Proving Olympiad Algebraic Inequalities without Human Demonstrations

Chenrui Wei[1]
chenruiw97@gmail.com

Mengzhou Sun[2]
sunm07@u.nus.edu

Wei Wang[1,*]
wangwei@bigai.ai

[1]State Key Laboratory of General Artificial Intelligence, BIGAI, Beijing, China
[2]Department of Mathematics, National University of Singapore

Abstract

Solving Olympiad-level mathematical problems represents a significant advancement in machine intelligence and automated reasoning. Current machine learning methods, however, struggle to solve Olympiad-level problems beyond Euclidean plane geometry due to a lack of large-scale, high-quality datasets. The challenge is even greater in algebraic systems, which involve infinite reasoning spaces within finite conditions. To address these issues, we propose *AIPS*, an *Algebraic Inequality Proving System* capable of autonomously generating complex inequality theorems and effectively solving Olympiad-level inequality problems without requiring human demonstrations. During proof search in a mixed reasoning manner, a value curriculum learning strategy on generated datasets is implemented to improve proving performance, demonstrating strong mathematical intuitions. On a test set of 20 International Mathematical Olympiad-level inequality problems, AIPS successfully solved 10, outperforming state-of-the-art methods. Furthermore, AIPS automatically generated a vast array of non-trivial theorems without human intervention, some of which have been evaluated by professional contestants and deemed to reach the level of the International Mathematical Olympiad. Notably, one theorem was selected as a competition problem in a major city's 2024 Mathematical Olympiad. All the materials are available at *sites.google.com/view/aips2*.

1 Introduction

One of the key milestones in the field of artificial intelligence is the capability to reason (Pearl 1998) and prove theorems (Wu 1978; Chou et al. 2000; Trinh et al. 2024). However, theorem proving often involves long reasoning chains, complex mathematical structures, intricate calculations, and infinite reasoning spaces. Consequently, developing AI capable of proving complex mathematical theorems requires sophisticated reasoning and the ability to navigate through an extensive search space to construct a valid proof. The complexity of these problems lies in the need for effective heuristics and strategies to manage the vast number of possible actions and the lengthy sequences of logical steps necessary to arrive at a solution.

Existing work on grade school and college admission math problems has achieved notable success, e.g., GSM8K (Cobbe et al. 2021) and SAT Math (Achiam et al. 2023), which demonstrate better performance on tasks such as arithmetic and basic algebra. However, research focused on solving International Mathematical Olympiad (IMO)-level problems remains relatively sparse. Notable efforts in this area include AlphaGeometry (Trinh et al. 2024), and GPT-*f* (Polu and Sutskever 2020) on miniF2F (Zheng et al. 2021), which have made progress in solving Euclidean plane geometry at the Olympiad level and various mathematical competition problems, respectively.

*Corresponding author.

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

A significant challenge for learning-based methods in this domain is the scarcity of suitable datasets, which limits the ability to train models effectively and hampers progress in achieving human-level performance on these high-difficulty problems. The miniF2F dataset (Zheng et al. 2021) includes only 244 validation and 244 test mathematical problems from various competitions. AlphaGeometry (Trinh et al. 2024) addresses this issue by synthesizing millions of theorems and proofs across different levels of complexity to train a neural language model from scratch. Similarly, the INequality Theorem proving benchmark, INT (Wu et al. 2020), can synthesize a theoretically unlimited number of theorems and proofs in the domain of algebraic equalities and inequalities. However, INT focuses on testing a learning-assisted theorem proving agent's generalization ability rather than increasing the difficulty to competition level.

Another significant challenge in automated theorem proving is designing effective search strategies to navigate the vast space of possible proofs. Recent advancements have highlighted various approaches to enhance search efficiency and proof success rates. Some studies have shown that incorporating Monte Carlo Tree Search (MCTS) at test time can significantly aid in proving new theorems (Wu et al. 2020). Inspired by the success of AlphaZero (Zhang and Yu 2020), other research has explored HyperTree Proof Search (HTPS) (Lample et al. 2022), which learns from previous proof searches through online training, iteratively improving its strategy by learning which paths are more likely to lead to successful proofs. Another innovative approach starts the proof search from the root goal that needs to be proved (Polu and Sutskever 2020), expanding a maintained proof tree by prioritizing open goals based on their cumulative log probability.

In this work, we introduce *AIPS*, an *Algebraic Inequality Proving System*, which can generate a large number of high-quality theorems and solve IMO-level algebraic problems. AIPS focuses on ternary and quaternary inequalities, excluding n-variable inequalities represented recursively in formal verification systems. Among the generated theorems, some have proven to be very challenging, with one selected for a major city's 2024 Mathematical Olympiad. We present novel and challenging inequality theorems discovered by AIPS in the appendix, which have been carefully evaluated by IMO-level professional contestants and found to be comparable to IMO inequalities from around the year 2000.

