Learning Task Design Based on Fuzzy Logical Reasoning

Abstract

Designing appropriate algorithm architectures for diverse task objectives is an extremely challenging task. It requires algorithm engineers to possess sufficient professional expertise and a thorough understanding of both data and task contexts, and in practice, it is difficult for humans to execute this process comprehensively enough. Although automated algorithm design has been extensively studied, it acts as a black box, lacking interpretability, relies on large-scale search, and suffers from low efficiency. Based on the human triadic thinking model, this work shifts automated algorithm design from a search-driven paradigm to a logical reasoning paradigm. We attempt to extract logical rules from data, deduce algorithm structures from these rules, and employ the derived algorithms to complete a closed learning loop of abduction-deduction-induction for model learning, thus constructing the basic framework for algorithm design via logical reasoning. Specifically, we first mine latent knowledge from data with the support of a knowledge base and calculate the membership degree of rules corresponding to various operators. Next, we infer the algorithm structures that meet the target requirements through fuzzy logic reasoning. Finally, we fit the data and complete the learning process using the obtained algorithm architecture. As a frontier attempt to transform the search-based automated algorithm design pipeline into a reasoning-centric one, this approach achieves outstanding experimental results in operator combination design tasks across multiple datasets.

Keywords

Metacognitive Ability, Fuzzy Logic, Algorithm Design, Neuro-Symbolic

ACM Reference Format:

. 2018. Learning Task Design Based on Fuzzy Logical Reasoning. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Humans possess metacognitive abilities that enable them to automatically design learning plans based on learning objectives. Such metacognitive capabilities represent a high-level pursuit for bringing artificial intelligence closer to human intelligence, with the core goal of enabling models to autonomously master how to learn and, through training on relevant tasks, acquire the ability to adapt to new tasks

In the field of cognitive science, the core foundation of human metacognitive ability stems from three mutually collaborative basic reasoning capabilities: abduction, deduction, and induction [19]. These three abilities do not exist in isolation; instead, they form a closed-loop reasoning process that constitutes the underlying logical framework of human cognitive activity. For a given task, abduction enables the identification of the most plausible rules from real-world observations as explanations [2, 24]; deduction allows reasoning based on existing rules to derive feasible plans that achieve the goal [20]; and induction facilitates the accumulation of experience from implemented plans, thereby endowing the ability to accomplish tasks [13, 17].

In the field of artificial intelligence, empiricism based on machine learning enables models to acquire inductive ability, while symbolism based on logical reasoning endows models with deductive ability. Research on rule discovery from data, such as inductive logic programming and abductive learning, can further equip models with abductive ability [5, 9, 14]. However, these three capabilities have not been systematically integrated to form a closed loop, resulting in significant limitations in the current improvement of AI's metacognitive ability.

Currently, research in the field of artificial intelligence metacognition mainly focuses on two areas: meta-learning and automated machine learning. However, the process by which algorithms in these two fields automatically acquire learning strategies is a black box, lacking interpretability and exhibiting low efficiency. The former relies on training experience on numerous similar tasks and requires solving complex second-order optimization problems [10]; the latter depends on frequent training and evaluation procedures and involves searching in an exponentially complex space. This paper attempts to transform the approach for artificial intelligence models to acquire metacognitive abilities from complex optimization and search paradigms into a logical reasoning paradigm, realizing the process of extracting rules from limited data, inferring learning strategies from rules, and training models according to the derived strategies.

To realize the aforementioned meta-perceptual ability that enables artificial intelligence to automatically obtain algorithms designed based on actual data through logical reasoning, we first establish a theoretical framework that addresses the problem of evaluating the advantages and disadvantages of algorithms given specific data and targets. The theory indicates that the expected performance of a model trained by an algorithm on the target task depends on the algorithm's satisfiability with respect to the data and its completeness with respect to the target task. Among them, satisfiability represents the degree of matching between the algorithm and the data; the higher the satisfiability, the better the algorithm can perform on the pre-trained data. However, good performance on pre-training tasks does not necessarily ensure that the trained model will still perform well on downstream target tasks. For this reason, we use completeness to represent the contribution of pre-training tasks to the target task. Based on the satisfiability and completeness of an algorithm, we can estimate the performance of the model trained by this algorithm on the target task, and take this estimated performance as the optimization objective

Conference acronym 'XX, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

to find a sufficiently excellent training algorithm for the model. Supported by this theoretical foundation, we combine practical applications to convert the original theoretical indicators under the ideal data distribution into actual empirical indicators on real data. The performance of the trained model on the target task is directly estimated by the satisfiability and completeness of the algorithm on the pre-training data and target data. Since model pre-training incurs enormous overhead, this scheme of predicting target performance in advance based on theory can bring a very significant improvement in efficiency, especially for pre-training tasks with large-scale data. This process directly estimates the target performance through theory, rather than requiring a complete training and evaluation process like meta-learning and auto-ml to obtain the real target performance. This is precisely the reason why a great deal of resource overhead is saved by logical reasoning compared with other modes.

