

# Optimizing generative AI by backpropagating language model feedback

Corresponding Author: Professor James Zou

**This file contains all reviewer reports in order by version, followed by all author rebuttals in order by version.**

Version 0:

Reviewer comments:

Referee #1

(Remarks to the Author)

The paper "TEXTGRAD: Automatic 'Differentiation' via Text" introduces a novel framework designed to optimize complex AI systems built upon large language models (LLMs). The authors draw an analogy to traditional backpropagation in neural networks, leveraging textual feedback as gradients to refine various components within a computation graph. The framework has been tested across a diverse range of applications including question-answering, code optimization, molecular design, and radiotherapy treatment planning, demonstrating significant improvements in performance metrics.

Strengths:

1. Innovation and Originality:

- The concept of using textual gradients and metaphorically aligning them with traditional numerical gradient-based backpropagation is both novel and creative. It opens a new frontier for optimization in AI systems.
- The framework is also general and versatile, applicable to multiple domains without needing domain-specific adjustments. This showcases the potential broad applicability of the approach.

2. Technical Soundness:

- The implementation of TEXTGRAD is robust, utilizing existing LLMs in a unique optimization paradigm. The method integrates seamlessly with existing machine learning frameworks like PyTorch, which is beneficial for ease of use and adoption.
- The experiments are thorough and demonstrate clear performance benefits across varied tasks, underscoring the utility and effectiveness of the framework.

3. Broader Impact:

- By improving the performance of complex AI systems and making optimization more accessible, TEXTGRAD could accelerate research and innovation in multiple fields, particularly in areas like drug discovery and medical treatment planning.
- The open-source nature of the framework encourages widespread use and further improvements from the research community.

Weaknesses:

1. Scalability and Efficiency:

- The paper does not provide sufficient details about the computational efficiency and scalability of TEXTGRAD in large-scale applications. An analysis of the computational overhead and performance trade-offs would significantly strengthen the work.

2. Comparative Analysis:

- While the paper makes some comparisons against state-of-the-art methods, it would benefit from a more extensive set of comparisons with other recent advancements in AI optimization. Including baseline results from a wider set of benchmarks and methods would provide a more comprehensive performance assessment.

3. Contextual Limitations:

- The reliance on LLMs, which are typically large and require significant computational resources, might limit the

accessibility of TEXTGRAD in resource-constrained environments. Discussing potential mitigations or adaptations for more lightweight models could enhance the paper's relevance.

In addition, the way the paper is written in many places may confuse readers or make them feel that the paper is over-claiming. For example, if I understand correctly, the components being optimized through TEXTGRAD in an AI system are textual contents like prompts. However, the paper may give the readers the impression that it introduces a new way to optimize a broader set of components, including model parameters, in an AI system. For instance, the abstract states, "Inspired by this, we introduce TEXTGRAD, a powerful framework performing automatic 'differentiation' via text. TEXTGRAD backpropagates textual feedback provided by LLMs to improve individual components of a compound AI system."

My suggestion:

While the paper "TEXTGRAD: Automatic 'Differentiation' via Text" introduces an innovative and potentially impactful framework, I do not think it meets the publication standard of Nature, unless my concerns can be well addressed. Specifically, the issues related to scalability, computational efficiency, comparative analysis, and contextual limitations need to be thoroughly examined and resolved to ensure the robustness and broader applicability of the framework.

(Remarks on code availability)

The code is a good one.

Referee #2

(Remarks to the Author)

Yuksekgonul et al. introduces TextGrad as a strategy for "performing automatic differentiation via text". Claims are made in the abstract as to improvements in question and answering (51% to 55%), LeetCode-Hard coding problem solutions, and the design of druglike small molecules and high-specificity oncology treatment plans. The premise of the work is that SOTA machine learning approaches may require expertise to design, but can be posed as a computation graph that is modified/improved by feedback from an LLM. TextGrad is introduced as a new framework with analogy to how PyTorch can be used for parameter optimization in traditional supervised learning. There are two 'modes' of optimization: optimization of instances and optimization of prompts. These capabilities are enabled through engineered prompts that explain the high-level goal, include mention of past iterations to provide momentum in SGD, etc.

Overall, this work tries to demonstrate the generality and broad relevance of the framework to several tasks across several domains. The manuscript is written clearly, formatted nicely, and is bold in its claims. However, I believe that with the exception of the standard benchmarks used to evaluate LLMs and prompt engineering (the first two demonstrations), the demonstrations on molecular design and radiotherapy treatment are shallow and do not show evidence of performance that exceeds what is already capable with other, sometimes task-specific and sometimes non-ML approaches. It is already clear that one *can* use LLMs to approach these tasks, but it is not clear that one *should* and this paper does not illustrate their value adequately, especially in the case of drug discovery. Even setting aside that this is a toy task, no comparisons are made to show SOTA or near-SOTA performance. My comments below around the drug discovery example might appear harsh, but I believe it is irresponsible to have overstated claims about the role of AI, let alone LLMs, for such tasks that are significantly more nuanced than what is discussed or evaluated. I cannot be supportive of publication based on this section, and hope that other experts for the other demonstrations can speak to the significance of those as well.

