Open-Book Neural Algorithmic Reasoning

Hefei Li, Chao Peng*, Chenyang Xu*, Zhengfeng Yang
Shanghai Key Laboratory of Trustworthy Computing
Software Engineering Institute
East China Normal University, Shanghai, China
51255902127@stu.ecnu.edu.cn,
{cpeng, cyxu, zfyang}@sei.ecnu.edu.cn

Abstract

Neural algorithmic reasoning is an emerging area of machine learning that focuses on building neural networks capable of solving complex algorithmic tasks. Recent advancements predominantly follow the standard supervised learning paradigm – feeding an individual problem instance into the network each time and training it to approximate the execution steps of a classical algorithm. We challenge this mode and propose a novel open-book learning framework. In this framework, whether during training or testing, the network can access and utilize all instances in the training dataset when reasoning for a given instance.

Empirical evaluation is conducted on the challenging CLRS Algorithmic Reasoning Benchmark, which consists of 30 diverse algorithmic tasks. Our open-book learning framework exhibits a significant enhancement in neural reasoning capabilities. Further, we notice that there is recent literature suggesting that multi-task training on CLRS can improve the reasoning accuracy of certain tasks, implying intrinsic connections between different algorithmic tasks. We delve into this direction via the open-book framework. When the network reasons for a specific task, we enable it to aggregate information from training instances of other tasks in an attention-based manner. We show that this open-book attention mechanism offers insights into the inherent relationships among various tasks in the benchmark and provides a robust tool for interpretable multi-task training.

1 Introduction

Deep neural networks have achieved remarkable advancements in various areas, such as image processing [18, 6] and natural language processing [16, 21]. In recent years, as deep learning continues to evolve, there has been an increasing desire to see deep neural networks take on more complex tasks. Algorithmic reasoning tasks [27, 5, 28] have emerged as a particularly crucial category. In classical domains, deep neural networks have demonstrated their ability to learn predictive patterns from training data. The aspiration now is to extend this capability to the field of algorithmic reasoning, which motivates a burgeoning domain — *Neural Algorithmic Reasoning* (NAR).

Neural algorithmic reasoning was initially coined by [30]. The central objective of this domain is to develop and train neural networks with the capability to imitate classical rule-based algorithms, such as sorting algorithms and graph algorithms. Networks built in this manner demonstrate the ability to perform algorithmic computations similar to traditional algorithms in reasoning tasks, while showcasing improved computational efficiency compared to them [17]. Moreover, recent literature [31, 22] shows that owing to the characteristics of deep learning, these networks exhibit flexibility in handling diverse input formats, making them robust even in scenarios where certain input features are missing.

*Correspondence to Chao Peng and Chenyang Xu. The authors are ordered alphabetically.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-0314

Challenging Benchmark for NAR. CLRS Algorithmic Reasoning Benchmark proposed by [26] is currently the most popular and definitive benchmark for evaluating the algorithmic capabilities of neural networks. This benchmark comprises 30 diverse algorithmic reasoning tasks extracted from the foundational algorithms textbook "Introduction to Algorithms" [4], including sorting, searching, dynamic programming, graph algorithms, string algorithms, and more. Beyond the task diversity, another notable challenge of this benchmark is the *significant differences* in scale between problem instances in the training and test sets. The test instances are substantially larger in scale compared to those in the training set.

There have been many recent advances in exploring CLRS [7, 19, 2, 8, 24, 3]. As classical algorithms can often be represented by graph structures, several successful approaches leverage the Graph Neural Network (GNN) framework, including models such as PGN [29] and MPNN [9]. In addition to directly applying these classical GNNs, the literature has observed that the execution of some classical algorithms often relies on specific data structures. Consequently, there have been proposals to integrate classical GNNs with data structures like priority queues [12] or stacks [14] to enhance neural reasoning capabilities.

However, we notice that all prior approaches predict algorithmic executions based solely on their parameters and the features of a single input. Although this mode is commonly used in traditional supervised learning tasks [20, 1], it may not be well-suited for NAR due to the inherent difference between complicated reasoning tasks and traditional tasks like image processing. In practical scenarios, when recognizing images, extensive background knowledge is typically not required; but when faced with complex reasoning tasks, a substantial amount of background knowledge is often necessary to complete various aspects of the reasoning process. In such situations, having real-time illustrative examples or formulas available for reference can significantly reduce our memory burden, thereby enhancing task completion. This naturally raises a question:

If allowing a neural network to access additional examples for reference during reasoning, will its reasoning capability improve as a result?

1.1 Our Contributions

We explore the aforementioned question and introduce open-book neural algorithmic reasoning. In this model, the neural architecture is enhanced with an additional memory component that stores representations of instances in the training dataset. Whether during training or testing, whenever the network engages in reasoning for a specific instance, it has the capability to leverage this supplementary memory to aggregate information from other instances within the training set, akin to an open-book exam. The main results of the paper are summarized as follows:

- We present a general framework for open-book NAR. This framework builds upon the foundation of previous NAR architectures by introducing two additional modules for embedding and information aggregation from the training set, and can seamlessly integrate with existing methods. We further provide a detailed implementation of the framework, which is grounded in the cross-attention mechanism. This design not only caters to single-task training but also proves to be highly effective in scenarios involving multi-task training.
- Empirical evaluations are conducted on the challenging CLRS Benchmark [26]. We incorporate the proposed framework with three popular network architectures in the literature. The results demonstrate that each architecture's reasoning capability can be improved significantly when utilizing the training instances through the framework. Across the majority of the reasoning tasks within the benchmark, the framework yields state-of-the-art results.
- Multi-task training is also investigated in the paper. As highlighted in [11], on certain reasoning tasks, a generalist network trained on all datasets in CLRS outperforms the networks trained in a single-task manner. We provide an interpretation of this observation using the proposed open-book framework. Specifically, when training a neural network to solve a task, we input information from other task datasets into the framework for its use. The results show that our open-book framework can nearly replicate the effects of multi-task training for each algorithmic task, while in some tasks, it even achieves higher accuracies. Additionally, our attention-based implementation enables us to analyze the attention weights of various tasks, facilitating a deeper understanding of the intrinsic relationships among

tasks. A "paired training" experiment is further conducted to verify the effectiveness of the learned attention weights.