At the method level, we represent the meta-cognitive process of AI through three stages. In the abduction stage, we extract the satisfiability and completeness rules of the operators in the operator library and the membership degrees of these rules from the data in the form of fuzzy logic. On the basis of the properties of single operators, we calculate the membership degrees of the satisfiability and completeness rules of combined operators through the properties of fuzzy logic [23]. Among them, the satisfiability of combined operators is based on the conjunction logic, and the membership degree of their rules can be calculated by the t_{norm} operator in fuzzy logic; while the completeness is based on the disjunction logic, and the membership degree of their rules can be calculated by the t_{conorm} operator in fuzzy logic, thus completing the process of extracting the rules of single operators and combined operators from the data.

In the deduction stage, based on the rules in the existing rule base, we search for the optimal operator combination with the goal of the optimal expected performance on the target task. For this purpose, we initially tried five solution schemes: Greedy reasoning adds one operator at a time according to the current state, and each time selects the operator that optimizes the current estimated value of the algorithm's target performance; Dynamic programming reasoning discretizes the satisfiability membership degree and completeness membership degree, takes one of the membership degrees as the state, maintains the optimal value of the other membership degree, and finally searches for the optimal solution with the highest estimated performance among all states; Pareto reasoning regards satisfiability and completeness as two optimization objectives, discards the current non-Pareto optimal solutions while adding operators in turn, and searches for the optimal solution among the finally retained solutions; Simulated annealing formulates the problem as a 0-1 optimization problem, takes whether to select each operator as a 0-1 variable for optimization, and uses local inversion as the exploration of the neighborhood; The bisection method performs a binary search on the optimization objective according to the monotonicity of the target, forms a satisfiability solution problem for each target value, and uses the branch and bound method to solve and judge whether the current target value has a solution until it converges to the optimal target value.

In the induction stage, we have obtained the optimal feasible algorithm. We only need to pre-train a machine learning model on this algorithm and fine-tune it on the target data to form a dedicated model for the current target. The trained model can be further used for rule extraction in subsequent tasks to form a closed loop, thereby realizing the artificial intelligence meta-cognitive process based on the abduction-deduction-induction ternary model.

In terms of experiments, we constructed an operator library based on self-supervised learning, containing a total of 115 self-supervised operators. On this basis, we first verified the theoretical part proposed in this paper, proving that the target performance estimation method based on our theory is highly positively correlated with the actual performance verified through experiments. Secondly, we verified the feasibility of the algorithm design based on logical reasoning proposed by us, which achieved excellent performance and efficiency on multiple datasets. Finally, we conducted ablation experiments and sensitivity analysis: the former proved that both satisfiability and completeness rules are necessary, while the latter, focusing on the fuzzy logic operators t_{norm} and t_{conorm} corresponding to the conjunction and disjunction logic, proved that conventional fuzzy logic operators are all applicable under our algorithm framework.

2 Related Work

In the field of cognitive science, the tripartite reasoning framework of abduction, deduction, and induction has been widely recognized as the core foundation of human metacognition. Peirce first systematically proposed the logical relationships among the three reasoning modes [19], emphasizing that abduction is responsible for generating plausible explanations from observations, deduction for deriving logical conclusions from existing rules, and induction for generalizing experience from specific cases to form new knowledge. Subsequent studies further verified that these three reasoning capabilities form a closed-loop system in human cognitive activities, which is the key to humans being able to autonomously design learning plans and adapt to new tasks. However, most of these studies focus on theoretical analysis of human cognitive mechanisms and lack effective methods to map this tripartite reasoning framework to AI systems, making it difficult to translate human metacognitive principles into practical AI capabilities.

In AI research, the acquisition of single reasoning capabilities has been extensively explored, but their systematic integration for metacognition remains insufficient [1, 17, 22]. Empiricism-based machine learning methods, such as deep learning and reinforcement learning, have endowed models with strong inductive capabilities by learning statistical patterns from large-scale data. However, these methods usually lack interpretability and rely heavily on data quality and quantity, making it difficult to form autonomous learning strategies. Symbolism-based approaches, including logical reasoning and knowledge graphs, have realized rigorous deductive reasoning by formalizing domain knowledge into logical rules [18, 20], but they often struggle with the uncertainty and complexity of real-world data, leading to poor generalization. For abductive capability, research such as inductive logic programming (ILP) [9] and abductive learning (ABL) [2, 24] has made progress in extracting logical rules from data, but these methods are mostly limited to specific tasks and fail to form effective collaboration with deductive and inductive capabilities, resulting in fragmented reasoning

processes that cannot support comprehensive meta-cognitive activities.

Currently, meta-learning and automated machine learning (AutoML) are the two mainstream paradigms for realizing AI metacognition, but they have inherent limitations in interpretability and efficiency. Meta-learning aims to enable models to learn "how to learn" by training on a series of similar tasks, thereby quickly adapting to new tasks. However, most meta-learning methods, such as model-agnostic meta-learning (MAML), rely on complex second-order optimization and a large number of similar tasks for pre-training, leading to high computational overhead and poor interpretability of the learned meta-strategies [10]. AutoML automates the design of machine learning pipelines (e.g., feature engineering, model selection, and hyperparameter tuning) through search algorithms [6], but it usually involves exhaustive search in an exponentially complex space, requiring frequent model training and evaluation, which is inefficient especially for large-scale pre-training tasks. Both paradigms treat the process of acquiring learning strategies as a "black box" and fail to leverage logical reasoning to realize interpretable and efficient autonomous learning, which is inconsistent with the core characteristics of human metacognition.