The description of the strategy relies on pseudo-mathematical statements about partial derivatives that are not true derivatives, i.e., empirical applications of chain rule-like reasoning. Conceptually, what is reported bears a strong resemblance to ideas in self-refinement and other cited works like Reflexion. The relationship to other prompt optimization frameworks is discussed in the Related Work, including ProTeGi, which uses the exact same textual gradient analogy. This work packages these ideas into a convenient framework that may not represent an entirely novel conceptual advance, but is presented in an intuitive manner.

Modest improvements in performance are observed for a few different tasks (e.g., Table 2); I am not the best reviewer to judge the significance of these differences. Table 3 shows performance matching DSPy on two tasks and moderately better performance on a third.

It is not clear how some of the hyperparameters for the model are chosen, e.g., how many iterations to use for refinement (sometimes 3, sometimes 12). Does performance deteriorate if too many iterations are used?

To comment on the chemistry and drug discovery related tasks from the perspective of a domain expert, there is a lot I am concerned about. The significance of a Vina score and QED value are overstated. QED is an extraordinarily simple and uninformative metric that has been overused in computational benchmarks, but is trivial to maximize with a score of 0.948 using almost any molecular generative model. QED does not inherently favor molecules with "better absorption properties" and docking scores don't inherently prefer "many functional groups". Comparing a Vina score of a generated molecule to a clinically approved molecule is not evidence that a better molecule has been found. The trajectory does show steady improvement from the starting molecule, but the seed molecules are artificially poor binders. If the optimization is continued for more iterations, are even better molecules identified? The claim that TextGrad produces competitive results without a prior training set is also highly flawed, as the base LLM has been trained on data including knowledge of molecular structures, approved drugs, and various other aspects of molecular design. This goes beyond memorization of approved drugs for specific protein targets and is not mitigated by an evaluation of Tanimoto similarity to ChEMBL molecules. The

explanations provided by the model that are claimed to be informative/interpretable are shallow and could be summarized in the most basic of introductory articles about medicinal chemistry “moves” and basic concepts in binding; moreover, they do not seem specific to the protein target of interest and instead are generic comments about what kind of interactions might be relevant for binding in general. The evaluation of this molecular optimization task lacks any discussion of other approaches, whether based on generative design or based on virtual screening of known molecules from compound libraries. Section F.2 claims that the `vina_qed_ratio` can be tuned to change the tradeoff but no results are shown to indicate that the scalarization of the two objectives is quantitatively respected by the model. I believe that defending claims of utility for drug design requires a stricter standard than claims of improving performance on standardized Q&A/LeetCode benchmarks.

For the application of radiotherapy treatment plan optimization, what is the performance of a task-specific optimizer that approaches the problem as a numerical optimization? Or any other nested optimization that directly tries to optimize the performance metrics reported in Figure 3de? Though I am not an expert in this application, I suspect there are plausible baselines that have been omitted. Details are not provided as to the origin for the “human-optimized” plans.

Referee #3

(Remarks to the Author)

A. Summary of the key results

The paper introduces TextGrad, a framework analogous to backpropagation and automatic differentiation, designed for optimizing text prompts or instances through backpropagated textual feedback. This general and effective approach is applied across diverse scenarios: optimizing code snippets, molecular structures, question-answering, and designing radiation oncology treatment plans. The results demonstrate that the proposed method significantly improves the performance of LLM agents. Additionally, the authors present a library that enables developers to easily apply the approach using programming syntax similar to PyTorch.

B. Originality and significance: if not novel, please include reference

The use of textual feedback as an optimizer is not new and has been discussed in the literature (see, for example, the paper list <https://github.com/AGI-Edgerunners/LLM-Optimizers-Papers>). To me, the method of using textual feedback as a gradient is similar to approaches found in [1] and [2], as cited in this paper. However, to the best of my knowledge, none of the prior works provides a general framework for implementing this idea, nor do they leverage PyTorch's programming syntax to allow developers to easily construct complex, domain-specific applications. The authors demonstrate that their library enables this approach to be applied across a range of diverse and knowledge-intensive fields. In the same way that deep learning has benefited from libraries that simplify implementation, making it accessible to various fields, the proposed framework simplifies the use of LLM as an optimizer. I can see the potential impact of this framework being significant.