1.2 Other Related Work

Our work is closely aligned with the exploration of non-parametric models [23, 25, 13], where models abstain from training specific parameters and, instead, utilize dependencies among training data points for predictions. Our framework can be viewed as a fusion of deep neural networks and non-parametric models. We have noted analogous efforts in recent work within the field of image processing [15]. This work focuses on the CIFAR-10 dataset, employing self-attention mechanisms among different points in the dataset to finish image classification tasks.

2 Preliminaries

This section introduces the setting of an NAR dataset formally and outlines the standard paradigm employed in NAR.

NAR Dataset. The objective of an NAR task is to train a neural network such that it can imitate each execution step of a classical algorithm on given problem instances. Hence, a NAR dataset is labeled by a specific problem and the algorithm employed to solve it. Each data point includes a problem instance, represented by a graph structure, and the corresponding algorithm execution on that instance, conveyed through a sequence of graph-structured states. Denote by $\mathbf{x}$ the problem instance and by $\mathbf{y} = \{\mathbf{y}^{(1)}, ..., \mathbf{y}^{(t)}, ...\}$ the algorithm execution, where $\mathbf{y}^{(t)}$ signifies the graph-structured states (e.g., the current nodes in the queue of breadth-first search) at the t-th step of the algorithm.

Training Objective. The training objective of the neural network is to perform sequential reasoning tasks over a given problem instance. At each step t, the network takes as input the pair $\left(\mathbf{x}, \mathbf{y}^{(t-1)}\right)$ and produces the output $\mathbf{y}^{(t)}$. This process enables the neural network to learn and predict the evolution of the algorithmic execution on the problem instance in a step-wise fashion.

Encode-Processor-Decode Paradigm. To achieve the aforementioned step-wise objective, the literature follows the standard *encode-process-decode* paradigm [10], which consists of three modules: Encoder, Processor, and Decoder. At each step t, the input $\left(\mathbf{x}, \mathbf{y}^{(t-1)}\right)$ traverses through these modules sequentially[2]:

- The encoder module encompasses multiple neural networks that operate on $\left(\mathbf{x}, \mathbf{y}^{(t-1)}\right)$, thereby transforming it into a collection of graph-structured hidden states. Use $G = (V, E)$ to denote the graph structure. Following this module, we obtain h_v corresponding to each node $v \in V$, h_{vu} associated with each edge $(v, u) \in E$, and h_g representing the hidden state of the entire graph G.
- The processor module usually consists of a graph neural network. This module maintains the historical hidden states of nodes, edges, and the graph: $\{h_v^{(t-1)}\}_{v \in V}$, $\{h_{vu}^{(t-1)}\}_{(v,u) \in E}$, $h_g^{(t-1)}$, and integrate them with the newly generated states $\{h_v\}$, $\{h_{v,u}\}$, h_g to yield updated states. We borrow the language of the message-passing architecture [9] to formalize this process. For brevity, the following focuses only on updating the state of each node v. At each step t, the node computes and aggregates messages m_{uv} from its incoming edges, updating its own hidden state:

$$z_v^{(t)} \leftarrow f_1\left(h_v, h_v^{(t-1)}\right) \; ; \quad m_{uv} \leftarrow f_2\left(z_v^{(t)}, z_u^{(t)}, h_{uv}, h_g\right) \quad \forall (u, v) \in E \; ;$$
$$M_v \leftarrow \bigoplus_{u:(u,v) \in E} m_{uv} \; ; \qquad h_v^{(t)} \leftarrow f_3\left(z_v^{(t)}, M_v\right).$$

  Different processors employ different layers f_1, f_2, f_3, and aggregation function $\bigoplus$.
- The decoder module utilizes the states $\mathbf{h}^{(t)}$ as input to forecast the algorithmic execution $\mathbf{y}^{(t)}$ at step t. It is noteworthy that recent literature [11] also incorporates $\mathbf{x}$ and $\mathbf{y}^{(t-1)}$ within this module.

[2]For simplicity, we abuse the notion slightly, allowing $\mathbf{y}^{(t-1)}$ to represent the outcome of the last step.

https://doi.org/10.52202/079017-0314

3 Open-Book Reasoning

The paradigm above can be denoted by a function $\mathcal{F}$ mapping $\mathbf{x}$ to $\mathbf{y}$ for each data point. Given a NAR dataset, this function implies a standard supervised learning mode: during a training step, a (or a mini-batch of) random datapoint $(\mathbf{x}, \mathbf{y})$ is selected. The loss between $\mathcal{F}(\mathbf{x})$ and $\mathbf{y}$ is then computed, and the parameters in $\mathcal{F}$ are updated accordingly. In this section, we go beyond the individual $\mathbf{x} \rightarrow \mathbf{y}$ mode in conventional supervised learning, exploring a more general and practical learning paradigm.