Fuzzy logic has been widely used in handling uncertain information, providing a potential tool for integrating symbolic reasoning with real-world data uncertainty [23]. By introducing membership degrees to represent the ambiguity of concepts, fuzzy logic can effectively model the uncertainty in rule extraction and reasoning. Existing studies have applied fuzzy logic to rule-based systems and logical reasoning, improving the robustness of symbolic methods in real-world scenarios. Few studies have used fuzzy logic operators (e.g., t_{norm} and t_{conorm}) to calculate the reliability of combined rules, which is crucial for realizing efficient reasoning about algorithm performance.

3 Theory Support

To make automated algorithm design based on logical reasoning feasible, we first need to address the question of how to evaluate the quality of an algorithm, which motivates us to explore its theoretical nature. In previous studies on AutoML and meta-learning, evaluating an algorithm requires a complete process of training and testing the algorithm. Since searching for the optimal algorithm inevitably involves numerous rounds of training and testing, we aim to estimate algorithm performance through theoretical approximation instead of actual training and testing. This will significantly reduce the substantial overhead caused by repeated training and evaluation.

First, unlike the traditional supervised learning paradigm, the data available in real-world scenarios is usually unlabeled. We therefore need to perform self-supervised training on such unlabeled data, and we aim to use as effective self-supervised operators as possible, so that the model, after being trained with these operators, can achieve better performance on the target downstream task.

To this end, we introduce theories of self-supervised learning and perform model pre-training by leveraging the consistency of operators applied to data. In self-supervised learning, given an input sample $x \in \mathcal{D}$ and an operator family A, the learning objective is to ensure that the output of the machine learning model f remains consistent under different operators within the same family, i.e.,

$$f(A'(x)) = f(A''(x)), \quad A', A'' \in A,\ x \sim \mathcal{D}_x \tag{1}$$

which is abbreviated as $\text{Consistency}(A, x, f)$.

The ultimate goal of learning is to enable the model to perform best on the target task:

$$f(A'(x)) = y, \quad A' \in A,\ (x, y) \sim \mathcal{D}_{x,y}. \tag{2}$$

In self-supervised learning, the pre-training task is regarded as an intermediate step toward the target task, which can be expressed by the following equation:

$$\begin{aligned} p(y \mid x) = p\left(y \mid \text{Consistency}(A, x, f),\ x\right) \\ \cdot\ p\left(\text{Consistency}(A, x, f) \,\middle|\, x\right) \end{aligned} \tag{3}$$

where $p\left(\text{Consistency}(A, x, f) \,\middle|\, x\right)$ denotes the satisfaction probability of the algorithm consistency. A higher algorithm satisfaction indicates that the algorithm is sufficiently compatible with the data and thus performs better on the pre-training task. By contrast, $p\left(y \mid \text{Consistency}(A, x, f),\ x\right)$ represents the completeness probability of the algorithm consistency. A higher algorithm completeness implies that satisfying consistency yields greater benefits for achieving the target task, and training on the pre-training task can make a larger contribution to the downstream target task. Specifically, the degree of satisfiability defined on the pre-training tasks can be defined as

$$p_{satisfiability}(A, f, \mathcal{D}_{x,y}) = p_{x \sim \mathcal{D}_x}\left(\text{Consistency}(A, x, f) \,\middle|\, x\right) \tag{4}$$

The degree of completeness defined on the pre-training tasks can be defined as

$$\begin{aligned} &p_{completeness}(A, f, \mathcal{D}_{x,y}) \\ =&p_{x,y \sim \mathcal{D}_{x,y}}\left(y \mid \text{Consistency}(A, x, f),\ x\right) \end{aligned} \tag{5}$$

It can be proven that when the pre-training algorithm is A, the performance of the model on the downstream target task is:

$$\begin{aligned} p_{target}(A, f, \mathcal{D}_{x,y}) = p_{satisfiability}(A, f, \mathcal{D}_{x,y}) \\ \circ\ p_{completeness}(A, f, \mathcal{D}_{x,y}) \end{aligned} \tag{6}$$

At this point, we have identified the problem proposed at the beginning of this chapter. We use $p_{\text{target}}(A, f, \mathcal{D}_{x,y})$ as the criterion for evaluating the superiority or inferiority of the pre-training algorithm A, which includes two indicators: $p_{\text{satisfiability}}$ and $p_{\text{completeness}}$.

The final goal thus becomes finding the optimal algorithm A^*:

$$A^* = \arg\max_{A \in \mathcal{A}} p_{target}(A, f, \mathcal{D}_{x,y}) \tag{7}$$

4 Settings

Based on the above theoretical derivation, in practical applications, we are given a target task $\text{Task}_{\text{target}}$, which can be represented by a small set of labeled data $D_{\text{target}} = [(x_{t1}, y_{t1}), \ldots, (x_{tm}, y_{tm})]$ where $(x_{ti}, y_{ti}) \sim \mathcal{D}_{x,y}$, along with a large amount of unlabeled data $D_{\text{pretrain}} = \left[x_{p1}, \ldots, x_{pn}\right]$ where $x_{pi} \sim \mathcal{D}_x$ and an operator library $\mathcal{A}$, the objective is to construct a learning algorithm $A = \alpha_1 \circ \alpha_2 \circ \cdots \circ \alpha_d$ using operators $\alpha_i \in \mathcal{A}$ (where d denotes the depth of the algorithm, i.e., the number of operators). This algorithm A should be trained on D_{pretrain} and achieve optimal performance on $\text{Task}_{\text{target}}$. In this paper, considering that most collectable data D_{pretrain} in real-world scenarios is unlabeled, we adopt

a self-supervised operator library for $\mathcal{A}$. On this basis, our estimation of p_{target} must be grounded in the available data, namely estimating

$$p_{\text{estimation}}(A, f, D_{\text{pretrain}}, D_{\text{target}}) = \hat{p}_{\text{satisfiability}}(A, f, D_{\text{pretrain}}) \circ \hat{p}_{\text{completeness}}(A, f, D_{\text{pretrain}}, D_{\text{target}}) \quad (8)$$