[1] Large Language Models as Optimizers

[2] Automatic Prompt Optimization with “Gradient Descent” and Beam Search

C. Data & methodology: validity of approach, quality of data, quality of presentation

Overall, the proposed approach is well-motivated and clearly articulated. The authors validate the framework across a diverse range of applications, effectively demonstrating its utility. The experimental design is both solid and reasonable, and the results strongly support the authors' claims. I have several suggestions and questions about the experiment settings, please see Section F.

D. Appropriate use of statistics and treatment of uncertainties

Certain factors may introduce variance in the evaluation process. Additionally, some experimental results rely on models released by OpenAI, which may evolve or be removed over time. In such cases, future attempts to reproduce the results might not be possible. Therefore, it would be great to address these uncertainties and discuss their potential impact on reproducibility.

E. Conclusions: robustness, validity, reliability

Overall, the experiment results demonstrate the effectiveness of the proposed framework. With testing the approach in various datasets and tasks, the authors also demonstrate the robustness of the approach. See more comments and discussion in Section F.

F. Suggested improvements: experiments, data for possible revision

Please see comments/suggestions in the following:

1. I'm curious whether the proposed approach can be applied to optimizing few-shot examples in an in-context learning setting.
2. Please correct me if I'm wrong — while the proposed framework is general, the applications discussed in the paper don't seem to involve complex backpropagation chains and multiple variables, as shown in Figure 1(b). I acknowledge that the authors have demonstrated the framework's effectiveness in several real-world tasks, but it would be interesting to test it in more complex agent-based tasks, such as WebArena, which require more intricate actions and tool use.
3. I wonder how robust the prompts or instances identified by this framework are. Specifically, how does the performance hold up when applying prompts or instances obtained from one LLM (e.g., GPT-4) to another (e.g., Gemini)?
4. I wonder how “correct” the textual feedback provided by the LLMs in this framework is. If the feedback is inaccurate or varies over time, this could potentially impact the optimization process.

5. There are several self-reflection and self-correction approaches in the literature (e.g., see the survey: <https://arxiv.org/abs/2308.03188>). I am curious how the proposed approach compares to these.

G. References: appropriate credit to previous work?

Overall, the paper provides appropriate references to related work. However, I would like to see more discussion with related concurrent work such as “LLMs as optimizer” as it seems to me the idea is quite similar at first glance.

H. Clarity and context: lucidity of abstract/summary, appropriateness of abstract, introduction and conclusions

The paper is well-written. The abstract and introduction provide strong motivation and clear intuition for the work. It articulates the details of the proposed framework with illustrative examples and provides excellent real-world applications that demonstrate how the framework can be used.

(Remarks on code availability)

I did a light review of the

Version 1:

Reviewer comments:

Referee #1

(Remarks to the Author)

The authors have made commendable efforts to address and incorporate the feedback provided by all reviewers. I found the reading process quite enlightening, as it offered new insights and knowledge. The work presented is certainly both interesting and valuable.

However, after careful consideration, particularly in light of the points raised by reviewer 3, I still feel that the manuscript may not fully align with the publication standards of Nature. While this assessment is inherently subjective, it is important to note that a framework such as Auto-Grad, which can be likened to Pytorch in its domain, might not require the platform of Nature for its dissemination and recognition. At its current stage, a highly influential application or very broad usage might be necessary to meet the criteria for publication in Nature.

Referee #2

(Remarks to the Author)

The authors have made numerous changes to the manuscript that I believe has strengthened it and clarified its contributions. I'm supportive of its publication.

1. I appreciate the additional evaluations using a smaller base model and the discussion of cost required for optimization.

2. I appreciate the additional contextualization with respect to Reflexion, OPRO, and DSPy. The emphasis on (and benefit of) generality is noted.

3. I appreciate the decision to move the molecular optimization example to the Appendix and the additional caveats of the analysis (i.e., the existence of many additional considerations in molecular design). I know my initial comments were very picky about the language, so I am grateful that they were considered seriously. The inclusion of comparisons from the PMO benchmark provide better evidence of strong empirical performance in this proof of concept.

When skimming some of the examples in the appendices, I noticed in D.2 that the GPQA Diamond example has misformatted superscripts; might be worth checking others.

Referee #3

(Remarks to the Author)

I would like to thank the authors for thoroughly addressing all the feedback and for including new experiments on MathVista and ScienceQA, as well as exploring the setting of optimizing compound AI systems.

The results on MathVista and ScienceQA are promising, with TextGrad demonstrating improvements over strong published baselines in both cases. These experiments address my concerns.