3.1 Framework

We introduce an open-book reasoning framework. Within the framework, when the network is tasked with solving problem instance $\mathbf{x}$ and deducing $\mathbf{y}$, it not only utilizes $\mathbf{x}$ as input but is also allowed to leverage information from other data points within the training set during the reasoning process.

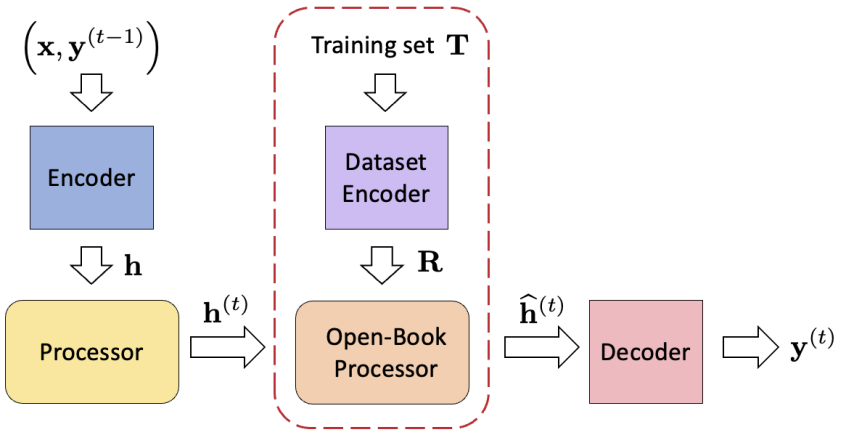

Figure 1: An illustration of the open-book framework. At each reasoning step t, we simultaneously input $(\mathbf{x}, \mathbf{y}^{(t-1)})$ and instances from the training set $\mathbf{T}$, yielding $\mathbf{y}^{(t)}$.

The intuition behind the open-book framework is analogous to our real-world scenario of solving algorithmic problems or engaging in other reasoning tasks. In practical situations, we often consult textbooks and refer to example problems to aid in completing tasks. Typically, the structure and solutions of these examples provide substantial assistance in our reasoning process. Denoting the training set as $\mathbf{T}$, the framework essentially aims to learn a comprehensive function $\mathcal{F} : \mathbf{x} \cup \mathbf{T} \rightarrow \mathbf{y}$.

An illustration of the framework is present in Figure 1. In addition to the original three modules, we introduce two new modules: *Dataset Encoder* and *Open-Book Processor*:

- The dataset encoder module employs an encoding function f_E to compute the latent feature of each data point $\mathbf{d}_i = (\mathbf{x}_i, \mathbf{y}_i)$ in the training set: $\mathbf{r}_i \leftarrow f_E(\mathbf{x}_i, \mathbf{y}_i)$. It is worth noting that this encoder module is essentially different from the original one. It maps an entire data point $\mathbf{d}_i = (\mathbf{x}_i, \mathbf{y}_i)$, encompassing the ground truth of each node (and edge) at each step, into a single representation $\mathbf{r}_i$.
- The open-book processor module is incorporated between the original processor and decoder modules. The output $\mathbf{h}^{(t)}$ from the processor no longer directly feeds into the decoder; instead, it passes through the open-book processor, where it undergoes information aggregation with the training data representation $\mathbf{R} = \{\mathbf{r}_1, ..., \mathbf{r}_i, ...\}$ (generated by the dataset encoder). Subsequently, the open-book processor produces the latent features $\widehat{\mathbf{h}}^{(t)}$ required by the decoder. Formally, for each node $v \in V$, $\widehat{h}_v^{(t)} \leftarrow f_P\left(h_v^{(t)}, \mathbf{R}\right)$.

The central component of the framework is the open-book processor module. Within this module, the current problem instance to be solved is integrated with examples from the training set. It is crucial to acknowledge that this integration has both advantages and disadvantages. While it often enhances the architecture's reasoning capabilities, there are instances when it may lead to counterproductive effects, particularly during multi-task training. We will elaborate on this in the experiments.

https://doi.org/10.52202/079017-0314

3.2 Attention-Based Implementation

Diverse implementations within the framework can be achieved by employing different functions for f_E and f_P. For the ease of investigating multitask training, we adopt an attention mechanism-based implementation. A description of the network implementation and training is given in Algorithm 1.

Algorithm 1 Attention-Based Implementation of Open-Book Reasoning

Input: Training set $\mathbf{T}$.