Additionally, AIPS incorporates a value network to evaluate newly generated inequalities, selecting subgoal candidates based on the top scores provided by the value network. The value network is trained on synthetic datasets with increasing difficulty in a curriculum manner. In our experiments, AIPS proved difficult theorems up to the IMO level and solve 10 out of 20 problems in an IMO-level inequality test, significantly surpassing the performance of previous Large Language Model-based theorem provers (Polu and Sutskever 2020; Polu et al. 2022; Yang et al. 2024; Song et al. 2024).

The main contributions in this paper are summarized as follows:

1. We propose a symbolic deductive engine capable of efficiently generating high-quality and solving high-difficulty algebraic inequality theorems. This engine addresses the bottleneck of lacking large-scale, high-quality data in this field.
2. We demonstrate that a symbolic algebraic inequality prover can be significantly enhanced under the guidance of a value network, especially when the value network is trained in a curriculum manner.
3. Our AIPS can generate challenging and elegant inequality theorems, including one selected for a major city's Mathematical Olympiad. AIPS proves 10 out of 20 IMO-level inequalities, surpassing state-of-the-art methods and producing highly human-readable proofs.

2 Related Work

Automated Theorem Proving. Automated theorem proving has been a focus of artificial intelligence since the 1950s (Harrison et al. 2014; Wu 1978). Modern theorem provers, based on tactic and premise selection, search for proofs by interacting with proof assistants such as Lean (De Moura et al. 2015), Coq (Barras et al. 1999) and Isabelle (Nipkow et al. 2002). They struggle with the rapidly expanding search space and the scarcity of high-quality datasets in most mathematical domains. The challenge is even greater for proving algebraic inequalities, which involve complex computational rules. Previous efforts to address this issue have focused on augmenting tactic selection and premise prediction in interactive theorem provers (Polu and Sutskever 2020; Polu et al. 2022; Yang et al.

2024). However, these provers have only been able to solve problems of limited difficulty in this field. In this paper, our AIPS can solve highly complex algebraic inequality theorems up to the IMO level.

Datasets and Benchmarks for Theorem Proving. Formal mathematical libraries, such as Isarstep (Li et al. 2020), Mathlib (van Doorn et al. 2020), and CoqGym (Yang and Deng 2019), currently serve as the primary datasets for theorem proving. These libraries, manually curated by humans, include many intricate and profound proofs, such as the formal proofs of the Four-Color Theorem (Gonthier et al. 2008), the Liquid Tensor Experiment (Scholze 2022), and Fermat's Last Theorem (Buzzard and Taylor 2024). Due to the labor-intensive nature of manual proof writing, these libraries are relatively small, typically containing around 200,000 theorems. While they encompass a wide range of mathematical fields, the number of theorems in specific areas is quite limited.

Synthetic theorems can provide large-scale datasets for learning-based theorem provers (Polu and Sutskever 2020; Wu et al. 2020). However, these theorems are often of limited difficulty. Recently, significant progress has been made in synthesizing geometry theorems (Trinh et al. 2024) using neural theorem provers. In this paper, we develop AIPS for algebraic inequalities, which can automatically and efficiently generate a large number of intricate theorems, with some reaching the IMO level. These theorems will significantly improve neural theorem proving methods.

Search Strategy for Efficient Inference. Deep learning has achieved remarkable success in enhancing search algorithms (Silver et al. 2016, 2017). Proof search in theorem proving, however, is more challenging compared to self-play games like Go, as it may involve an infinite search space within finite conditions. INT (Wu et al. 2020) incorporates MCTS, while HyperTree Proof Search (HTPS) (Lample et al. 2022) employs online training to improve search strategy. GPT-*f* (Polu and Sutskever 2020) learns a value network to guide backward search. Our AIPS integrates the benefits of both HTPS and GPT-*f*, introducing a value curriculum learning strategy.

3 Algebraic Inequality Proving System

3.1 Symbolic Deductive Engine for Algebra

Interactive theorem provers, such as Lean, can verify mathematical operations but lack the ability to perform automated mathematical reasoning by combining computational rules. This challenge is amplified in the automatic proof of algebraic inequalities, which often involves numerous calculations, extensive transformation rules, and complex theorem matching. To address this, we design a symbolic deductive engine for algebra, encompassing dozens of fundamental theorems and transformation rules for algebraic inequalities. It integrates with the symbolic computation system SymPy [2], enabling effective algebraic reasoning.