Empirical estimation of satisfiability can be defined as:

$$\hat{p}_{\text{satisfiability}}(A, f, D_{\text{pretrain}}) = p_{x \in D_{\text{pretrain}}}\left(\text{Consistency}(\alpha_i, x, f) \,\middle|\, x\right) \quad (9)$$

Empirical estimation of completeness can be defined as:

$$\hat{p}_{\text{completeness}}(A, f, D_{\text{pretrain}}, D_{\text{target}}) = p_{x, y \in D_{\text{target}},\, x' \in D_{\text{pretrain}}}\left(f(A(x)) = y \,\middle|\, \text{Consistency}(A, x', f),\ x'\right). \quad (10)$$

We use p_e, p_s, p_c to denote $p_{estimation}$, $p_{satisfiability}$, $p_{completeness}$ for simplicity, respectively.

5 Method

The logic-based meta-capability learning method proceeds in three steps: first, it applies abduction to extract rules about the operators in $\mathcal{A}$ from D_{pretrain} based on $\text{Task}_{\text{target}}$, thereby constructing a fuzzy logic-based operator rule base $\mathcal{R}$; second, it uses deduction to infer the algorithm structure A^* that maximizes the membership degree of the conclusion the algorithm can complete the target task based on the fuzzy logic operator rule base $\mathcal{R}$; finally, it trains the inferred algorithm structure A on D_{train} to obtain the target model f^*.

5.1 Abduction

First, we need to extract the rule base $\mathcal{R}$ from D_{pretrain} based on the target task $\text{Task}_{\text{target}}$, with the goal of achieving better performance on $\text{Task}_{\text{target}}$. We first formalize the propositions that can be extracted from the operator library. For each operator $\alpha_i \in \mathcal{A}$ in the operator library, we separately extract fuzzy logic propositions regarding its satisfiability and completeness, where p_s and p_c denote the membership degrees of these two types of propositions, respectively.

For the satisfiability proposition of operator α_i, it is estimated on the unlabeled dataset D_{pretrain}. The corresponding fuzzy logic proposition is:

$$p_s(\alpha_i, f, D_{\text{pretrain}}) : \text{Consistency}(\alpha_i, x, f) \mid x \in D_{\text{pretrain}}. \quad (11)$$

For the completeness proposition of operator α_i, we need to estimate the contribution of completing the pre-training task to accomplishing the target task, i.e., whether the pre-training task is complete. This estimation must be performed jointly with D_{target}, and the corresponding fuzzy logic proposition is:

$$p_c(\alpha_i, f, D_{\text{pretrain}}, D_{\text{target}}) : f(\alpha_i(x)) = y\, x, y \in D_{target} \,\middle|\, \text{Consistency}(\alpha_i, x, f),\ x' \in D_{\text{pretrain}} \quad (12)$$

Thus, we extract the satisfiability and completeness propositions for all operators in the library at very low cost, and estimate the membership degree of each proposition from the data. However, since algorithm design requires finding a good operator combination, estimating the satisfiability and completeness membership degrees separately for every combination would incur extremely high overhead. Fortunately, according to the properties of fuzzy logic rules, the membership degrees of the satisfiability and completeness rules for an operator combination can be computed directly from the membership degrees of the individual operator rules.

First, for satisfiability, there exists the rule

$$(\text{Consistency}(\alpha_i, x, f) \mid x) \text{ and } (\text{Consistency}(\alpha_j, x, f) \mid x) \rightarrow (\text{Consistency}(\alpha_i \circ \alpha_j, x, f) \mid x \in X_{pretrain}) \quad (13)$$

This reveals that the satisfiability of the combined operator depends on the joint satisfiability of each individual operator, which is a "conjunction" relationship in symbolic logic. In fuzzy logic, the logical conjunction operation can be expressed as a t_{norm} operation on membership degrees:

$$t_{\text{norm}}(p_s(a_i), p_s(a_j)) : \text{Consistency}(\alpha_i \circ \alpha_j, x, f) \mid x \quad (14)$$

For completeness, there exists the rule:

$$\left(f(\alpha_i(x)) = y, (x, y) \in D_{target} \,\middle|\, \text{Consistency}(\alpha_i, x, f),\ x' \in D_{\text{pretrain}}\right) \text{ or } \left(f(\alpha_j(x)) = y, (x, y) \in D_{target} \,\middle|\, \text{Consistency}(\alpha_j, x', f),\ x' \in D_{\text{pretrain}}\right) \rightarrow \left(f(\alpha_i \circ \alpha_j(x)) = y, (x, y) \in D_{target} \,\middle|\, \text{Consistency}(\alpha_i \circ \alpha_j, x', f),\ x' \in D_{\text{pretrain}}\right). \quad (15)$$