while current epochs $\leq$ maximum training epochs **do**
  Initialize the hidden state $\mathbf{h}^{(0)}$ and $\widehat{\mathbf{h}}^{(0)}$.
  Randomly pick a *target* data point $\mathbf{d} = (\mathbf{x}, \mathbf{y}) \in \mathbf{T}$ and several *auxiliary* data points $\mathbf{d}_1, \mathbf{d}_2, ...\mathbf{d}_\ell \in \mathbf{T}$. ▷ **Dataset Encoder**
  for each auxiliary data point $\mathbf{d}_i = (\mathbf{x}_i, \mathbf{y}_i)$ **do**
    Let $G = (V, E)$ represent the underlying graph of $\mathbf{d}_i$. This data point encompasses a state sequence $< \mathbf{x}_i = \mathbf{y}_i^{(0)}, \mathbf{y}_i^{(1)}, ..., \mathbf{y}_i^{(t)}, ... >$ on the graph.
    Randomly select two adjacent states $\mathbf{y}_i^{(p)}, \mathbf{y}_i^{(p+1)}$ from the sequence.
    for each node $v \in V$ **do**
      Let y_v and y'_v be node v's states in $\mathbf{y}_i^{(p)}$ and $\mathbf{y}_i^{(p+1)}$ respectively.
      Employ a linear layer $z_v \leftarrow \mathsf{linear}\left(\frac{1}{2} \cdot (y_v + y'_v)\right)$.
    end for
    $\mathbf{r}_i \leftarrow \frac{1}{|V|} \cdot \sum_{v \in V} z_v$.
  end for
  Define $\mathbf{R} := [\mathbf{r}_1, ..., \mathbf{r}_\ell]$.
  for each algorithm execution step t of the target data point **do**
    Feed $\mathbf{x}$ and the outcome from the previous step into the encoder and processor sequentially to obtain the hidden states $\mathbf{h}^{(t)}$.
    Employ a linear layer: $\mathbf{R}^{(t)} \leftarrow \mathbf{R} \parallel \mathsf{linear}(\mathbf{h}^{(t)})$. ▷ **Open-Book Processor**
    Set a QKV attention function with linear layers $\mathsf{query}(\cdot), \mathsf{key}(\cdot), \mathsf{value}(\cdot)$.
    Compute a cross-attention between $\mathbf{h}^{(t)}$ and $\mathbf{R}^{(t)}$:

$$\widehat{\mathbf{h}}^{(t)} \leftarrow \mathsf{softmax}\left(\frac{\mathsf{query}\left(\mathbf{h}^{(t)}\right) \cdot \mathsf{key}\left(\mathbf{R}^{(t)}\right)}{\sqrt{d_k}}\right) \cdot \mathsf{value}\left(\mathbf{R}^{(t)}\right),$$

    where d_k is the dimension of the key vectors.
    Add a gate function: $\widehat{\mathbf{h}}^{(t)} \leftarrow \mathsf{gate}\left(\mathbf{h}^{(t)}, \widehat{\mathbf{h}}^{(t)}\right)$.
    Feed $\widehat{\mathbf{h}}^{(t)}$ into the decoder module, yielding the predictions of this step.
  end for
  Compute the prediction loss and update the network parameters.
end while

From the description, a *target* data point and several *auxiliary* data points are randomly selected in each training iteration. The target data point serves as the focal point for neural optimization in this iteration: the network predicts its ground truths, computes the loss, and consequently updates the network parameters. The auxiliary data points assist the network in reasoning for the target data point. Their latent features are obtained through the dataset encoder, and they subsequently influence predictions through the open-book processor.

At each algorithmic step t, the hidden states $\mathbf{h}^{(t)}$ are computed conventionally. However, these states are not directly input into the decoder. Instead, they need to undergo cross-attention with the representations of auxiliary data points within the open-book processor module. This design allows the network to incorporate hints provided by the auxiliary data points during the reasoning process.

The construction of the dataset encoder is a bit subtle. We observe a crucial aspect that all these pieces of information ultimately serve the decoder module. In a single algorithmic step, the decoder's role is to facilitate the transition between two adjacent states throughout the entire reasoning process. Therefore, to better provide effective hints to the final decoder, for each auxiliary data point, we

https://doi.org/10.52202/079017-0314

randomly sample a pair of adjacent states from its corresponding state sequence. Subsequently, we employ a linear layer to yield the latent representations of these data points.

Remark. The testing process is essentially similar to the training. It is worth noting that during the testing phase, the target data points are sourced from the testing set, while the auxiliary data points must still originate from the training set.

4 Experiments

This section evaluates the open-book implementation empirically on the CLRS benchmark. We aim to investigate the following three questions during the experiments:

- For various processor architectures present in the literature, can the open-book framework consistently enhance their empirical performances across the majority of the algorithmic tasks within the CLRS benchmark?
- There is a recent literature [11] proposing a multi-task training approach for CLRS. They train a common network for various tasks in the benchmark and find that some tasks benefit from the multi-task approach, achieving higher accuracy than when trained individually. In the context of the open-book setting, does this phenomenon imply that incorporating training sets from various tasks into the open-book framework may enhance the network's performance on certain tasks?
- Can the attention-based implementation serve as a robust tool for interpretable multi-task training? When integrating training sets from various tasks into the open-book framework for a specific task, the network eventually learns attention weights in the open-book processor, signifying the task's relevance to other tasks. Does this imply that if a task performs better in multi-task training than in single-task training, retaining only those tasks with prominent attention for multi-task training can still outperform single-task training?

To tackle these questions, we conduct three types of experiments[3]: single-task augmenting, multi-task augmenting, and multi-task interpretation. Note that our "multi-task augmenting" experiment differs essentially from traditional multi-task training; here, we still train the network for a specific task, but with the inclusion of datasets from other tasks in the dataset encoder. Additional ablation experiments are also conducted. Due to space limitations, we defer them to the full version of this paper. We initially outline the experimental setup and subsequently delve into each experiment.

4.1 Setup

Baselines. We incorporate the open-book framework into three existing processor architectures: PGN [29], MPNN [9] and Triplet-GMPNN [11]. Given that the feature dimension of hidden states is set to 128 in the literature, we adjust the parameters of the dataset encoder and open-book processor to ensure seamless integration. The results (F1 scores) achieved by open-book reasoning are compared with them. Moreover, we also compare the performance with other recent architectures like Memnet [26] and NPQ [12].