However, the claim regarding the reduction of hallucinations in the multimodal language model for VQA is not convincing to me. First, the authors mention that the experiment is only on "the spatial relation type, using 200 images from this subset". In such a case, the authors should only claim that TextGrad improves spatial reasoning in vision-language models. Second, it is unclear why the authors chose to conduct experiments on HQH rather than other well-established VQA or spatial relation benchmarks. Third, the definition of hallucination in HQH seems problematic. For example, in Figure 3d, the initial answer appears to involve an object recognition error rather than what is typically considered hallucination. In vision-language, hallucination is not a well-defined term, but it is generally understood as VL models generate captions that include objects not present in the input images. However, HQH seems to consider all types of errors as hallucinations. I would recommend avoiding the use of such a vague term without further clarification.

Overall, the paper is interesting and demonstrates that TextGrad can be effectively applied across various scenarios. Therefore, I lean into accepting the paper. However, I encourage the authors to dive deeper into each experimental scenario, providing intuition on how TextGrad contributes to performance improvements and ensuring that all claims are supported by the results.

**Open Access** This Peer Review File is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

In cases where reviewers are anonymous, credit should be given to 'Anonymous Referee' and the source.

The images or other third party material in this Peer Review File are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

To view a copy of this license, visit <https://creativecommons.org/licenses/by/4.0/>

# Reply to the Reviewers

Manuscript 2024-06-12036

TextGrad: Automatic “Differentiation” via Text  
Nature

## Overall Comments to Reviewers

We sincerely thank the reviewers for their thorough evaluation of our manuscript.

We thank reviewers for recognizing **TEXTGRAD** as a versatile and intuitive framework that will have substantial impact. We are excited that Reviewer 1 found our work “both novel and creative” and noted that it “opens a new frontier for optimization in AI systems.”. We appreciate Reviewer 1’s recognition of **TEXTGRAD**’s “general and versatile” nature and its potential for “broad applicability.” We are grateful that Reviewer 2 found our manuscript to be “written clearly, formatted nicely”, and found our framework “convenient”. We also thank Reviewer 3 for highlighting that our framework “simplifies the use of LLM as an optimizer” and noting that they “can see the potential impact of this framework being significant.”

The reviewers’ insights have been invaluable in helping us improve our work. As Reviewer 1 stated, we believe **TEXTGRAD** can “accelerate research and innovation in multiple fields”. We have carefully incorporated all of the reviewers’ feedback in our updated manuscript. To this end, we have added substantial new experiments to demonstrate the significant improvements achieved by **TEXTGRAD** compared to other sophisticated approaches across diverse settings.

## High-level summary of new results

We performed extensive experiments to demonstrate the generality, scalability, and robustness of **TEXTGRAD**. Below is a list of all the new results, which we explain in detail in the individual responses to reviewers.

**1. Adding new experiments in new settings:** We explore new and more challenging settings to expand the applications of **TEXTGRAD** from the initial submission.

- **Compound AI systems:** We explore optimizing compound AI systems in two new domains: i) agentic systems that use tools and retrieval, and ii) optimizing prompts and instances simultaneously to reduce hallucinations (Figure 2). We optimize the solution chains from Chameleon [1], an agentic system that uses tools to solve visual question answering problems, and improve its performance from 77.5% to 85.2%. Further, in the hallucination benchmark HQH [2], we reduce the failure rate of gpt-4o from 30% to 21% (new Figure 3 in the main paper).
- **Multimodal solution optimization:** We added applications of **TEXTGRAD** to multimodal domains such as scientific question answering (Table 1), where **TEXTGRAD** improves the accuracy from 63.8% to 66.1% on ScienceQA [3], and from 91.6% to 92.4% on MathVista [4] (new Table 2 in the main paper).

- **Optimizing in-context demonstrations** We demonstrate that not only instructions, but also the in-context demonstrations can be optimized using Textual Gradients (Table 2). In particular, without making any changes to our framework, we can optimize the in context demonstrations to improve prompt optimization, and this brings consistent performance gains in our prompt optimization benchmarks (new Supplementary Table 5 in the main paper).

**2. Expanding our experiments with new baselines:** To better ground our work, we added new baselines to our experiments as requested by reviewers:

- **Baselines for language modeling benchmarks:** First, for the language modeling benchmarks, Table 2 includes the new OPRO [5] baseline for prompt optimization, Table 1 includes the Reflexion [6] baseline for instance optimization. Figure 1 shows that in solution optimization and prompt optimization, **TEXTGRAD** achieves results beyond the Pareto frontier of cost-accuracy of existing approaches (new Supplementary Figure 8 in the main paper).
- **Baselines for molecule optimization experiments:** Second, for molecule optimization, Figure 7 includes 6 baselines from the *Practical Molecule Optimization* benchmark [7], including methods based on Monte Carlo Tree Search, Reinforcement Learning, and Genetic Algorithms. Compared to these baselines, **TEXTGRAD** converges faster, and also achieves better Vina scores across 58 proteins (new Supplementary Figure 7 in the main paper).
- **Baselines for treatment plan optimization:** Third, in our radiotherapy treatment planning study, we added a new baseline with a Bayesian Optimization (BO) algorithm similar to Wang *et al.* [8]. In Table 5, we find that **TEXTGRAD**-optimized plans achieve significantly better PTV than the BO algorithm (new Supplementary Table 1 in the main paper).