This reveals that the completeness of the combined operator requires that at least one operator can correctly distinguish the target under the premise of consistency, which is a disjunction relationship in symbolic logic. In fuzzy logic, the logical disjunction operation can be expressed as a t_{conorm} operation on membership degrees:

$$t_{\text{conorm}}(p_c(a_i), p_c(a_j)) : (f(\alpha_i \circ \alpha_j(x)) = y, (x, y) \in D_{target} \,\middle|\, \text{Consistency}(\alpha_i \circ \alpha_j, x', f),\ x' \in D_{\text{pretrain}}) \quad (16)$$

Finally, a complete rule base can be formed by adding the rule for estimating performance based on satisfiability and completeness:

$$p_s(A) * p_c(A) : f(A(x)) = y, (x, y) \in D_{target} \quad (17)$$

5.2 Deduction

After obtaining the fuzzy rule base, we need to infer an operator combination $A = a_1 \circ a_2 \circ \cdots \circ a_d$ from the rule base such that the membership degree $p_e(A)$ of the fuzzy rule $f(A(x)) = y$ is maximized. To this end, an optimization algorithm must be designed for this membership degree optimization problem. This paper explores several feasible approaches for optimizing the membership degree of the fuzzy rules:

5.2.1 Greedy Inference. The greedy method is the most straightforward approach to this problem. For an algorithm with depth d, we initialize $A_0 = \emptyset$. When d' operators have been selected, we iterate over all candidate operators and choose the α_i that maximizes $p_e(A_{d'} \circ \alpha_i)$, then update $A_{d'+1} = A_{d'} \circ \alpha_i$. Finally, we set $A^* = A_d$.

Although the greedy method is simple and efficient, it cannot foresee the combined effect of subsequent selections and does not support backtracking. This may lead to a local optimum that differs significantly from the global optimum.

5.2.2 Dynamic Programming Inference. Dynamic programming (DP) reformulates the problem as a variant of the knapsack problem. Since computing $p_e(A)$ requires dynamically maintaining $p_s(A)$, and $p_c(A)$, we can use one of them as the DP state and the remaining

one as the optimization objective, thus transforming the problem into a dynamic programming problem. For example, we define the state $p_c[d'][p_s]$: the maximum completeness achievable with d' operators, given satisfiability p_s.

Because p_s, and p_c are continuous values, while DP states require discrete indices, we round the continuous values to single-digit precision according to percentage.

In the state transition step, for each operator α_i, we iterate over the number of selected operators d', and satisfiability p_s, then update the maximum completeness:

$$\begin{aligned} & p_c[d'+1]\left[p_s(A_{d'} \circ \alpha_i)\right] \\ & = \max\left(p_c(A_{d'} \circ \alpha_i),\ p_c[d'+1]\left[p_s(A_{d'} \circ a_i)\right]\right). \end{aligned} \tag{18}$$

After all state transitions, we traverse all states. For each state $p_c[d'][p_s(A')]$, compute the objective value:

$$p_e(A') = p_s(A') \cdot p_c[d'][p_s(A')]. \tag{19}$$

The A' that maximizes $p_e(A')$ is taken as the final solution $A^* = A'$.

5.2.3 Pareto Inference. We maintain a list that stores only *non-dominated* state points, each represented by the pair $(p_s(A'), p_c(A'))$. When adding a new operator α_i, each state in the list generates a new state: $\left(p_s(A' \circ \alpha_i),\ p_c(A' \circ \alpha_i)\right)$. We then check whether the new state is Pareto optimal with respect to the current list. If a state in the list dominates the new state (i.e., has larger p_s, and p_c), the new state is discarded. Duplicates and dominated states are pruned iteratively.

In the end, only a small number of critical states remain. We select the state whose pair product $p_s(A') \cdot p_c(A')$ is maximized as the final solution A^*.

5.2.4 Simulated Annealing Inference. The optimization variable is a 0-1 vector $[b_1, b_2, \ldots, b_{|\mathcal{A}|}]$, where b_i indicates whether the i-th operator is selected. In the initial state, we randomly generate a feasible solution by selecting $d' \le d$ operators and setting $b_i = 1$.

At each neighborhood operation, we randomly flip one binary variable (select $\leftrightarrow$ not select), while ensuring the total number of selected operators $d' \le d$. During annealing, the temperature starts high and gradually decreases until a stopping criterion is met. At each step, the acceptance probability is computed based on the current temperature to decide whether to accept the new solution or keep the current one.

By iterating through neighborhood generation, temperature decay, acceptance probability calculation, and best solution update, we obtain the 0-1 vector b^* that optimizes p_e. We then reconstruct the operator sequence: initialize $A_0 = \emptyset$, and iteratively set

$$A_{i+1} = \begin{cases} A_i \circ a_i & \text{if } b_i^* = 1, \\ A_i & \text{if } b_i^* = 0. \end{cases} \tag{20}$$

Finally, we take $A^* = A_{|\mathcal{A}|}$.

5.2.5 Bisection Inference. We transform the optimization problem into a 0-1 programming problem. The optimization variable is a 0-1 vector $[b_1, b_2, \ldots, b_{|\mathcal{A}|}]$, where b_i indicates whether the i-th operator is selected. The goal is to find b^* that maximizes $p_e(A)$.

