categorization, and topic classification. Notably, these datasets also exhibit diversity in text domains, data distribution, and languages. For example, AG and Yahoo cover the news and Q&A domains, YELP and AMZ-B contain different types of user reviews, and SST and SST-B focus on sentiment classification. In addition, the YELP-B dataset has a serious class imbalance issue, with positive reviews accounting for nearly 80% of the data, while DBP and AMZ-B involve multiple languages. These variations in different datasets ensure that our BGNAS is thoroughly evaluated under diverse conditions.

BGNAS is comprehensively evaluated across diverse text classification scenarios. We perform NAS and assess the derived architectures on eight public datasets, as summarized in Table I. To address the absence of validation samples in some datasets, as set in [12], we randomly select 5% of the training samples for the required validation set in the NAS.

In the proposed BGNAS, individuals are represented using a form of multimatching encoding, and the maximum number of generations for the evolutionary bipartite graph-based search space is set to 50. The iteration interval for bipartite graph-based search space reduction is set to 10, and the support threshold for associated matching rules is set to 2. In each iteration, the number of search epochs for RL-based NAS is set to 10. To evaluate the bipartite graph-based search space for each individual, we adopt the training settings used in [12]. Specifically, we configure the batch size as 128, set the maximum input length to 64, and define the hidden unit dimension for each layer as 32. Incorporating dropout (with a ratio of 0.5) in the embedding layers, final output layers, and self-attention layers, we use the Adam optimizer with an initial learning rate of 0.1. The learning rate gradually decays to zero following a cosine annealing schedule, with the length of each cosine cycle set to the generation number. Following the population evolution, we use the architecture with the highest accuracy as the text representation.

Moreover, we evaluate our architecture against state-of-the-art networks designed by human experts, including FastText [65], 29-CNN [66], bidirectional LSTM (Bi-LSTM)-Max [67], and a 24-LAYERS TRANSFORMER [39]. We also compare the proposed approach with the DAG search spaces defined in ENAS [42], along with other NAS algorithms, including ONE-SHOT [68], RANDOM SEARCH [69], DARTS [4], and ECGP [28]. In addition, we compare three NAS algorithms with improved DAG search spaces for text representation: TextNAS [12], AutoAttend [14], and DNA-MHE [3].

### B. Comparative Results

*1) Results on SST:* In this section, we verify the effectiveness of the proposed BGNAS for text classification tasks. The evaluation results and comparisons with other state-of-the-art neural networks, including both manually designed and NAS-based, on the SST dataset are summarized in Table II. In the compared NAS algorithms, ONE-SHOT [68] and Random Search [69] use a random search strategy. DARTS [4] and DNA-MHE [3] use gradient-based search methods. AutoAttend [14] and ECGP [28] use EA-based search methods. ENAS [42] and TextNAS [12], along with our proposed BGNAS, adopt an RL-based search method.

From Table II, we observe that the neural architecture searched by BGNAS achieves competitive performances compared with the manually designed networks, while also requiring fewer parameters. In addition, BGNAS outperforms neural architectures found by other NAS algorithms obtaining the best results in terms of accuracy and parameters. BGNAS demonstrates an encouraging improvement over random search-based and gradient-based NAS methods in terms of accuracy and search cost. Compared with population-based NAS methods, BGNAS achieves significantly better performance than AutoAttend. Although ECGP incurs lower search costs than BGNAS, it ends up with substantially more parameters (132.0 versus 130.1 M) and relatively lower accuracy (47.00% versus 53.74%) than BGNAS. It should be noted that our BGNAS is the most efficient NAS method among RL-based methods. The accuracy of BGNAS has improved by 2.09% from ENAS and 1.13% from TextNAS on the SST dataset, respectively. This illustrates the superiority of the bipartite graph-based search space for text classification.

*2) Transferability Analyses:* In this section, we validate the generality of the proposed BGNAS in other text classification tasks. Table III lists the experimental results on seven datasets and shows comparison of them with other state-of-the-art NAS methods and the manually designed networks. We illustrate the average text classification accuracy over five independent runs in Table III, with the best value in each section shown in bold. During the transfer of the derived architecture from SST to seven other text classification datasets, our BGNAS consistently outperforms the existing top NAS results and manually designed models. These results highlight that even when initially searched on smaller datasets like SST, the

[1]https://nlp.stanford.edu/sentiment/code.html
[2]http://groups.di.unipi.it/gulli/AG_corpus_of_news_articles.html