3.1.1 Representation for Algebraic Expressions and Theorems

Algebraic expressions are represented symbolically with an underlying expression tree structure as shown in Fig. 1. The basic computational rules include self-equivalence transformations of inequalities and various built-in SymPy functions, such as combining fractions (`sympy.together`) and expanding expressions (`sympy.expand`). Our deductive engine's library also includes fundamental algebraic inequality theorems: the Arithmetic Mean-Geometric Mean Inequality (AM-GM), the weighted AM-GM Inequality, Cauchy's Inequality, Jensen's Inequality, the discrete Hölder's Inequality, Schur's Inequality, the binary and ternary Muirhead's Theorem. Each inequality is represented as a category of theorem matching, containing variables, conditions, conclusions, and equality conditions.

3.1.2 Pattern Matching for Inequality Theorems

During symbolic reasoning, the system attempts to apply inequality theorems to a particular algebraic expression or inequality, as shown in Fig. 1. When matching algebraic expressions with inequality theorems, it first traverses the expression tree to determine how the value of the entire expression changes as the node's value increases, updating the node's label accordingly. If the change cannot be determined, no theorem matching is performed on the subtree of that node. After completing the labeling, the system matches the next layer of determinable nodes with theorems. If a match

[2]https://www.sympy.org/

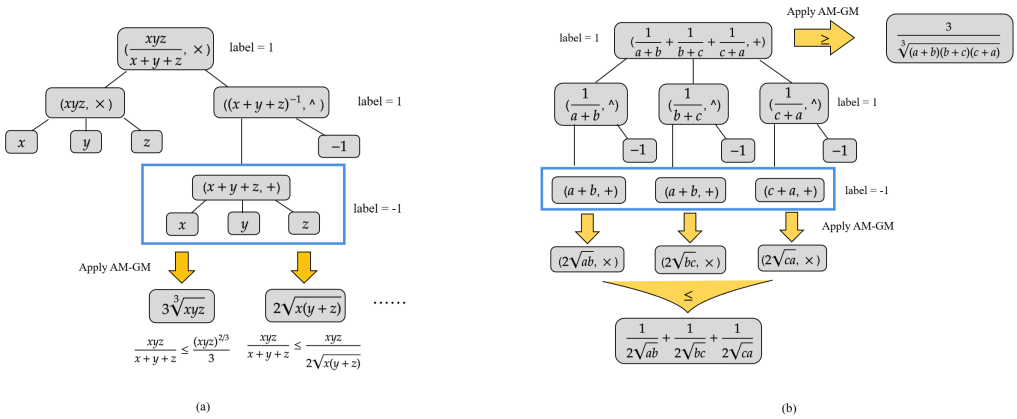


Figure 1: Examples of expression trees and pattern matching for the AM-GM inequality are illustrated. In (a), for $x, y, z \geq 0$, the value of $\frac{xyz}{x+y+z}$ decreases as $x+y+z$ increases, so the label of the node $x+y+z$ is -1. By applying the AM-GM inequality, we derive a series of upper bounds with respect to the root, e.g., $\frac{(xyz)^{2/3}}{3}$ and $\frac{xyz}{2\sqrt{x(y+z)}}$. In (b), when traversing the expression tree of $\frac{1}{a+b}+\frac{1}{b+c}+\frac{1}{c+a}$, pattern matching for the AM-GM inequality at various nodes yields different types of bounds, such as the upper bound $\frac{1}{2\sqrt{ab}}+\frac{1}{2\sqrt{bc}}+\frac{1}{2\sqrt{ca}}$ and the lower bound $\frac{3}{((a+b)(b+c)(c+a))^{\frac{1}{3}}}$.

is successful, the matched sub-expression is replaced with the new expression obtained using the theorem. Based on the previous labels, it then determines whether the entire expression increases or decreases, thereby deriving a new inequality. For certain inequality theorems, such as Jensen's Inequality, pattern matching is particularly complex and time-consuming. Therefore, to improve the efficiency of reasoning at each step, we have imposed time limits on the matching process for some theorems.

3.1.3 Forward Reasoning

Forward reasoning in theorem proving involves matching variables and conditions to a theorem and deducing new conclusions. In our engine, new inequalities can be obtained by matching theorems to both sides of an inequality or by applying self-equivalence transformation rules. If any two of the resulting inequalities can be connected (e.g., applying $a \leq b$ and $b \leq c$ to derive $a \leq c$), the system continues to link them to form new inequalities. Therefore, our engine has the capability to perform forward reasoning to generate large-scale data.

3.2 Olympiad-Level Inequality Proof Set

One of the main challenges in enabling learning-based models to solve complex mathematical problems is the scarcity of large-scale, high-quality datasets. To overcome this obstacle, we develop a theorem generator that effectively generates Olympiad-level inequality theorems by enhancing the methods described in Section 3.1.3.