**3. Studying the cost and robustness of **TEXTGRAD**:** Based on comments from our reviewers, we extensively studied the feasibility (cost and scalability) and the robustness of optimization using **TEXTGRAD** in various applications. In particular:

- **Cost-accuracy frontier:** First, we study the feasibility of **TEXTGRAD** by exploring the Cost-Accuracy frontier (Figure 1) in prompt optimization and solution optimization. For instance, we find that in all of our settings, prompt optimization has a fixed cost of  $< \$1$ , and solution optimization has a cost of \$0.10 per instance. **TEXTGRAD** achieves results beyond the Pareto frontier of existing approaches (new Supplementary Figure 8 in the main paper).
- **Optimization using open-source models:** Second, we explore performing optimization using open-source models in Table 3. We find that recent open-source models such as the Llama-3.1 family [9] can also be capable optimizers. For instance, by optimizing prompts for Llama-3.1-70B using itself, we can push its performance on Object Counting from 85.9% to 93.1%, and on Word Sorting from 87.9% to 95.8%. As open-source models get better by the day, we expect **TEXTGRAD** using an open-source model as an optimizer to be competitive with optimizers based on closed-source models (new Supplementary Table 3 in the main paper).

- **Robustness of prompts and transferability to open-source models:** Third, we study the robustness of prompts learned using closed-source models in Table 4. We find prompts are often transferable to open-source models, providing another avenue for improving the performance of open-source models. For instance, by using the prompts optimized for gpt-3.5-turbo-0125, we can improve the performance of Qwen 2.5 - 32B on Object Counting from 83.8% to 92.9%, and on Word Sorting from 47.5% to 55.6% (new Supplementary Table 4 in the main paper).
- **Steerability in multi-objective molecule optimization:** Fourth, we analyze the steerability of **TEXTGRAD**'s solutions in multi-objective molecule optimization in Figure 8, and find that **TEXTGRAD** is able to alter the distribution of generated molecules in response to a user tuned prioritization of the Vina and QED metrics (new Supplementary Figure 5 in the main paper).
- **Sensitivity analysis in molecule optimization:** Fifth, in molecule optimization, we perform a sensitivity analysis study to analyze the depth and specificity of **TEXTGRAD** gradients for targeted ligand design. We find that **TEXTGRAD**'s feedback are specific to the optimized molecule and protein. An ablation with randomized, promiscuously applied gradients produced molecules that were significantly inferior across multiple properties (new Supplementary Figure 6).

**4. Clarifications:** Finally, we added clarifications to the writing and added further discussion on relevant work as requested by reviewers, such as adding a more detailed discussion on OPRO [5] and using a more careful language with the claims in the molecule optimization section.

**A note on generality:** Importantly, a crucial novelty and benefit of our framework is its generality. For instance, looking at Reflexion or OPRO, we see that for each task, there is a need to create separate implementations, prompts, and a user has to go through a great deal of effort to actually use their approaches effectively. Achieving meaningful improvements with these methods requires such a great effort that a *user essentially needs to redesign existing methods, or invent new methods to apply to new tasks.*

Generality is one novel and key benefit of **TEXTGRAD**. Once viewing optimization as the first-class citizen task, achieving performant results in novel applications becomes turnkey. In particular, a user need not modify the framework to apply to a new domain. **TEXTGRAD** works out of the box for all the domains we tested without any modifications, and we spent a great deal of work to make sure optimization works as broadly as possible.

For clarity, we use **green** to indicate original reviewer comments. We use **orange** to highlight revisions we made to our writing, both here and the revised version of the manuscript. We use black to respond to reviewer's comments here.