TABLE II
COMPARISON WITH OTHER STATE-OF-ART METHODS ON THE SST DATASET

| Model | Accuracy (%) | Params (M) | Search Cost (GPU days) | Search Method |
|---|---|---|---|---|
| FastText [65] | 47.50 | 236.6 | - | manual |
| 29-CNN [66] | 49.60 | 234.2 | - | manual |
| Bi-LSTM-Max [67] | 51.00 | 196.9 | - | manual |
| 24-LAYERS TRANSFORMAER [39] | 49.51 | 265.3 | - | manual |
| ONE-SHOT [68] | 50.37 | 207.5 | 16.7 | random |
| RANDOM SEARCH [69] | 49.20 | 216.7 | 15.1 | random |
| DARTS [4] | 51.65 | 238.7 | 32.1 | gradient-based |
| DNA-MHE [3] | 52.37 | 269.1 | 16.3 | gradient-based |
| ECGP [28] | 47.00 | 132.0 | **5.8** | population-based |
| AutoAttend [14] | 53.71 | 377.7 | 116.7 | population-based |
| ENAS [42] | 51.55 | 263.3 | 12.5 | RL-based |
| TextNAS [12] | 52.51 | 245.8 | 11.6 | RL-based |
| **BGNAS (ours)** | **53.74** | **130.1** | 10.9 | **RL-based** |

TABLE III
AVERAGE TEXT CLASSIFICATION ACCURACY ON SEVEN DATASETS

| Model | SST-B | AG | DBP | YELP-B | YELP | YAHOO | AMZ-B |
|---|---|---|---|---|---|---|---|
| FastText [65] | — | **92.50** | 98.60 | 95.70 | 63.90 | 72.30 | 94.60 |
| 29-CNN [66] | 87.30 | 92.17 | 93.67 | **95.72** | **64.72** | **72.68** | 92.97 |
| Bi-LSTM-Max [67] | **88.60** | 92.49 | 98.61 | — | — | — | — |
| 24-LAYERS TRANSFORMAER [39] | 86.66 | 92.17 | **98.77** | 94.07 | 61.22 | 72.67 | **95.59** |
| ONE-SHOT [68] | 87.08 | 92.06 | 98.89 | 95.78 | 64.78 | 73.20 | 95.20 |
| RANDOM SEARCH [69] | 87.15 | 92.54 | 98.98 | 96.00 | 65.23 | 72.47 | 94.87 |
| DARTS [4] | 87.12 | 92.24 | 98.90 | 95.84 | 65.12 | 73.12 | 95.48 |
| DNA-MHE[3] | 90.19 | 93.15 | 98.99 | 96.54 | 66.62 | 74.51 | **96.45** |
| ECGP [28] | 85.00 | 92.17 | 94.28 | 94.51 | 63.70 | 72.52 | 94.90 |
| AutoAttend [14] | 88.17 | 93.53 | 99.08 | 96.62 | 66.82 | 74.48 | 96.04 |
| ENAS [42] | 88.90 | 92.39 | 99.01 | 96.07 | 64.60 | 73.16 | 95.80 |
| TextNAS [12] | 90.33 | 93.14 | 99.01 | 96.41 | 66.56 | 73.97 | 95.94 |
| **BGNAS (ours)** | **90.66** | **93.55** | **99.09** | **96.72** | **66.86** | **74.58** | **96.45** |

bipartite graph-based search space representation, with the learning of the topological order of operators, retains the ability to generalize effectively to similar text classification tasks. It is worth noting that in the comparison of BGNAS with manual models, the BGNAS network achieves competitive performance, with an accuracy improvement of over 2.1% in the best case. BGNAS outperforms the existing state-of-the-art NAS results on the DBP and YAHOO datasets, involving text classification with a larger number of categories than the SST datasets. It can be concluded that our BGNAS, derived from SST, is indeed transferable to a more complicated text classification task while maintaining its superiority. Interestingly, our transferred BGNAS performs slightly better than DNA-MHE, which searches for multilingual texts on the AMA-B dataset. The experimental results in Table III further demonstrate that the proposed method enables the transformation of the topological order of operators into complex bipartite graph architectures. In particular, the use of bipartite graph-based search space proves effective for solving multilingual text classification tasks.

### C. Effect of Bipartite Graph Representation

In this section, we further verify the effectiveness of the bipartite graph representation for text classification. Since the proposed BGNAS can obtain the most effective operators based on the associated matching rules from the bipartite graph-based search space, we extend the existing candidate operators [12] and use five categories of commonly used candidate operators for the text sequences as follows.