Directly optimizing $p_e(A)$ is difficult, so we use monotonicity to combine binary search with linear programming. The constraints are: at most d operators are selected, and each operator can be chosen at most once (0-1 selection).

The core of this method is bisection search for the optimal score and 0-1 integer programming for feasibility verification, a standard industrial-grade approach for continuous-value combinatorial optimization:

- Perform binary search on the optimal score p_e^*.
- For each candidate score p_e' to verify, construct the constraint: does there exist an operator subset B' such that the corresponding A' satisfies $p_e(A') \ge p_e'$?
- Use an integer programming solver to check feasibility.

Upon convergence, we obtain the globally optimal score p_e^* and backtrack to recover the optimal operator combination A^*. This approach involves no discretization and no precision loss.

All the above algorithms can solve the fuzzy logic membership optimization problem, with different trade-offs among efficiency, performance, and precision.

5.3 Induction

we have obtained the optimal feasible algorithm. We only need to pre-train a machine learning model f^* on this algorithm A^* and fine-tune it on the target data to form a dedicated model for the current target. The trained model can be further used for rule extraction in subsequent tasks to form a closed loop, thereby realizing the artificial intelligence meta-cognitive process based on the abduction-deduction-induction ternary model.

$$f^* = \arg\max_{f \in \mathcal{F}} \quad p_e(A^*, f, D_{pretrain}, D_{target}) \tag{21}$$

6 Experiments

In terms of experiments, we constructed an operator library based on self-supervised learning, containing a total of 115 self-supervised operators. On this basis, we first verified the theoretical part proposed in this paper, proving that the target performance estimation method based on our theory is highly positively correlated with the actual performance verified through experiments. Secondly, we verified the feasibility of the algorithm design based on logical reasoning proposed by us, which achieved excellent performance and efficiency on multiple datasets. Finally, we conducted ablation experiments and sensitivity analysis: the former proved that both satisfiability and completeness rules are necessary, while the latter, focusing on the fuzzy logic operators t_{norm} and t_{conorm} corresponding to the conjunction and disjunction logic, proved that conventional fuzzy logic operators are all applicable under our algorithm framework.

6.1 Experimental Settings

For the depth of the algorithm, we set $d = 10$. For the choice of t-norm and t-conorm, we use the commonly used product/algebraic sum pair, i.e., $t_{\text{norm}} = a \cdot b$ and $t_{\text{conorm}} = a + b - a \cdot b$.

We selected 11 commonly used unsupervised pre-training tasks in current self-supervised learning settings, and applied different augmentation strength parameters, ultimately constructing a benchmark containing 115 unsupervised tasks which are shown in 1.

All operations in the table are implemented based on the torchvision.transforms module. Under the same dataset, model, and hyperparameters, we used different tasks to perform both performance estimation and actual training. These unsupervised tasks include ResizedCrop, Rotation, Translate, Shear, Scale, Brightness, Contrast, Saturation, Hue, HorizontalFlip, and VerticalFlip—all of which are widely adopted in self-supervised pretraining [3, 4, 11, 12]. The prior assumption introduced by these tasks assumes that the label remains consistent after data transformations; specifically, the model is expected to produce identical predictions for two independently augmented views of the same input sample.

Table 1: Candidate Operators for Experiments

Name	Strength	Operation
ResizedCrop	$s \in [0, 10] \cap \mathbb{Z}$	ResizedCrop(scale=s/10)
Rotation	$s \in [0, 10] \cap \mathbb{Z}$	Rotation(degrees=s/10*180)
Translate	$s \in [0, 10] \cap \mathbb{Z}$	Affine(translate=s/10)
Shear	$s \in [0, 10] \cap \mathbb{Z}$	Affine(shear=s/10)
Scale	$s \in [1, 10] \cap \mathbb{Z}$	Affine(scale=s/10)
Brightness	$s \in [0, 10] \cap \mathbb{Z}$	ColorJitter(brightness=s/10)
Contrast	$s \in [0, 10] \cap \mathbb{Z}$	ColorJitter(contrast=s/10)
Saturation	$s \in [0, 10] \cap \mathbb{Z}$	ColorJitter(saturation=s/10)
Hue	$s \in [0, 5] \cap \mathbb{Z}$	ColorJitter(hue=s/10)
HorizonFlip	$s \in [0, 10] \cap \mathbb{Z}$	HorizontalFlip(p=s/10)
VerticalFlip	$s \in [0, 10] \cap \mathbb{Z}$	VerticalFlip(p=s/10)

Under the basic setup, we conducted experiments on the CIFAR-10, CIFAR-100 [16] and ImageNet-200 [21] datasets. For each class, we selected only 5 labeled examples—i.e., only 50 labeled samples in total for CIFAR-10, 500 for CIFAR-100 and 1000 for ImageNet-200. For performance estimation based on proposed indicators, we used 50 unlabeled examples per class—i.e., 500 unlabeled samples for CIFAR-10, 5,000 for CIFAR-100 and 10000 for ImageNet-200. After obtaining the optimal algorithm A^*, we perform pre-training on all unsupervised data. All the training samples excepted labeled ones were used as unlabeled data—i.e., 49,950 samples for CIFAR-10 and 49,500 for CIFAR-100 and 99,000 for ImageNet-200. All evaluations of actual performance were conducted on the full test sets.