3.2.1 Synthetic Theorem Generation

We randomly generate thousands of cyclically symmetric symbolic expressions, which serve as the initial premises for our reasoning process. Utilizing 32 CPUs, we run Algorithm 1 for 8 hours, resulting in the generation of 191,643 inequality theorems. The generated inequalities are stored in a tree structure, with each node containing the necessary information for extracting proofs and training machine learning models. Fig. 2 shows the procedure of generating a synthetic theorem in our AIPS, and Fig. 3(a) shows the distribution of inference depths in the generated inequalities.

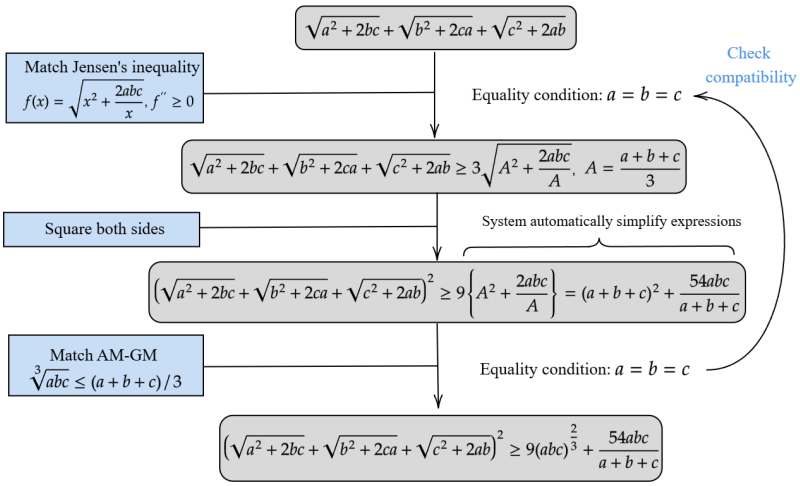


Figure 2: An example of generating synthetic theorems in AIPS. When the initial premise $\sqrt{a^2+2bc}+\sqrt{b^2+2ca}+\sqrt{c^2+2ab}$ successfully matches with Jensen's inequality, a new inequality is generated. By subsequently applying transformation rules and matching other fundamental inequalities, such as the AM-GM inequality, the deductive engine incrementally generates new inequality theorems. When an inequality theorem is applied, the system verifies whether the equality condition holds, e.g., $a=b=c$.

3.2.2 Synthetic Theorem Evaluation

To evaluate the quality of our dataset, we select 10 problems with reasoning lengths exceeding five steps, and invite two National Mathematical Olympiad gold medalists and one silver medalist to assess the difficulty and elegance of these problems. Their evaluations reveal that our dataset contains a vast array of non-trivial theorems, some of which surpass the difficulty of inequalities found in early IMO competitions. Notably, one inequality theorem from our dataset is selected for a major city's Mathematical Olympiad. All the 10 problems and evaluation results are provided in the Appendix.

3.3 Neural Algebraic Inequality Prover

By leveraging the capabilities of the deductive engine introduced in Section 3.1 and the Best-First-search algorithm (Dechter and Pearl 1985), we develop an algebraic inequality prover. This prover formulates the algebraic inequality proving as a sequential decision-making process by selecting theorems to generate highly human-readable proofs. As shown in Fig. 4, given a goal and related conditions, AIPS first generates a list of subgoals by applying a set of theorems at each iteration. A value neural network is then used to evaluate these newly generated subgoals along with the previous unresolved subgoals. The top-value subgoal is selected for the next step of reasoning. This iterative process continues until the proof is successfully completed, as shown in Fig. 3(b).

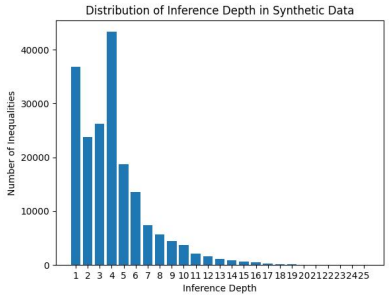
3.3.1 Searching Proofs by Combining Value Network with Symbolic Prover

The procedure of searching for inequality proofs is generally divided into three parts: mixed reasoning for subgoal generation, evaluation, and planning.

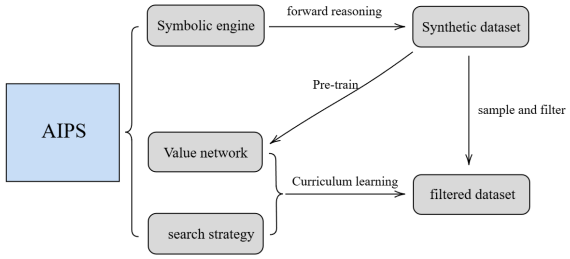
Subgoal Generation. There are two methods for generating subgoals in AIPS. The first method involves applying fundamental inequality theorems. Let X be the set of variables. Suppose the inequality theorem to prove is $u(X)\leq v(X)$ under a condition set $\mathcal{P}$. AIPS first homogenizes the inequality to $f(X)\leq g(X)$ on both sides by applying conditions in $\mathcal{P}$. Then, by applying theorems