1) *Convolution Layers:* Four kinds of 1-D convolution layers with filter sizes 1, 3, 5, and 7, respectively, are defined as candidate operators. As in [12], we keep the dimension of output equal to the input.
2) *Recurrent Layers:* Two kinds of recurrent layers, namely, Bi-LSTM and GRU, are used as candidate operators in the implementation. Bi-LSTM and GRU have the advantage of capturing long-term dependencies in text representation.
3) *Pooling Layer:* The maximum and average pooling layers with a filter window are used, respectively. The multiple-choice filter size, such as 1, 3, and 5, can be used in any maximum and average pooling operators.
4) *Multihead Self-Attention Layers:* There are three kinds of multihead self-attention layers with the number of attention heads set as 8, 16, and 32, respectively.
5) *Auxiliary Layers:* Two auxiliary layers are leveraged in this work, including the zero operator and identity operator. Zero operator refers to a specific operator that has zero parameters and produces zero-valued outputs. The identity operator passes the input directly to the output without any transformation, commonly used in skip connections or residual connections.

TABLE IV
AVERAGE ACCURACY PERFORMANCE OF DIFFERENT SEARCH ALGORITHMS WITHOUT OR WITH BIPARTITE GRAPH-BASED SEARCH SPACE ON ALL DATASETS

| Model | Search | Transfer | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | SST | SST-B | AG | DBP | YELP-B | YELP | YAHOO | AMZ-B |
| RANDOM SEARCH [69] | 46.21 | 84.12 | 86.14 | 90.17 | 86.10 | 54.23 | 63.41 | 82.17 |
| **RANDOM SEARCH + BS** | 49.40 | 85.35 | 89.59 | 93.48 | 89.09 | 56.23 | 68.37 | 89.70 |
| DNA-MHE [3] | 49.91 | 88.99 | 90.15 | 95.39 | 89.54 | 56.02 | 71.51 | 89.45 |
| **DNA-MHE + BS** | 51.62 | 89.01 | 91.32 | 96.93 | 90.54 | 57.81 | 72.56 | 91.45 |
| AutoAttend [14] | 51.76 | 87.46 | 90.68 | 93.21 | 89.02 | 56.68 | 69.32 | 80.07 |
| **AutoAttend + BS** | 51.79 | 88.53 | 91.98 | 93.42 | 90.24 | 57.91 | 69.99 | 83.24 |
| TextNAS [12] | 48.37 | 88.03 | 91.92 | 95.21 | 91.07 | 57.76 | 70.21 | 81.33 |
| **TextNAS + BS** | 50.05 | 89.35 | 92.05 | 96.32 | 91.47 | 58.49 | 72.19 | 88.78 |
| BGNAS_DAG | 51.73 | 88.27 | 90.89 | 96.21 | 91.73 | 57.23 | 69.21 | 89.34 |
| **BGNAS (ours)** | **52.95** | **89.99** | **93.46** | **98.89** | **94.37** | **59.47** | **73.21** | **94.21** |

In summary, the candidate operator set $\mathcal{O}$ consists of 17 kinds of candidate operators in total for text representation, so the initial number of operator nodes in bipartite graph-based search space is 17.

Table IV shows the average classification accuracy performance of different NAS algorithms with or without bipartite graph-based search space on all datasets. The best average classification accuracy over five independent runs is highlighted in bold on the different datasets. Similarly, we first conduct NAS and evaluate its performance on the SST dataset. Subsequently, we transfer the searched architectures to seven other text classification datasets. From Table IV, we directly perform experiments using bipartite graph-based search space with four state-of-the-art NAS algorithms, including RANDOM SEARCH [69], DNA-MHE [3], AutoAttend [14], and TextNAS [12]. We also consider a variant named BGNAS_DAG, where we use a DAG to represent the search space, and the candidate operators are further constrained by the proposed BGNAS framework. To further validate the effectiveness of the bipartite graph-based search space, especially when combined with other NAS approaches, the DAG search space is replaced with the proposed bipartite graph-based search space for the four NAS search algorithms, referred to as RANDOM SEARCH + BS, DNA-MHE + BS, AutoAttend + BS, and TextNAS + BS, respectively.