Computational Details. The experiments are conducted on a machine equipped with an i7-13700K CPU, an RTX 4090 GPU, and an RTX A6000 GPU. The results are averaged over 4 runs. To ensure fair comparisons, we follow the widely-used experimental hyperparameter settings in [11], where the batch size is 32 and the network is trained for 10,000 steps by Adam optimizer with a learning rate of 0.001. During each training and testing iteration, we allow Algorithm 1 to sample 240 auxiliary data points and use only one attention head. The average training time for each reasoning task is approximately 0.5 GPU hours.

4.2 Single-Task Augmenting

This subsection considers a single-task environment: for each reasoning task in CLRS, both target and auxiliary data points in Algorithm 1 are sourced from its own dataset. We create comparison charts for results on three existing architectures. Due to space, only one chart Figure 2 is presented

[3]The codes are provided in https://github.com/Hoferlee1/Open-Book

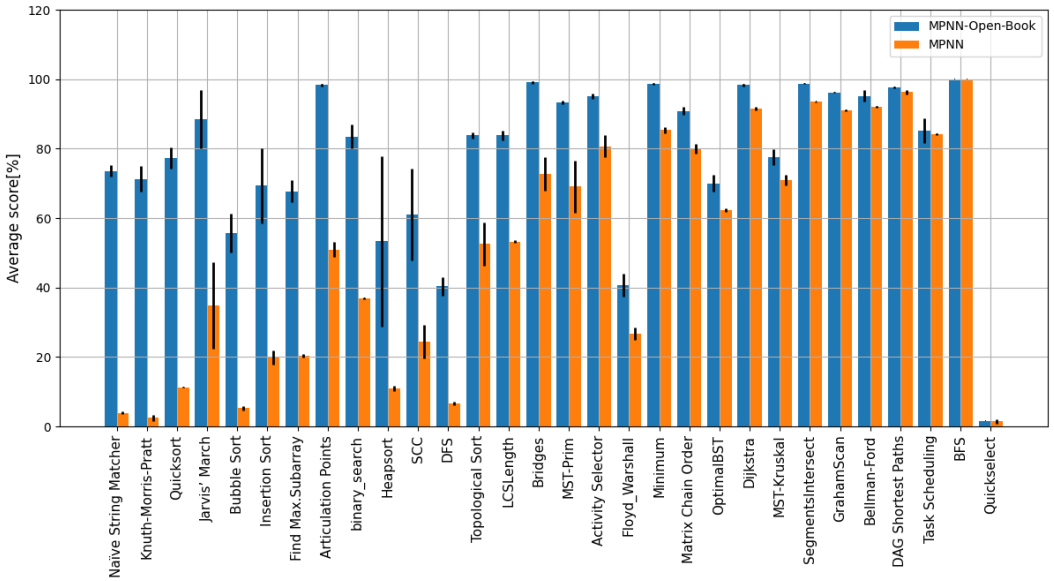

Figure 2: Comparison of the MPNN architecture's performance before and after augmentation with the open-book framework. The 30 tasks are arranged in descending order of improvement magnitude.

Table 1: The summary of our results on each task category in CLRS. The best-performing results in each row are highlighted in bold. To save space, we use the column "Prior Best" to denote the best results among four existing approaches: Memnet [26], PGN [26], MPNN [26], and NPQ [12], and the column "Ours" to denote the best results achieved by applying the open-book framework to the three existing architectures.

Task Category	Prior Best	Triplet-GMPNN	Ours
Graphs	64.98%±2.59	81.41%±1.53	**85.37%±1.73**
Geometry	92.48%±1.35	94.09%±0.77	**96.55%±0.50**
Strings	4.08%±0.57	49.09%±4.78	**72.41%±2.66**
Dynamic Programming	76%±2.47	81.99%±1.30	**82.14%±1.45**
Divide and Conquer	65.23%±2.56	**76.36%±0.43**	74.52%±1.88
Greedy	84.13%±2.59	91.22%±0.40	**93.40%±2.12**
Search	56.11%±0.36	58.61%±1.05	**63.15%±0.90**
Sorting	71.53%±0.97	60.38%±5.25	**83.65%±3.06**

in the main body, while the other two are deferred to the full version of this paper.The figure uses bar charts to illustrate average scores for each task, with standard deviations denoted by black lines. Additionally, we arrange the tasks in descending order of improvement magnitude to better illustrate trends.

We also provide tables to comprehensively compare the accuracies that the open-book framework yields with existing results. In CLRS, the 30 tasks are partitioned into 8 categories: Divide and Conquer, Dynamic Programming, Geometry, Graphs, Greedy, Search, Sorting, and Strings. So we present two tables: one showcasing the performance on the 30 individual tasks and another displaying the average performance for each of the 8 task categories. Due to space constraints, the latter is included in the main body (Table 1), while the former is deferred to the full version of this paper.