Algorithm 1 Generating Theorems

function `Generate_Theorems`(*expression* P, *loops* N)
  Initialize *Theorem Set* S,
  Inequality Transformation Rules O, *Inequality Sets* $A1$, $A2$, $A3$
  Apply S to P to obtain a series of inequalities and add those whose equality conditions hold to a set R
  for $i \leftarrow 1$ to N **do**
    for each inequality $ineq$ in R **do**
      Apply rules O to $ineq$ to obtain $A1$
    end for
    for each inequality $ineq$ in R **do**
      Apply theorems S to one side of $ineq$ and check if it can be linked to the original inequality. If so, add it to $A2$
    end for
    for each inequality $ineq$ in $A2$ **do**
      Check if $ineq$ meets the equality condition and add it to $A3$ if it does
    end for
    Update R by selecting M inequalities from the union of $A3$ and $A1$ according to the length of inequalities
  end for
  return R
end function



(a) Distribution of inference depths. In the process of generating synthetic theorems, we limit the reasoning steps. Unlike geometry problem, long reasoning chains in inequality generation can lead to trivial theorems. Solutions to challenging IMO inequalities typically involve only two or three steps of matching inequality theorems.



(b) Self-evolving process of AIPS. After pre-training on the initial synthetic dataset, AIPS is capable of proving some challenging theorems. Guided by the value network, it then attempts to solve problems in an increasingly difficult filtered dataset. By extracting nodes on the proof path as positive labels and other nodes as negative labels, it fine-tunes the value network and gradually improves proving performance in a curriculum manner.

Figure 3: (a) Distribution of inference depths in our dataset. (b) Self-evolving process of AIPS.

to the left-hand side of the target inequality, AIPS generates a series of new inequalities:

$$f(X) \leq h_1(X), \ldots, f(X) \leq h_n(X)$$

This results in subgoals $h_i(X) \leq g(X)$. Similarly, by applying theorems to the right-hand side, AIPS also generates subgoals $f(X) \leq s_j(X)$. The second method involves applying transformation rules such as `sympy.expand` and `sympy.apart` to the goal, generating subgoals that are equivalent to the original inequality.

Evaluation. AIPS employs a value function V_θ to assess the difficulty of each inequality. Formally, we have a function f parameterized by η that encodes the inequality expression s. The encoded embedding vector $f_\eta(s)$ is then fed into a deep neural network g_ϕ, which outputs a value in the interval [0,1]. We choose f to be a transformer encoder with average pooling (Vaswani et al. 2017).

Planning. With the evaluation function V_θ, we use the Best-First search algorithm for planning. We also test the performance of MCTS algorithm, where the result is less satisfactory.

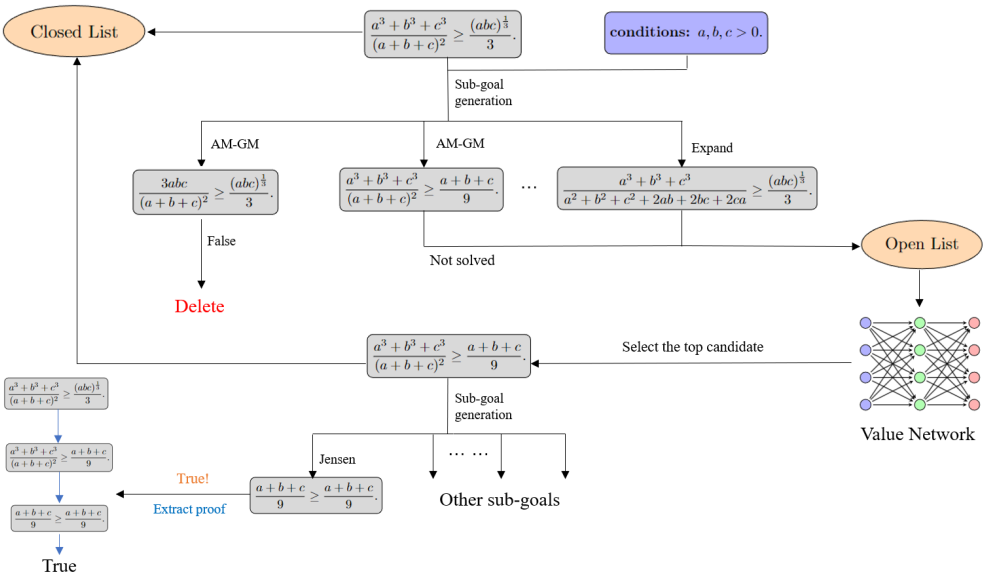


Figure 4: Overview of how AIPS proves a simple theorem. At each step, the deductive engine attempts to match inequality theorems with each side of the goal and applies all transformation rules to the expression, resulting in a list of new subgoals. The searched goal is placed into a closed list, ensuring that it will not be examined again. If one of the new subgoals is true, indicating that the inequality holds, then the theorem is proved. Otherwise, the new subgoals are added to the open list, along with other subgoals generated previously. A value network then evaluates all subgoals in the open list, and the top-value one is chosen for the next iteration of proof search.