Based on the results of the four comparative experiments in Table IV, it is evident that the proposed BGNAS can search for the architecture with the best average accuracy on all datasets using the evolutionary bipartite graph search space, outperforming SEARCH + BS, DNA-MHE + BS, AutoAttend + BS, and TextNAS + BS. In particular, the four kinds of NAS algorithms consistently find better architectures from the bipartite graph-based search space than from the original DAG search space for text classification on the SST datasets. When the architecture derived from SST is applied to seven text classification datasets, the efficacy of the evolutionary bipartite graph search space persists in enhancing the performance of NAS algorithms. For instance, excluding the proposed BGNAS, TextNAS + BS demonstrates the highest accuracy on the SST-B, AG, YELP-B, and YELP datasets, while DNA-MHE + BS excels on the DBP, YAHOO, and AMZ-B datasets. Furthermore, it appears that our BGNAS can effectively exploit the topological order of candidate operators based on associated matching rules, as evidenced by an absolute advantage in the comparative experiments with BGNAS_DAG.

TABLE V
COMPARISON OF THE BEST ACCURACY WITH OTHER STATE-OF-ART METHODS ON SST

| Model | Search Cost (GPU days) | Params (M) | Accuracy (%) |
|---|---|---|---|
| RANDOM SEARCH [69] | 15.7 | 236.7 | 46.41 |
| **RANDOM SEARCH + BS** | 13.4 | 225.6 | 49.49 |
| DNA-MHE [3] | 17.4 | 277.3 | 50.14 |
| **DNA-MHE + BS** | 16.9 | 245.1 | 51.93 |
| AutoAttend [14] | 136.3 | 383.7 | 51.90 |
| **AutoAttend + BS** | 126.7 | 347.7 | 52.04 |
| TextNAS [12] | 13.6 | 245.8 | 49.39 |
| **TextNAS + BS** | 12.1 | 231.2 | 50.21 |
| BGNAS_DAG | 14.7 | 252.7 | 52.06 |
| **BGNAS (ours)** | **11.1** | **176.3** | **53.19** |

To verify whether different NAS methods still exhibit generalization in the expanded bipartite graph-based search space, we provide a detailed comparison of the performance of RANDOM SEARCH, DNA-MHE, AutoAttend, and TextNAS on the SST dataset with or without the bipartite graph-based search space in Table V. These results are based on the best accuracy value for a series of NAS approaches over five independent runs. In Table V, we observe that the parameters of weight sharing in the bipartite graph can be significantly decreased. More interestingly, enhancing the performance of various NAS search algorithms with lower search costs remains beneficial.

Moreover, we present a visualization of the searched architecture found by the proposed BGNAS in Fig. S1. It can be observed that a series of associated rules can be discovered in the bipartite graph search space. In addition, we provide examples of frequent matchings during the NAS search process in Table S1, where the number of operators ranges from 2 to 9. It is worth noting that the autodesign principles are generally aligned with human experience for text representation.

### *D. Effectiveness Validation of Approximated Submodular Function*

Fig. 6 illustrates the accuracy changes on the standard and extended bipartite graph-based search space over 50 iterations

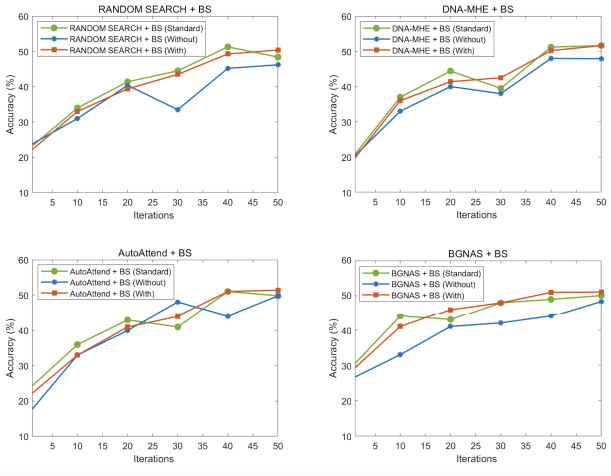



Fig. 6. Accuracy changes on the SST dataset for four NAS search methods (random, gradient-based, population-based, and RL-based) between the standard bipartite graph-based search space and the extended bipartite graph-based search space, which are observed over 50 iterations. In particular, we evaluate RANDOM SEARCH + BS, DNA-MHE + BS, AutoAttend + BS, and the proposed BGNAS with or without the approximated submodular function $g(.)$.

for RANDOM SEARCH + BS, DNA-MHE + BS, AutoAttend + BS, and BGNAS, respectively. In this comparison, four NAS methods (random, gradient-based, population-based, and RL-based search) are specifically evaluated with or without the approximated submodular function $g(.)$. This work reports that the experimental results of different NAS algorithms on the extended BS search space [without considering the approximated submodular function $g(.)$] cannot be as good as those with a standard DAG search space, except for our proposed BGNAS. Due to the need to consider the topological structure and a larger number of candidate operators up to billions, it is more challenging to optimize an extended BS search space for text classification. In particular, compared with other algorithms studied in this work, our algorithm demonstrates a more stable trend and achieves the highest accuracy. This can be attributed to our typical goal of identifying a reduced number of operators that are ideally suited for the text classification task from the expanded bipartite graph search space.