From the figures and tables, we observe that our approach outperforms the original architectures in the majority of tasks. The improvements provided by the open-book framework are particularly significant for certain tasks, such as the Naive String Matcher task (see Figure 2). However, we also notice a relatively large standard deviation in performance for some tasks. We attribute this variability to the fact that during testing, we sample data from the training set and input it into the dataset encoder each time. The quality of the sampled data influences the final inference results, leading to performance fluctuations.

https://doi.org/10.52202/079017-0314

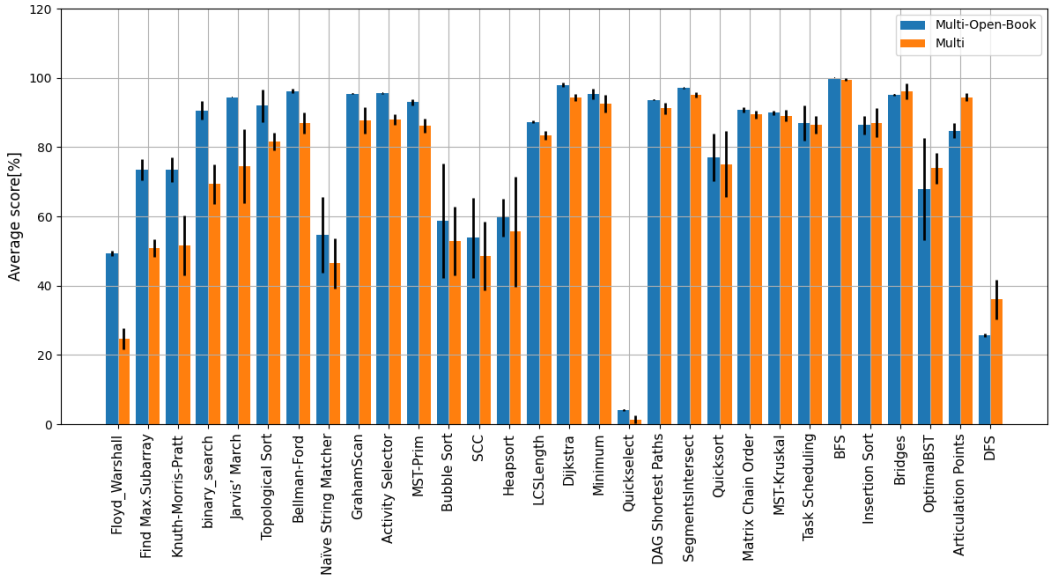

Figure 3: Comparisons between our multi-task augmented approach and Triplet-GMPNN. The 30 tasks are arranged in descending order of improvement magnitude.

4.3 Multi-Task Augmenting

This subsection considers a "multi-task" environment: for each task in CLRS, Algorithm 1 selects target points from its own dataset, while the sampled auxiliary points are drawn from all datasets in CLRS. Since CLRS comprises 30 datasets, in each iteration, we randomly sample 8 instances from each dataset, ensuring that the total number of auxiliary points remains the same as in the single-task experiment, i.e., 240. Given that Triplet-GMPNN is the only architecture used for multi-task training in the literature, both this subsection and the following one "multi-task interpretation" focus exclusively on the results obtained by integrating the open-book framework with Triplet-GMPNN.

The results are present in Figure 3. We find that incorporating data from different tasks into the open-book processor indeed replicates multi-task training. Our multi-task augmented method closely matches the previous multi-task training results, and even outperforms them on the vast majority of tasks. It is worth noting that multi-task training requires simultaneous training on all 30 algorithmic tasks, which is extremely time-consuming. If the goal is simply to enhance performance on a specific task using multi-task training, the cost is substantial. However, with the open-book framework, we can nearly achieve the effects of multi-task training on a target task in approximately the same amount of time it takes to train a single algorithm.

4.4 Multi-Task Interpretation

This subsection delves into interpreting multi-task training. In our multi-task augmenting experiments, the acquired attention weights in the open-book processor reveal the significance of each task in relation to others. Specifically, for each task, we aggregate the attention weights of each node at every algorithmic step on each test instance. The resulting 30-dimensional vector is then normalized, serving as the total attention vector for that task relative to other tasks in the benchmark. Table 2 shows the task with the highest attention weight for each task. Moreover, we present a heatmap regarding the attention weights among CLRS tasks in the full version of this paper.

Surprisingly, the table indicates that the majority of tasks exhibit a preference for attention toward tasks outside their own categories, contrary to our initial expectations. Only four bolded pairs show high attention to tasks within the same category, with most of these being graph algorithms. An intuitive explanation for this phenomenon is that tasks within the same category might not contribute additional information compared to the dataset used for training the task itself. Instead, tasks from other categories seem to play a crucial role in improving training accuracy.

https://doi.org/10.52202/079017-0314

Table 2: For each target (task), we show the task with the highest attention weight among other tasks in column "Auxiliary". We use bold text to indicate when the paired tasks belong to the same algorithmic category.

Target	Auxiliary	Target	Auxiliary
Activity Selector	Topological Sort	Jarvis' March	MST-Kruskal
Articulation Points	Knuth-Morris-Pratt	Knuth-Morris-Pratt	Quicksort
Bellman-Ford	Bridges	LCS Length	Dijkstra
BFS	Task Scheduling	Matrix Chain Order	Jarvis' March
Binary Search	**Quickselect**	Minimum	Quicksort
Bridges	Optimal BST	MST-Kruskal	Heapsort
Bubble Sort	Task Scheduling	**MST-Prim**	**Bridges**
DAG Shortest Paths	Naïve String Matcher	Naïve String Matcher	LCS Length
DFS	Binary Search	Optimal BST	Find Max. Subarray
Dijkstra	**Bellman-Ford**	Quickselect	Dijkstra
Find Max. Subarray	Jarvis' March	Quicksort	BFS
Floyd-Warshall	Heapsort	Segments Intersect	Topological Sort
Graham Scan	Quicksort	SCC	Task Scheduling
Heapsort	Activity Selector	Task Scheduling	Heapsort
Insertion Sort	Minimum	**Topological Sort**	**DAG Shortest Paths**

Table 3: Comparisons among three training manners under Triplet-GMPNN.