---

# 1 Responses to Reviewer 1

## Comment #1.1

*The paper "TEXTGRAD: Automatic 'Differentiation' via Text" introduces a novel framework designed to optimize complex AI systems built upon large language models (LLMs). The authors draw an analogy to traditional backpropagation in neural networks, leveraging textual feedback as gradients to refine various components within a computation graph. The framework has been tested across a diverse range of applications including question-answering, code optimization, molecular design, and radiotherapy treatment planning, demonstrating significant improvements in performance metrics.*

*Strengths:*

- The concept of using textual gradients and metaphorically aligning them with traditional numerical gradient-based backpropagation is both novel and creative. It opens a new frontier for optimization in AI systems.*
- The framework is general and versatile, applicable to multiple domains without needing domain-specific adjustments. This showcases the potential broad applicability of the approach.*
- The implementation of TEXTGRAD is robust, utilizing existing LLMs in a unique optimization paradigm. The method integrates seamlessly with existing machine learning frameworks like PyTorch, which is beneficial for ease of use and adoption.*
- The experiments are thorough and demonstrate clear performance benefits across varied tasks, underscoring the utility and effectiveness of the framework.*
- By improving the performance of complex AI systems and making optimization more accessible, TEXTGRAD could accelerate research and innovation in multiple fields, particularly in areas like drug discovery and medical treatment planning.*
- The open-source nature of the framework encourages widespread use and further improvements from the research community.*

## Response to #1.1

We sincerely thank you for your thorough evaluation of our manuscript and its strengths. We appreciate your recognition of **TEXTGRAD** as "both novel and creative" and noting that it "opens a new frontier for optimization in AI systems." We are grateful for your acknowledgment of **TEXTGRAD**'s "general and versatile" nature and its potential for "broad applicability."

## Comment #1.2

*Weakness: The paper does not provide sufficient details about the computational efficiency and scalability of TEXTGRAD in large-scale applications. An analysis of the computational overhead and performance trade-offs would significantly strengthen the work.*

## Response to #1.2

Thank you for this great suggestion! To give further results on the scalability and efficiency of **TEXTGRAD**, we study the cost of running optimization along with the cost accuracy frontier across a variety of methods.

### Cost vs Accuracy Frontier:

First, we report the cost-accuracy frontier of prompt optimization in Figure 1. In particular, we compare the fixed cost of optimizing a prompt for gpt-3.5-turbo-0125 using the existing techniques to the performance obtained via that method. This cost is paid once to optimize a prompt. On x-axis, we report this cost, and on the y-axis, we report the performance.

Overall, we highlight two observations. First, **TEXTGRAD** is always at the Pareto frontier. Second, the cost of running prompt optimization is often  $< \$1$ , highlighting that prompt optimization is inexpensive, as it only needs to be done once and then the optimized prompt can be reused.

Secondly, we report the cost-accuracy frontier for instance optimization also in Figure 1. Here, we report the average cost of running instance optimization across all the test instances. We again observe that **TEXTGRAD** is at the Pareto frontier here, and the cost of running optimization per instance is usually  $< \$0.10$ . This again highlights that running optimization using these models is often acceptable.

We provide further results on the scalability and accessibility in the answer to Comment 1.4.