### *E. Parameter Sensitivity Analysis*

This section aims to conduct experiments to analyze the parameter sensitivity of the proposed BGNAS. The support threshold $\text{Sup}_{\text{min}}$ for associated matching rules and the rule iteration interval Interval for reducing the bipartite graph-based search space are user-defined hyperparameters. Fig. 7 shows comparison of text classification accuracy on the SST dataset obtained by BGNAS under different hyperparameter combinations of $\text{Sup}_{\text{min}}$ and Interval. We set $\text{Sup}_{\text{min}}$ from 1 to 5 with a step size of 1, while Interval is set from 5 to 25 with a step size of 5. As can be seen from Fig. 7, the proposed BGNAS performs the best when $\text{Sup}_{\text{min}} = 2$ and Interval $= 10$. First, the setting of $\text{Sup}_{\text{min}}$ has a significant influence on the results of BGNAS, and BGNAS can obtain relatively good performance when $\text{Sup}_{\text{min}}$ is equal to 2 or 3. Second, the proposed BGNAS can achieve relatively good performance

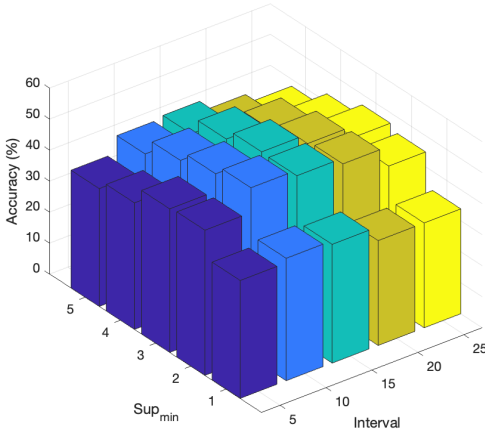



Fig. 7. Influence analysis of hyperparameters ($\text{Sup}_{\text{min}}$ and Interval) on BGNAS by the accuracy values.

when the iteration interval is set to 10, and it is not necessary to set Interval to 5, as this results in a slightly lower accuracy but with higher computational cost. Therefore, the current parameter settings for BGNAS are effective.

## VI. Conclusion

In this article, we have presented a BGNAS algorithm for text classification, where the bipartite graph-based search space can effectively describe the topological order of candidate operators for text representation. Moreover, we use an EA to gradually reduce the scale of bipartite graph-based search space, with the introduction of the approximated submodular function for an RL search algorithm in NAS. The reduced bipartite graph-based search space is more succinct, efficiently simplifying the computational complexity of BGNAS. The effectiveness of BGNAS is evaluated on eight public datasets in both traditional DAG search space and bipartite graph-based search space. Furthermore, it has been shown that the proposed bipartite graph search space effectively represents a set of candidate operators for capturing the complex contextual semantics of text data, thereby enhancing generalization capability.

Although BGNAS can obtain effective and efficient neural architectures for text classification, there is still much room for improvement. To achieve better performance on other text-related tasks, BGNAS should be further developed to capture the complex contextual semantics of text data with lower computational cost. In addition, to enhance the generalization capability of the bipartite graph-based search space, BGNAS should explore its application in real-world text-related tasks. In the future, BGNAS can potentially be integrated with graph neural networks to refine their structure for multimodal tasks or specific domains.

## References

[1] A. Yang, Y. Liu, C. Li, and Q. Ren, "Deeply supervised block-wise neural architecture search," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jan. 17, 2024, doi: 10.1109/TNNLS.2023.3347542.

[2] Y. Jaafra, J. L. Laurent, A. Deruyver, and M. S. Naceur, "Reinforcement learning for neural architecture search: A review," *Image Vis. Comput.*, vol. 89, pp. 57–66, Sep. 2019.

[3] X. Yan, H. Huang, Y. Jin, L. Chen, Z. Liang, and Z. Hao, "Neural architecture search via multi-hashing embedding and graph tensor networks for multilingual text classification," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 8, no. 1, pp. 350–363, Feb. 2024.