Task	Single-Task	Multi-Task	Paired-Task
Heapsort	31.04%±5.82	**55.62%**±15.91	46.63%±10.43
Knuth-Morris-Pratt	19.51%±4.57	51.61%±8.63	**65.67%±12.36**
Insertion Sort	78.14%±4.64	87.00%±4.16	**95.78%±0.80**
LCS Length	80.51%±1.84	83.43%±1.19	**85.86%±1.47**
Quicksort	64.64%±5.12	75.10%±9.52	**88.43%±6.25**
SCC	43.43%±3.15	48.48%±9.96	**73.39%±3.00**
Jarvis'March	91.01%±1.30	74.51%±10.71	**94.44%±0.63**
MST-Kruskal	89.80%±0.77	89.08%±1.64	**90.55%±1.12**
MST-Prim	86.39%±1.33	86.26%±2.08	**92.56%±0.99**
Topological Sort	87.27%±2.67	81.65%±2.53	**87.30%±4.62**
Dijkstra	96.05%±0.60	94.29%±1.04	**97.44%±0.50**
Binary Search	77.58%±2.35	69.30%±5.65	**79.17%±2.79**
Bubble Sort	67.68%±5.50	52.94%±9.96	**70.30%±6.77**
Graham Scan	93.62%±0.91	87.74%±3.87	**94.58%±0.87**
Minimum	97.78%±0.55	92.50%±2.53	**98.32%±0.14**

We proceed to a more in-depth examination of the relationships among tasks learned by the framework. We select a partner for each task according to Table 2 – namely, the task it pays the most attention to. We conduct training and testing in a multi-task manner for each task paired with its chosen partner, and refer to this type of training as paired-task training. In this experiment, we only focus on tasks that either demonstrate accuracy improvements or slight declines in multi-task training compared to single-task training, and train them in the paired manner. The results are given in Table 3. The table validates our hypothesis. On these tasks, paired-task training achieves improvements compared to single-task training, with most tasks even surpassing the performance of multi-task training.

4.5 Experimental Summary

The experiments address the three questions posed at the beginning of the section.

- The open-book framework can significantly enhance the reasoning capabilities of various existing architectures, yielding state-of-the-art results across the majority of tasks in CLRS.
- By feeding data from various tasks into the dataset encoder, the framework can successfully replicate the effects of multi-task training, and in most datasets, even outperform it.

https://doi.org/10.52202/079017-0314

- The attention-based implementation provides a valuable tool for the interpretability of multi-task training. By examining the learned attention weights, we can gain insights into the influences and intrinsic relationships among tasks during multi-task training.

5 Conclusion

This paper considers open-book neural algorithmic reasoning, introducing a novel open-book framework accompanied by an attention-based implementation. Through empirical evaluations, we demonstrate that this implementation not only enhances the reasoning capabilities of the existing architecture but also functions as an effective tool for interpretable learning.

Several interesting direction for future research exist, such as exploring more effective implementations within the open-book framework. Note that although our current implementation demonstrates performance improvements for the majority of tasks in CLRS, there are instances where the open-book approach may yield counterproductive results. Refining the current architecture to ensure performance enhancements across all tasks remains a significant challenge.

Acknowledgements

This work is supported by the National Key Research Project of China under Grant No. 2023YFA1009402, the Scientific and Technological Innovation 2030 Major Projects under Grant 2018AAA0100902, NSFC Programs (62161146001, 62302166, 62372176), Shanghai Key Lab of Trustworthy Computing, Henan Key Laboratory of Oracle Bone Inscription Information Processing (AnYang Normal University), and the Key Laboratory of Interdisciplinary Research of Computation and Economics (SUFE), Ministry of Education.

References

[1] Abeer Aljuaid and Mohd Anwar. Survey of supervised learning for medical image processing. *SN Comput. Sci.*, 3(4):292, 2022.

[2] Beatrice Bevilacqua, Kyriacos Nikiforou, Borja Ibarz, Ioana Bica, Michela Paganini, Charles Blundell, Jovana Mitrovic, and Petar Velickovic. Neural algorithmic reasoning with causal regularisation. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 2272–2288. PMLR, 2023.

[3] Wilfried Bounsi, Borja Ibarz, Andrew Dudzik, Jessica B. Hamrick, Larisa Markeeva, Alex Vitvitskyi, Razvan Pascanu, and Petar Velickovic. Transformers meet neural algorithmic reasoners. *CoRR*, abs/2406.09308, 2024.

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

[5] Andreea Deac, Petar Velickovic, Ognjen Milinkovic, Pierre-Luc Bacon, Jian Tang, and Mladen Nikolic. Neural algorithmic reasoners are implicit planners. In *NeurIPS*, pages 15529–15542, 2021.

[6] Shisheng Deng, Dongping Liao, Xitong Gao, Juanjuan Zhao, and Kejiang Ye. A survey on cross-domain few-shot image classification. In *BigData*, volume 14203 of *Lecture Notes in Computer Science*, pages 3–17. Springer, 2023.

[7] Cameron Diao and Ricky Loynd. Relational attention: Generalizing transformers for graph-structured tasks. In *ICLR*. OpenReview.net, 2023.

[8] Andrew Joseph Dudzik and Petar Velickovic. Graph neural networks are dynamic programmers. In *NeurIPS*, 2022.

[9] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 2017.