There are two primary reasons for this. First, the action space for each state is extremely large, leading to explosive growth of the MCTS searching tree. Second, the high cost of reasoning steps makes the simulation step in MCTS nearly impractical, often exceeding time limits.

We also note that our prover can be combined with any heuristic function, and thus design various baselines in our experiments.

3.3.2 Pre-training Value Network Using a Heuristic Function

We define the tree-depth $\mathcal{D}$ of an inequality as the maximum depth of the expression trees on both sides. Proving an algebraic inequality is equivalent to reducing the tree-depth of the inequality to one. We use $\mathcal{D}$ as the supervision information to train initial heuristic function f_{init} in the Best-First search algorithm. That is to say, we pre-train a value network V_θ as f_{init} on the synthetic dataset by utilizing the tree-depth $\mathcal{D}$.

3.3.3 Fine-tuning Value Network on Filtered Synthetic Data

We create a new dataset by removing all inequalities with inference depth less than 4. We then randomly sample 1,200 problems and sort them by tree-depth in ascending order. For inequalities with the same tree-depth, they are sorted by the length of their string representation, with shorter lengths placed first.

The fine-tuning procedure involves sequentially proving these inequalities and updating the parameters of the value network. If an inequality is successfully proved, we record the set of subgoals on the proof path as T and the set of subgoals that are searched but not on the proof path as F. The values of the elements in T are scaled down by a factor of ϵ, while the values of the elements in F are increased. Using these labels, we perform a training round on the value network V_θ, and then proceed to the next problem. This iterative process is used to adjust the network parameters. See Section 2 in Appendix for more details.

4 Experiments

We evaluate AIPS on an Olympiad-level algebraic inequality problem test set. It outperforms the state-of-the-art methods in terms of the number of solved problems, demonstrating the strong algebraic intuitions developed by the learned value network.

4.1 An Olympiad-Level Inequality Benchmark

Current benchmarks for Olympiad-level math problems, such as miniF2F (Zheng et al. 2021) and Fimo (Liu et al. 2023), cover a wide array of topics but often lack a dedicated section for algebraic inequalities. In inequality benchmarks like INT (Wu et al. 2020), the problems are typically of limited difficulty. To address this gap, we collect all ternary and quaternary algebraic inequality problems from IMO since 1990. Additionally, we include challenging problems from IMO shortlists and various national mathematical Olympiads, such as the USAMO, the USA National Team Selection Tests, and the Polish, Japanese, and Korean Mathematical Olympiads, all of which are of comparable difficulty to the IMO. In total, we compile 20 problems for our test set, naming it MO-INT-20 (Math-Olympiad-INequality-Test-20). All problems are checked to ensure they are not in AIPS's training datasets. We also translate the test problems into Lean for subsequent experiments.

4.2 Comparison Methods

Current theorem provers include interactive theorem provers, large language models capable of generating natural language proofs, and neural symbolic theorem provers. We compare LeanCopilot (Song et al. 2024), the open-source state-of-the-art interactive theorem prover in Lean. Additionally, we evaluate general large language models like GPT-4, GPT-4 Turbo and Gemini 1.5 Pro, as well as the math-specific language model Llemma-7b (Azerbayev et al. 2023). For neural symbolic theorem provers, we design various baselines, including our deductive engine paired with breadth-first search and MCTS, our deductive engine equipped with tree-depth in Section 3.3.2 or LLM heuristics as the value function, and our AIPS with only pretrained value network.

It should be noted that we cannot compare with several existing interactive theorem provers (Polu and Sutskever 2020; Polu et al. 2022) since these provers are not open source to be reproduced. However, it is reported that these provers can only prove a few early Olympiad inequalities, as detailed in the appendix of their respective papers.

4.3 Comparison Results and Analysis

We test 11 different provers on the inequalities in MO-INT-20, with each problem limited to 90 minutes of solving time, consistent with the standard problem-solving time in the IMO. All neural-symbolic provers are tested on a single CPU core (equivalent to 1.5 CPU hours per problem). The comparison results are shown in Table 1. It can be seen that our AIPS achieves the best performance and solves 10 out of 20 problems.

Table 1: Model Performances on the MO-INT-20. **DE denotes our deductive engine**. BFS and MCTS are Breadth-First Search and Monte Carlo Tree Search, respectively.