[4] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," 2018, *arXiv:1806.09055*.

[5] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 2, pp. 550–570, Feb. 2023.

[6] H. Huang, J. Su, Y. Zhang, and Z. Hao, "An experimental method to estimate running time of evolutionary algorithms for continuous optimization," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 275–289, Apr. 2020.

[7] Z. Lu, R. Cheng, Y. Jin, K. C. Tan, and K. Deb, "Neural architecture search as multiobjective optimization benchmarks: Problem formulation and performance assessment," *IEEE Trans. Evol. Comput.*, vol. 28, no. 2, pp. 323–337, Apr. 2022.

[8] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for NAS," 2019, *arXiv:1912.06059*.

[9] Z. Lu, R. Cheng, S. Huang, H. Zhang, C. Qiu, and F. Yang, "Surrogate-assisted multiobjective neural architecture search for real-time semantic segmentation," *IEEE Trans. Artif. Intell.*, vol. 4, no. 6, pp. 1602–1615, Jun. 2022.

[10] H. Tan et al., "RelativeNAS: Relative neural architecture search via slow-fast learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 1, pp. 475–489, Jan. 2023.

[11] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*.

[12] Y. Wang et al., "TextNAS: A neural architecture search space tailored for text representation," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 5, pp. 9242–9249.

[13] H. Zhu and Y. Jin, "Real-time federated evolutionary neural architecture search," *IEEE Trans. Evol. Comput.*, vol. 26, no. 2, pp. 364–378, Apr. 2022.

[14] C. Guan, X. Wang, and W. Zhu, "AutoAttend: Automated attention representation search," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 3864–3874.

[15] S. Liu, H. Zhang, and Y. Jin, "A survey on computationally efficient neural architecture search," *J. Autom. Intell.*, vol. 1, no. 1, Dec. 2022, Art. no. 100002.

[16] P. Ren et al., "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Comput. Survey*, vol. 54, no. 4, pp. 1–34, 2021.

[17] X. Dong and Y. Yang, "NAS-Bench-201: Extending the scope of reproducible neural architecture search," 2020, *arXiv:2001.00326*.

[18] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 1, pp. 1997–2017, 2019.

[19] X. Chen, L. Xie, J. Wu, L. Wei, Y. Xu, and Q. Tian, "Fitting the search space of weight-sharing NAS with graph convolutional networks," in *Proc. AAAI Conf. Artif. Intell.*, May 2021, vol. 35, no. 8, pp. 7064–7072.

[20] Y. Hu, X. Wang, and Q. Gu, "PWSNAS: Powering weight sharing NAS with general search space shrinking framework," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 11, pp. 9171–9184, Nov. 2023.

[21] Z. Sun et al., "AGNAS: Attention-guided micro and macro-architecture search," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 20777–20789.

[22] K. Babić, S. Martinčić-Ipšić, and A. Meštrović, "Survey of neural text representation models," *Information*, vol. 11, no. 11, p. 511, Oct. 2020.

[23] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Guest editorial evolutionary neural architecture search," *IEEE Trans. Evol. Comput.*, vol. 28, no. 3, pp. 566–569, Jun. 2024.

[24] X. Su et al., "Prioritized architecture sampling with monto-carlo tree search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 10963–10972.

[25] M. Ikonomakis, S. Kotsiantis, and V. Tampakas, "Text classification using machine learning techniques," *WSEAS Trans. Comput.*, vol. 4, no. 8, pp. 966–974, 2005.

[26] X. Li et al., "Improving one-shot NAS by suppressing the posterior fading," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jul. 2020, pp. 13836–13845.

[27] S. Yang, X. Yu, Y. Tian, X. Yan, H. Ma, and X. Zhang, "Evolutionary neural architecture search for transformer in knowledge tracing," 2023, *arXiv:2310.01180*.

[28] X. Wu et al., "Neural architecture search for text classification with limited computing resources using efficient Cartesian genetic programming," *IEEE Trans. Evol. Comput.*, vol. 28, no. 3, pp. 638–652, Jun. 2024.

[29] Y. Hu et al., "Angle-based search space shrinking for neural architecture search," in *Proc. Eur. Conf. Comput. Vis.*, Glasgow, U.K. Cham, Switzerland: Springer, 2020, pp. 119–134.

[30] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep learning–based text classification: A comprehensive review," *ACM Comput. Surv.*, vol. 54, no. 3 pp. 1–40, Apr. 2021.