In all experiments, we used ViT-B [8] with a patch size of 16 as the baseline model. The batch size was set to 64 and the estimation epoch is set to 5 while the training epoch is set to 500, the optimizer was Adam [15], the learning rate was fixed at 5e-5, and the weight decay was set to 0.01. The loss function for unsupervised pretext tasks was uniformly set to mean squared error (MSE) loss, and the loss function for the supervised target task was set to cross-entropy loss. All experiments were conducted using the PyTorch framework on 8 A800 GPUs.

6.2 Estimation

To verify that our proposed theory, which estimates the target performance based on algorithmic satisfiability and completeness, serves as a good alternative to the performance obtained by full training and validation, we validate our proposed estimation strategy on CIFAR10, CIFAR100, and TinyImageNet.

To intuitively illustrate the correlation between the performance p_e estimated using only a small amount of data and actual target performance p_t trained and validated on the full dataset, we plotted scatter diagrams. We represent operation using color and strength using opacity. All values on the axes are expressed as percentages. The comparison between estimated performance and target performance are shown in figs. 1 to 3. In terms of the final results, the Pearson correlation coefficients between estimated and target performance for self-supervised learning reaches 0.820, 0.949, 0.678 on CIFAR-10, CIFAR-100, ImageNet-200 respectively.

The experimental results sufficiently demonstrate that the target performance based on satisfiability and completeness is highly reliable. Through hypothesis testing, the proposed estimation method can accurately predict the real performance that relies on large-scale training and validation at an extremely low cost, above the 99% confidence level.

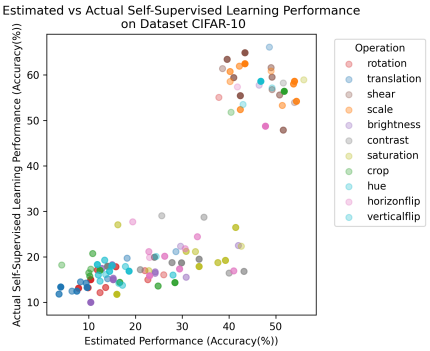

Figure 1: This figure illustrates the comparison between estimated performance and target performance after full self-supervised learning on CIFAR-10 with the basic setup.

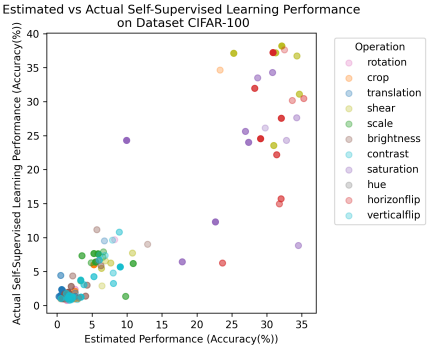

Figure 2: This figure illustrates the comparison between estimated performance and target performance after full self-supervised learning on CIFAR-100 with the basic setup.

6.3 Experiments results

We compare the target performance corresponding to different inference algorithms. The results demonstrate that our proposed method is effective.

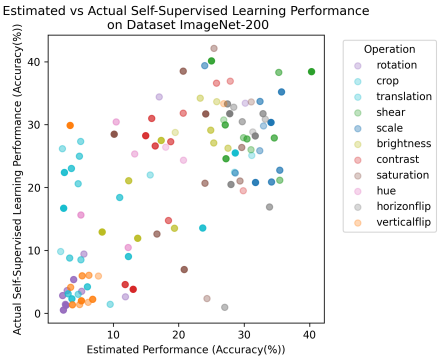

Figure 3: This figure illustrates the comparison between estimated performance and target performance after full self-supervised learning on ImageNet-200 with the basic setup.

We use A_{rand} to denote RandAugment as the baseline, and A^*_{auto} to denote AutoAugment, an automated machine learning algorithm. We denote RandAugment as the baseline by A_{rand} [7], and the Auto-ML algorithm AutoAugment by A^*_{auto}[6]. We use A^*_{greedy}, A^*_{dp}, A^*_{pareto}, $A^*_{\text{annealing}}$, and $A^*_{\text{bisection}}$ to denote the optimal algorithm structures obtained by the five inference algorithms, respectively. The comparison results are shown in tables 2 and 3:

Table 2: The results of algorithm performance.

Method	CIFAR10	CIFAR100	ImageNet200
A_{Rand}	86.7	81.2	73.6
A^*_{Auto}	95.8	89.2	77.3
A^*_{greedy}	95.2	87.1	75.2
A^*_{dp}	95.7	88.7	77.8
A^*_{pareto}	95.4	88.7	76.5
$A^*_{\text{annealing}}$	96.1	89.5	78.2
$A^*_{\text{bisection}}$	96.2	88.6	77.6

Table 3: The results of time consumption (GPU hours).

Method	CIFAR10	CIFAR100	ImageNet200
A^*_{Auto}	1012	1152	1432
A^*_{greedy}	48	51	57
A^*_{dp}	65	68	77
A^*_{pareto}	63	67	82
$A^*_{\text{annealing}}$	63	65	76
$A^*_{\text{bisection}}$	79	81	92

6.4 Ablation Study

To verify the necessity of the two metrics, satisfiability and completeness, for optimizing the operator combination, we conducted ablation experiments. The control groups include $p_e = p_c$ (without using the satisfiability metric), $p_e = p_s$ (without using the completeness metric), and $p_e = p_s \cdot p_c$ (using both metrics). We adopt Bisection as the basic inference method, and the experimental results on the CIFAR-10 dataset are shown in table 4.