Model Category	**Model**	**Problems Solved (20)**
Large Language Models	Gemini 1.5 Pro	1
	GPT-4	0
	GPT-4 Turbo	0
	Llemma-7b	0
Interactive Theorem Provers	LeanCopilot (LeanDojo)	0
Neural-Symbolic Provers	DE + GPT-4 Turbo's heuristics	6
	DE + BFS	4
	DE + MCTS	5
	DE + tree-depth heuristic function	7
	AIPS with pretrained value network	7
	AIPS	10

Analysis of Large Language Models' Performance. Large language models like GPT-4 have demonstrated remarkable reasoning abilities (Lewkowycz et al. 2022; Wei et al. 2022). However, in this test, only one of the four models, Gemini 1.5 Pro, successfully generates a fully correct natural language proof. When solving problems, large language models tend to either make trivial mistakes or indicate that they do not know how to solve them, despite the potential contamination of their training data by online proofs. These results reveal their limited math reasoning ability.

Analysis on a Formal Theorem Prover's Performance. Recent studies reveal the capabilities of neural theorem provers based on Interactive Theorem Prover (ITP) frameworks (Yang et al. 2024; Rute et al. 2024). These systems generally convert theorem proving into code completion tasks. We evaluate the performance of one such theorem prover, LeanCopilot (Song et al. 2024), developed from LeanDojo, on our test set. LeanCopilot is the current open-source state-of-the-art theorem prover based on Lean. The results indicate its limited ability to solve complex algebraic problems: None of the problems are solved through proof search in LeanCopilot. Additional tests on tactic suggestions (see Section 2 in Appendix) show that current formal theorem provers struggle to predict the complex premises required for proving inequalities.

Analysis on Neural Symbolic Provers' Performance. In this test, neural symbolic provers demonstrate a strong ability to prove algebraic inequalities using best-first search algorithm. By applying either breadth-first search or MCTS algorithm, our deductive engine successfully solves four and five problems, respectively. We also test performance under the guidance of a tree-depth heuristic function and a pre-trained value network using the best-first search algorithm, both of which solve seven problems. Additionally, we prompt GPT-4 Turbo and find it exhibit some algebraic intuition, successfully guiding the deductive engine to solve six problems—two more than the breadth-first search. However, it is worth noting that large language models (LLMs) may occasionally prioritize lengthy and meaningless subgoals. Due to the exponential growth of the number of new inequalities as the width and height of the expression trees increase, it can result in expression strings longer than the LLMs' input context length. For example in problem 4 from the 2014 Japan Mathematical Olympiad, it chooses a very long subgoal at iteration 2, resulting in subgoals at the next iteration being three times longer than its input context length.

Finally, following a curriculum learning strategy on 1,000 inequality problems, AIPS achieves the best performance, solving 10 out of 20 problems. Among the 10 problems from the IMO or IMO shortlist, it successfully solves five, reaching the average level of IMO contestants. We also test the performances of AIPS after 200, 400, 600, and 800 loops of fine-tuning value network (see Section 2 in Appendix). The results demonstrate that our value curriculum learning strategy is very effective, with the number of proof search steps significantly decreasing during the training process, and the number of solved problems increasing to 10 ultimately.

5 Conclusion

In conclusion, solving Olympiad-level mathematical problems is a significant milestone in machine intelligence and automated reasoning. The lack of large-scale, high-quality datasets presents a challenge, particularly in algebraic systems. To address this, we propose *AIPS*, an *Algebraic Inequality Proving System*, which autonomously generates complex inequality theorems and effectively solves Olympiad-level inequality problems without human input. Utilizing a value curriculum learning strategy, AIPS demonstrated strong mathematical intuition by solving 10 out of 20 International Mathematical Olympiad-level problems. One of these theorems was selected for a major city's 2024 Mathematical Olympiad.

In the future, by incorporating more fundamental theorems and operational rules, our AIPS could solve even more complex problems, discover a greater number of non-trivial theorems, and assist mathematicians in solving modern mathematical challenges. However, it currently lacks the ability to autonomously propose and comprehend new definitions. Instead, it relies on handwritten theorems and matching rules, which is time-consuming. Addressing this limitation is a crucial area for future research.

6 Acknowledgements

We extend our heartfelt gratitude to the three distinguished contestants—two National Mathematical Olympiad gold medalists and one silver medalist—for their invaluable evaluations of our synthetic theorems. We also express our sincere thanks to their coach Zhibin Liang, whose efforts made this collaboration possible. Furthermore, we deeply appreciate the insightful discussions from Jiajun Song, Yuxuan Wang, and Dr. Chi Zhang at Beijing Institute for General Artificial Intelligence. This work was supported in part by the National Natural Science Foundation of China under Grants 61976214.