[31] F. Menczer, D. Crandall, Y.-Y. Ahn, and A. Kapadia, "Addressing the harms of AI-generated inauthentic content," *Nature Mach. Intell.*, vol. 5, no. 7, pp. 679–680, Jul. 2023.

[32] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: A statistical framework," *Int. J. Mach. Learn. Cybern.*, vol. 1, pp. 43–52, Aug. 2010.

[33] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, Dec. 2013, pp. 3111–3119.

[34] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.

[35] Y. Kim, "Convolutional neural network for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1746–1751.

[36] Y. Wen, W. Zhang, R. Luo, and J. Wang, "Learning text representation using recurrent convolutional neural network with highway layers," 2016, *arXiv:1606.06905*.

[37] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. AAAI Conf. Artif. Intell.*, Jan. 2018, pp. 1–8.

[38] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[39] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Jun. 2017, pp. 5998–6008.

[40] S. Santra, J.-W. Hsieh, and C.-F. Lin, "Gradient descent effects on differential neural architecture search: A survey," *IEEE Access*, vol. 9, pp. 89602–89618, 2021.

[41] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1294–1303.

[42] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 4095–4104.

[43] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2018, vol. 32, no. 1, pp. 1–8.

[44] H. Zhang, K. Hao, L. Gao, X.-S. Tang, and B. Wei, "Enhanced gradient for differentiable architecture search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 7, pp. 9606–9620, Jul. 2024.

[45] Y. Feng, Z. Lv, H. Chen, S. Gao, F. An, and Y. Sun, "LRNAS: Differentiable searching for adversarially robust lightweight neural architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 3, 2024, doi: 10.1109/TNNLS.2024.3382724.

[46] Z. Ding, Y. Chen, N. Li, D. Zhao, Z. Sun, and C. L. Philip Chen, "BNAS: Efficient neural architecture search using broad scalable architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 9, pp. 5004–5018, Sep. 2021.

[47] X. Han, Y. Xue, Z. Wang, Y. Zhang, A. Muravev, and M. Gabbouj, "SaDENAS: A self-adaptive differential evolution algorithm for neural architecture search," *Swarm Evol. Comput.*, vol. 91, Dec. 2024, Art. no. 101736.

[48] X. Chu, B. Zhang, H. Ma, R. Xu, and Q. Li, "Fast, accurate and lightweight super-resolution with neural architecture search," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 59–64.

[49] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," 2019, *arXiv:1908.09791*.

[50] C. Peng, Y. Li, R. Shang, and L. Jiao, "ReCNAS: Resource-constrained neural architecture search based on differentiable annealing and dynamic pruning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 2, pp. 2805–2819, Feb. 2022.

[51] N. Nayman, A. Noy, T. Ridnik, I. Friedman, R. Jin, and L. Zelnik, "XNAS: Neural architecture search with expert advice," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, Jun. 2019, pp. 1975–1985.

[52] Y. Wang et al., "MedNAS: Multiscale training-free neural architecture search for medical image analysis," *IEEE Trans. Evol. Comput.*, vol. 28, no. 3, pp. 668–681, Jun. 2024.

[53] Y. Fan, F. Tian, Y. Xia, T. Qin, X.-Y. Li, and T.-Y. Liu, "Searching better architectures for neural machine translation," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 1574–1585, 2020.

[54] L. Chen, X. Yan, Z. Wang, and H. Huang, "Neural architecture search with heterogeneous representation learning for zero-shot multi-label text classification," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2023, pp. 1–8.

[55] X. Zhou, L. Feng, X. Wu, Z. Lu, and K. C. Tan, "Design principle transfer in neural architecture search via large language models," 2024, *arXiv:2408.11330*.

[56] X. Yuan, "An improved apriori algorithm for mining association rules," in *Proc. AIP Conf.*, vol. 1820, 2017, pp. 1–7.

[57] H. Zhang and Y. Vorobeychik, "Submodular optimization with routing constraints," in *Proc. AAAI Conf. Artif. Intell.*, Feb. 2016, vol. 30, no. 1, pp. 1–8.

[58] C. Bian, C. Feng, C. Qian, and Y. Yu, "An efficient evolutionary algorithm for subset selection with general cost constraints," in *Proc. 24th AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 3267–3274.

[59] A. Hashemi, M. Ghasemi, H. Vikalo, and U. Topcu, "Submodular observation selection and information gathering for quadratic models," in *Proc. Int. Conf. Mach. Learn.*, May 2019, pp. 2653–2662.