[10] Jessica B. Hamrick, Kelsey R. Allen, Victor Bapst, Tina Zhu, Kevin R. McKee, Josh Tenenbaum, and Peter W. Battaglia. Relational inductive bias for physical construction in humans and machines. In *CogSci*. cognitivesciencesociety.org, 2018.

[11] Borja Ibarz, Vitaly Kurin, George Papamakarios, Kyriacos Nikiforou, Mehdi Bennani, Róbert Csordás, Andrew Joseph Dudzik, Matko Bosnjak, Alex Vitvitskyi, Yulia Rubanova, Andreea Deac, Beatrice Bevilacqua, Yaroslav Ganin, Charles Blundell, and Petar Velickovic. A generalist neural algorithmic learner. In *LoG*, volume 198 of *Proceedings of Machine Learning Research*, page 2. PMLR, 2022.

[12] Rishabh Jain, Petar Velickovic, and Pietro Liò. Neural priority queues for graph neural networks. In *The 2023 ICML, Workshop on Knowledge and Logical Reasoning in the Era of Data-driven Learning*, volume 202. PMLR, 2023.

[13] Zhiying Jiang, Yiqin Dai, Ji Xin, Ming Li, and Jimmy Lin. Few-shot non-parametric learning with deep latent variable model. In *NeurIPS*, 2022.

[14] Jonas Jürß, Dulhan Hansaja Jayalath, and Petar Veličković. Recursive algorithmic reasoning. In *The Second Learning on Graphs Conference*, 2023.

 https://doi.org/10.52202/079017-0314

[15] Jannik Kossen, Neil Band, Clare Lyle, Aidan N. Gomez, Thomas Rainforth, and Yarin Gal. Self-attention between datapoints: Going beyond individual input-output pairs in deep learning. In *NeurIPS*, pages 28742–28756, 2021.

[16] Ivano Lauriola, Alberto Lavelli, and Fabio Aiolli. An introduction to deep learning in natural language processing: Models, techniques, and tools. *Neurocomputing*, 470:443–456, 2022.

[17] Yujia Li, Felix Gimeno, Pushmeet Kohli, and Oriol Vinyals. Strong generalization and efficiency in neural programs. *CoRR*, abs/2007.03629, 2020.

[18] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E. Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

[19] Sadegh Mahdavi, Kevin Swersky, Thomas Kipf, Milad Hashemi, Christos Thrampoulidis, and Renjie Liao. Towards better out-of-distribution generalization of neural algorithmic reasoning tasks. *Trans. Mach. Learn. Res.*, 2023, 2023.

[20] Mohammed Amine El Mrabet, Khalid El Makkaoui, and Ahmed Faize. Supervised machine learning: A survey. In *CommNet*, pages 1–10. IEEE, 2021.

[21] Wongyung Nam and Beakcheol Jang. A survey on multimodal bidirectional machine learning translation of image and natural language processing. *Expert Syst. Appl.*, 235:121168, 2024.

[22] Danilo Numeroso, Davide Bacciu, and Petar Velickovic. Dual algorithmic reasoning. In *ICLR*. OpenReview.net, 2023.

[23] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*, volume 3176 of *Lecture Notes in Computer Science*, pages 63–71. Springer, 2003.

[24] Gleb Rodionov and Liudmila Prokhorenkova. Neural algorithmic reasoning without intermediate supervision. In *NeurIPS*, 2023.

[25] Bing Shuai, Gang Wang, Zhen Zuo, Bing Wang, and Lifan Zhao. Integrating parametric and non-parametric models for scene labeling. In *CVPR*, pages 4249–4258. IEEE Computer Society, 2015.

[26] Petar Velickovic, Adrià Puigdomènech Badia, David Budden, Razvan Pascanu, Andrea Banino, Misha Dashevskiy, Raia Hadsell, and Charles Blundell. The CLRS algorithmic reasoning benchmark. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 22084–22102. PMLR, 2022.

[27] Petar Velickovic and Charles Blundell. Neural algorithmic reasoning. *Patterns*, 2(7):100273, 2021.

[28] Petar Velickovic, Matko Bosnjak, Thomas Kipf, Alexander Lerchner, Raia Hadsell, Razvan Pascanu, and Charles Blundell. Reasoning-modulated representations. In *LoG*, volume 198 of *Proceedings of Machine Learning Research*, page 50. PMLR, 2022.

[29] Petar Velickovic, Lars Buesing, Matthew C. Overlan, Razvan Pascanu, Oriol Vinyals, and Charles Blundell. Pointer graph networks. In *NeurIPS*, 2020.

[30] Petar Velickovic, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. Neural execution of graph algorithms. In *ICLR*. OpenReview.net, 2020.

[31] Louis-Pascal A. C. Xhonneux, Andreea Deac, Petar Velickovic, and Jian Tang. How to transfer algorithmic reasoning knowledge to learn new algorithms? In *NeurIPS*, pages 19500–19512, 2021.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our topic and contributions are present clearly in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have a discussion on the limitations in the experimental section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

https://doi.org/10.52202/079017-0314

Justification: The contributions of the paper are experimental.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The experimental section gives all the details needed to reproduce our results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will make our open-source code and datasets publicly available to facilitate others in reproducing the results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental setup and relevant hyperparameters for reproducibility can be found in the experimental section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Our table and figures provide the standard deviation.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

https://doi.org/10.52202/079017-0314

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: In the experimental section of this paper, we have specified the model of the graphics card used in our experiments, and further discussed the duration of the model training.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The paper is just a standard experimental study for a public benchmark.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The data and model used in the paper are properly respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

https://doi.org/10.52202/079017-0314

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.