Table 4: The results of ablation study

Satisfiability	Completeness	Performance
✓	×	92.1
×	✓	89.2
✓	✓	96.2

6.5 Sensitivity Analysis

To verify the generality of the proposed scheme across different fuzzy logic operators t-norm and t-conorm, we conducted experiments under various operator combinations. The results demonstrate that our scheme is effective under different common settings of fuzzy logic operators.

The widely used t-norm and t-conorm operators we compare also include: the Gödel norm, where $t_{\text{norm}}(a,b) = \min(a,b)$ and $t_{\text{conorm}}(a,b) = \max(a,b)$; and the Łukasiewicz bounded norm, where $t_{\text{norm}}(a,b) = \max(a+b-1,0)$ and $t_{\text{conorm}}(a,b) = \min(a+b,1)$.

We adopt Bisection as the basic inference method, and the experimental results on the CIFAR-10 dataset are shown in table 5.

Table 5: The results of sensitivity analysis.

t_{norm}	t_{conorm}	Performance
$\min(a,b)$	$\max(a,b)$	95.2
$\max(a+b-1,0)$	$\min(a+b,1)$	94.9
$a \cdot b$	$a+b-a \cdot b$	96.2

7 Conclusion

Different from previous high-complexity black-box methods based on complex search and second-order optimization, this paper attempts to endow artificial intelligence with the ability to adaptively design algorithms for solving target problems in a logic-reasoning-based manner. The proposed framework features a transparent process and strong interpretability, achieves significantly improved efficiency, and greatly reduces dependence on data and annotation resources, thus establishing a feasible foundation for further enhancing the meta-capabilities of artificial intelligence. Rooted in cutting-edge findings in cognitive science, supported by solid machine learning theories, and empowered by rigorous fuzzy logic tools, the method in this paper realizes a triple closed loop: extracting rules via abduction, inferring algorithms via deduction, and training models via induction. For the proposed framework, we provide diverse feasible solutions. Experiments verify that each of these solutions has its own merits, and further validate the performance variations under different settings.

In future work, we will further extend the generality of the framework to accommodate a wider range of operator libraries, such as more fine-grained neural network operators, and apply this framework to more diverse data modalities and task scenarios.

References

[1] Bikram Pratim Bhuyan, Amar Ramdane-Cherif, Ravi Tomar, and TP Singh. 2024. Neuro-symbolic artificial intelligence: a survey. *Neural Computing and Applications* 36, 21 (2024), 12809–12844.

[2] Le-Wen Cai, Wang-Zhou Dai, Yu-Xuan Huang, Yufeng Li, Stephen H Muggleton, and Yuan Jiang. 2021. Abductive Learning with Ground Knowledge Base.. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*. 1815–1821.

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9650–9660.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*. 1597–1607.

[5] Cristina Cornelio, Jan Stuehmer, Shell Xu Hu, and Timothy Hospedales. 2023. Learning where and when to reason in neuro-symbolic inference. In *Proceedings of the 11th International Conference on Learning Representations*.

[6] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. 2019. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 113–123.

[7] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. 2020. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 702–703.

[8] Alexey Dosovitskiy. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

[9] Richard Evans and Edward Grefenstette. 2018. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research* 61 (2018), 1–64.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*. 1126–1135.

[11] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. 2020. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems* 33 (2020), 21271–21284.

[12] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9729–9738.

[13] Pascal Hitzler and Md Kamruzzaman Sarker. 2022. *Neuro-symbolic artificial intelligence: The state of the art*.

[14] Yu-Xuan Huang, Wang-Zhou Dai, Le-Wen Cai, Stephen H Muggleton, and Yuan Jiang. 2021. Fast abductive learning by similarity-based consistency optimization. In *Advances in Neural Information Processing Systems*. 26574–26584.

[15] Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[17] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. 2019. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *Proceedings of the 7th International Conference on Learning Representations*.

[18] Terufumi Morishita, Gaku Morio, Atsuki Yamaguchi, and Yasuhiro Sogawa. 2023. Learning Deductive Reasoning from Synthetic Corpus based on Formal Logic. In *Proceedings of the 40th International Conference on Machine Learning*. 25254–25274.

[19] Charles Sanders Peirce. 1991. *Peirce on signs: Writings on semiotic*. UNC Press Books.

[20] Nikunj Saunshi, Orestis Plevrakis, Sanjeev Arora, Mikhail Khodak, and Hrishikesh Khandeparkar. 2019. A theoretical analysis of contrastive unsupervised representation learning. In *Proceedings of the 36th International Conference on Machine Learning*. 5628–5637.

[21] Xuan Yang. 2015. Tiny imagenet visual recognition challenge. (2015).

[22] Zhun Yang, Adam Ishay, and Joohyung Lee. 2023. Neurasp: Embracing neural networks into answer set programming. *Proceedings of the 29th International Joint Conference on Artificial Intelligence* (2023), 1755–1762.

[23] Lotfi A Zadeh. 2008. Is there a need for fuzzy logic? *Information sciences* 178, 13 (2008), 2751–2779.

[24] Zhi-Hua Zhou. 2019. Abductive learning: towards bridging machine learning and logical reasoning. In *Science China Information Sciences*, Vol. 62. 1–3.