References

Judea Pearl. Graphical models for probabilistic and causal reasoning. *Quantified representation of uncertainty and imprecision*, pages 367–389, 1998.

W-T Wu. On the decision problem and the mechanization of theorem proving in elementary geometry. *Scientia Sinica*, 21:157–179, 1978.

Shang-Ching Chou, Xiao-Shan Gao, and Jing-Zhong Zhang. A deductive database approach to automated geometry theorem proving and discovering. *Journal of Automated Reasoning*, 25(3): 219–246, 2000.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems, 2021. *URL https://arxiv. org/abs/2110.14168*, 2021.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. Minif2f: a cross-system benchmark for formal olympiad-level mathematics. *arXiv preprint arXiv:2109.00110*, 2021.

Yuhuai Wu, Albert Qiaochu Jiang, Jimmy Ba, and Roger Grosse. Int: An inequality benchmark for evaluating generalization in theorem proving. *arXiv preprint arXiv:2007.02924*, 2020.

Hongming Zhang and Tianyang Yu. Alphazero. *Deep Reinforcement Learning: Fundamentals, Research and Applications*, pages 391–415, 2020.

Guillaume Lample, Timothee Lacroix, Marie-Anne Lachaux, Aurelien Rodriguez, Amaury Hayat, Thibaut Lavril, Gabriel Ebner, and Xavier Martinet. Hypertree proof search for neural theorem proving. *Advances in neural information processing systems*, 35:26337–26349, 2022.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning. *arXiv preprint arXiv:2202.01344*, 2022.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J Prenger, and Animashree Anandkumar. Leandojo: Theorem proving with retrieval-augmented language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Peiyang Song, Kaiyu Yang, and Anima Anandkumar. Towards large language models as copilots for theorem proving in lean. *arXiv preprint arXiv:2404.12534*, 2024.

John Harrison, Josef Urban, and Freek Wiedijk. History of interactive theorem proving. In *Handbook of the History of Logic*, volume 9, pages 135–214. Elsevier, 2014.

Leonardo De Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. The lean theorem prover (system description). In *Automated Deduction-CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings 25*, pages 378–388. Springer, 2015.

Bruno Barras, Samuel Boutin, Cristina Cornes, Judicaël Courant, Yann Coscoy, David Delahaye, Daniel de Rauglaudre, Jean-Christophe Filliâtre, Eduardo Giménez, Hugo Herbelin, et al. The coq proof assistant reference manual. *INRIA, version*, 6(11), 1999.

Tobias Nipkow, Markus Wenzel, and Lawrence C Paulson. *Isabelle/HOL: a proof assistant for higher-order logic*. Springer, 2002.

Wenda Li, Lei Yu, Yuhuai Wu, and Lawrence C Paulson. Isarstep: a benchmark for high-level mathematical reasoning. *arXiv preprint arXiv:2006.09265*, 2020.

Floris van Doorn, Gabriel Ebner, and Robert Y Lewis. Maintaining a library of formal mathematics. In *International Conference on Intelligent Computer Mathematics*, pages 251–267. Springer, 2020.

Kaiyu Yang and Jia Deng. Learning to prove theorems via interacting with proof assistants. In *International Conference on Machine Learning*, pages 6984–6994. PMLR, 2019.

Georges Gonthier et al. Formal proof–the four-color theorem. *Notices of the AMS*, 55(11):1382–1393, 2008.

Peter Scholze. Liquid tensor experiment. *Experimental Mathematics*, 31(2):349–354, 2022.

Kevin Buzzard and Richard Taylor. A lean proof of fermat's last theorem. 2024.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of a. *Journal of the ACM (JACM)*, 32(3):505–536, 1985.

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

Chengwu Liu, Jianhao Shen, Huajian Xin, Zhengying Liu, Ye Yuan, Haiming Wang, Wei Ju, Chuanyang Zheng, Yichun Yin, Lin Li, et al. Fimo: A challenge formal dataset for automated theorem proving. *arXiv preprint arXiv:2309.04295*, 2023.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*, 2023.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Jason Rute, Miroslav Olšák, Lasse Blaauwbroek, Fidel Ivan Schaposnik Massolo, Jelle Piepenbrock, and Vasily Pestun. Graph2tac: Learning hierarchical representations of math concepts in theorem proving. *arXiv preprint arXiv:2401.02949*, 2024.

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: Yes.

   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: At the end of the conclusion part.

   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]

Justification: No related results.

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: See supplementary material

   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: [No]

Justification: We only provide dataset.

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: See supplementary material.

   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: [No]

   Justification: These are not related to this paper.

   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).
   - 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: See our paper.

   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 conforms with it.

   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: [No]

    Justification: Not related.

    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: 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: We are the owner.

    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.
    - 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: [Yes]

Justification: The synthetic dataset.

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: not related.

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: not related.

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.