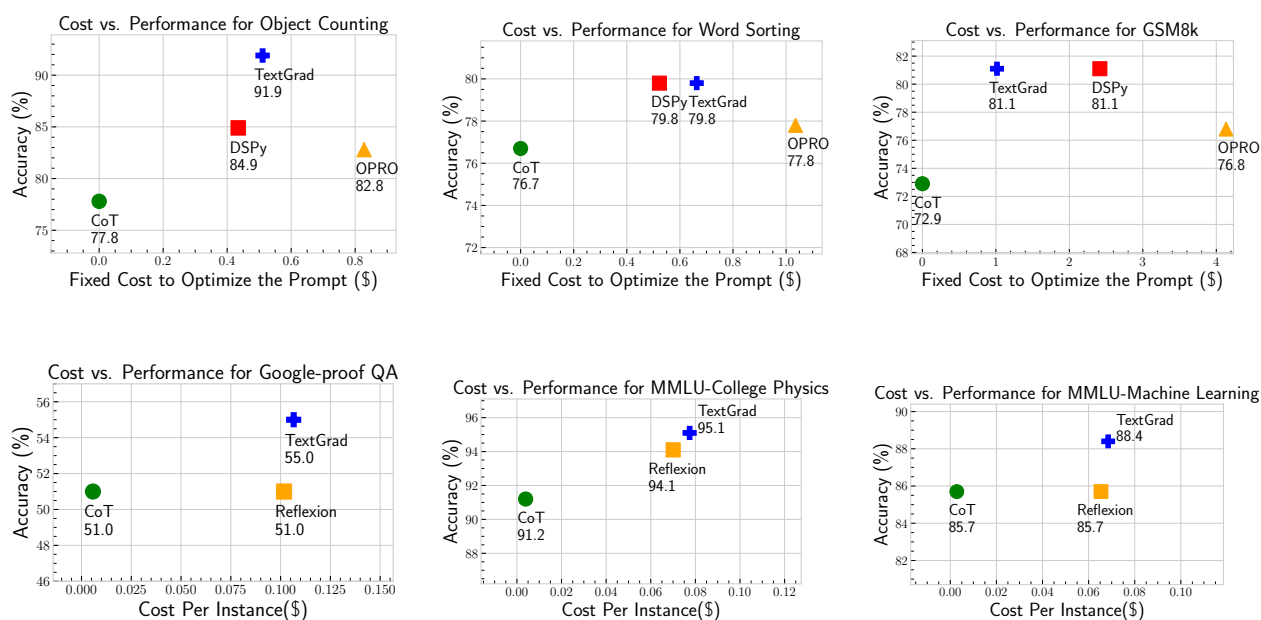


Figure 1: (new Supp. Figure 8 in the paper) Cost-Accuracy frontier for prompt optimization (top row) and instance optimization (bottom row). We observe that prompt optimization often has a cost < \$1, and instance optimization often has a cost of < 0.1\$, highlighting that optimization with **TEXTGRAD** is inexpensive.

### Comment #1.3

*While the paper makes some comparisons against state-of-the-art methods, it would benefit from a more extensive set of comparisons with other recent advancements in AI optimization. Including baseline results from a wider set of benchmarks and methods would provide a more comprehensive performance assessment.*

### Response to #1.3

Thank you for this suggestion! We have further expanded our experiments in several distinct ways.

### New setting 1: optimization in multimodal vision and language problems

We expand solution optimization to multimodal problems, specifically the MathVista [4] and the ScienceQA [3] tasks. MathVista is a comprehensive benchmark for evaluating models' capabilities in mathematical reasoning with visual contexts. It contains seven mathematical reasoning categories including arithmetic reasoning, algebraic reasoning, and statistical reasoning, and requires fine-grained visual understanding across natural images, diagrams, charts, and graphs. We utilize the *testmini* split with 1,000 examples for evaluation. ScienceQA is a multimodal benchmark of multiple-choice questions in scientific domains. It covers three subjects: nature science, social science, and language science, featuring a diverse set of domains and topics. We conduct experiments on ScienceQA-IMG, the *test* split in multimodal setting, which consists of 2,017 examples.

On both tasks, we test gpt-4o using a zero-shot chain-of-thought (CoT) prompting strategy for inference and follow default evaluation pipelines [3, 4]. **TEXTGRAD** is employed to optimize the solutions generated by gpt-4o, with solution refinement objectives defined as follows:

#### Solution Refinement Objective for Multimodal Question Answering

##### **For visual mathematical reasoning (MathVista):**

**LLM**("Please evaluate the existing answer to the visual math problem without solving it yourself. Verify that the answer provides accurate reasoning logic to address the question.

Image: {Image data}

Question: {Question}

Answer by the language model: {**Solution**}")

##### **For visual scientific reasoning (ScienceQA-IMG):**

**LLM**("Please evaluate the existing answer to the visual scientific problem without solving it yourself. Verify that the answer accurately understands the image, provides appropriate knowledge and reasoning logic to address the question.

Image: {Image data}

Question: {Question}

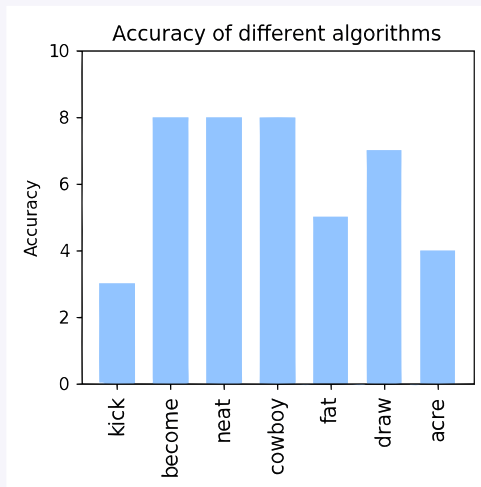
Answer by the language model: {**Solution**}")

The evaluation results are presented in Table 1. We compare **TEXTGRAD** and gpt-4o with CoT [10, 11] and gpt-4o with Reflexion, a widely used method that uses verbal feedback for self-improvement [6]. Notably, compared to Reflexion, **TEXTGRAD** achieves performance gains of 2.1% and 2.2% on MathVista and ScienceQA-IMG, respectively. **TEXTGRAD** also consistently outperformed all of these alternative approaches in the natural language problem solving tasks (Google-proof QA, MMLU-Machine Learning and MMLU-College Physics).