[60] T. Soma and Y. Yoshida, "A generalization of submodular cover via the diminishing return property on the integer lattice," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, Dec. 2015, pp. 847–855.

[61] R. Socher et al., "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. Conf. Empir. Methods Nat. Lang. Process.*, Oct. 2013, pp. 1631–1642.

[62] J. Lehmann et al., "DBpedia—A large-scale, multilingual knowledge base extracted from Wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[63] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. Adv. Neural Inf. Process. Syst.*, Sep. 2015, pp. 1–9.

[64] J. McAuley and J. Leskovec, "Hidden factors and hidden topics: Understanding rating dimensions with review text," in *Proc. 7th ACM Conf. Recommender Syst.*, Oct. 2013, pp. 165–172.

[65] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," 2016, *arXiv:1607.01759*.

[66] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," 2016, *arXiv:1606.01781*.

[67] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," 2017, *arXiv:1705.02364*.

[68] G. Bender, P.-J. Kindermans, B. Zoph, V. Vasudevan, and Q. Le, "Understanding and simplifying one-shot architecture search," in *Proc. Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 550–559.

[69] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in *Proc. Uncertainty Artif. Intell.*, 2020, pp. 367–377.

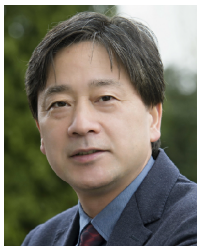
**Xueming Yan** (Member, IEEE) received the Ph.D. degree in computer science from the South China University of Technology, Guangzhou, China, in 2017.

She is currently an Associate Professor with the School of Information Science and Technology and the School of Cyber Security, Guangdong University of Foreign Studies, Guangzhou. Her research interests include graph neural networks, optimization, and natural language processing.

**Han Huang** (Senior Member, IEEE) received the B.Sc. degree in information management and information system from the School of Mathematics, South China University of Technology (SCUT), Guangzhou, China, in 2003, and the Ph.D. degree in computer science from SCUT in 2008.

He is currently a Full Professor with the School of Software Engineering, SCUT. His research interests include theoretical foundation and application of evolutionary computation and microcomputation.

Prof. Huang is a Distinguished Member of CCF.



**Yaochu Jin** (Fellow, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1988, 1991, and 1996, respectively, and the Dr.-Ing. degree from Ruhr University Bochum, Bochum, Germany, in 2001.

He was an Alexander von Humboldt Professor for Artificial Intelligence endowed by German Federal Ministry of Education and Research, with the Faculty of Technology, Bielefeld University, Bielefeld, Germany. He was also a "Finland Distinguished Professor" with the University of Jyväskylä, Jyväskylä, Finland; a "Changjiang Distinguished Visiting Professor" with Northeastern University, Shenyang, China; and a "Distinguished Visiting Scholar" with the University of Technology Sydney, Sydney, NSW, Australia. He is currently the Chair Professor of AI with the School of Engineering, the Head of the Department of Artificial Engineering, and leads the Trustworthy and General AI Laboratory, Westlake University, Hangzhou, China. He is also a Surrey Distinguished Chair and a Professor of computational intelligence, Department of Computer Science, University of Surrey, Guildford, U.K. His main research interests include evolutionary optimization and learning, trustworthy machine learning and optimization, and evolutionary developmental AI.

Prof. Jin is a member of the Academia Europaea. He was a recipient of the 2015, 2017, and 2020 *IEEE Computational Intelligence Magazine* Outstanding Paper Award; the 2018, 2021, and 2024 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award; and the 2025 IEEE Frank Rosenblatt Award. He is currently the President of the IEEE Computational Intelligence Society and the Editor-in-Chief of *Complex and Intelligent Systems*. He was named as a "Highly Cited Researcher" consecutively from 2019 to 2024 by Clarivate.

**Zilong Wang** received the B.S. degree in software engineering from the South China University of Technology, Guangzhou, China, in 2023, where he is currently pursuing the master's degree with the School of Software Engineering.

His main research interests include machine learning and natural language processing.

**Zhifeng Hao** (Senior Member, IEEE) received the B.Sc. degree in mathematics from Sun Yatsen University, Guangzhou, China, in 1990, and the Ph.D. degree in mathematics from Nanjing University, Nanjing, China, in 1995.

He is currently a Professor with the College of Mathematics and Computer Science, Shantou University, Shantou, Guangdong, China. His current research interests include various aspects of algebra, machine learning, data mining, and evolutionary algorithms.