Table 1: (New Table 2) Solution optimization for zero-shot question answering with gpt-4o. **TEXTGRAD** outperforms the baselines consistently across different tasks.

| Dataset                                       | Method                    | Accuracy (%) |
|-----------------------------------------------|---------------------------|--------------|
| <b>Natural Language Problem Solving Tasks</b> |                           |              |
| Google-proof QA [12]                          | CoT [10, 11]              | 51.0         |
|                                               | Reflexion [6]             | 51.0         |
|                                               | GPT-4o best reported [13] | 53.6         |
|                                               | <b>TEXTGRAD</b>           | <b>55.0</b>  |
| MMLU-Machine Learning [14]                    | CoT [10, 11]              | 85.7         |
|                                               | Reflexion [6]             | 85.7         |
|                                               | <b>TEXTGRAD</b>           | <b>88.4</b>  |
| MMLU-College Physics [14]                     | CoT [10, 11]              | 91.2         |
|                                               | Reflexion [6]             | 94.1         |
|                                               | <b>TEXTGRAD</b>           | <b>95.1</b>  |
| <b>Multimodal Problem Solving Tasks</b>       |                           |              |
| MathVista [4]                                 | CoT [10, 11]              | 63.8         |
|                                               | Reflexion [6]             | 64.0         |
|                                               | <b>TEXTGRAD</b>           | <b>66.1</b>  |
| ScienceQA-IMG [3]                             | CoT [10, 11]              | 91.6         |
|                                               | Reflexion [6]             | 90.2         |
|                                               | <b>TEXTGRAD</b>           | <b>92.4</b>  |

The effectiveness of **TEXTGRAD** in multimodal settings is further illustrated through two examples below. The first example from MathVista involves interpreting elements in a bar chart and performing arithmetic calculation. Initially, gpt-4o misinterprets one of the bars in the chart, leading to an incorrect solution. **TEXTGRAD** provides critical feedback, identifying this perceptual error and enabling gpt-4o to refine its initial solution. The second example from ScienceQA-IMG is a scientific question that requires an accurate understanding of the biological process of seed formation in fruit trees. While the initial solution includes correct intermediate steps, it ultimately fails to arrive at the correct answer. **TEXTGRAD** verifies the correct steps and highlights the logical mistake that leads to the incorrect conclusion, guiding gpt-4o to correct itself.



**Question:** What is the sum of the accuracies of the algorithms fat and acre?

**Solution at initialization:**

To find the sum of the accuracies of the algorithms "fat" and "acre," we need to accurately interpret the values from the bar chart.

From the chart: The accuracy of "fat" is represented by **the fourth bar from the left, which shows an accuracy of 6**. The accuracy of "acre" is represented by the last bar on the right, which shows an accuracy of 4.

Now, we add these two values together:  **$6 + 4 = 10$ . The correct sum of the accuracies of the algorithms "fat" and "acre" is 10.**

**Output of TEXTGRAD's loss function based on multimodal context:**

The existing answer provides a clear and logical approach to solving the problem by interpreting the bar chart and summing the accuracies of the specified algorithms. However, **there is an error in the interpretation of the values from the chart**.

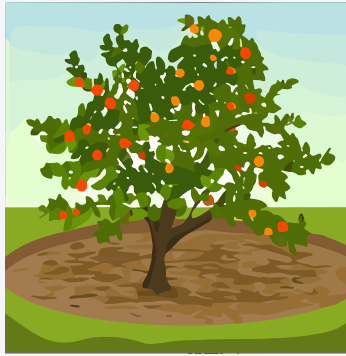
Upon reviewing the bar chart: **The accuracy of "fat" is represented by the fifth bar from the left, which shows an accuracy of 5**. The accuracy of "acre" is represented by the last bar on the right, which shows an accuracy of 4.

**Solution after optimization:**

To find the sum of the accuracies of the algorithms "fat" and "acre," we need to accurately interpret the values from the bar chart.

From the chart: The accuracy of "fat" is represented by **the fifth bar from the left, which shows an accuracy of 5**. The accuracy of "acre" is represented by the last bar on the right, which shows an accuracy of 4.

Now, we add these two values together:  **$5 + 4 = 9$ . The correct sum of the accuracies of the algorithms "fat" and "acre" is 9.**



**Question:** An adult orange tree can grow seeds. Which part of the orange tree makes the seeds? (A) the flowers (B) the fruit

**Solution at initialization:**

The image shows an orange tree with numerous fruits and flowers, indicating different stages of the reproductive process. Flowers are pollinated, leading to the fertilization of ovules, which develop into seeds. These seeds are then encased within the fruit as it forms.

The flowers of the orange tree are pollinated, leading to the development of seeds within the ovules of the flowers. These seeds are then encased within the fruit as the flower turns into the fruit. **Therefore, the part of the orange tree that makes the seeds is the fruit. Answer: (B) the fruit**

**Output of TEXTGRAD's loss function based on multimodal context:**

The existing answer provides a detailed explanation of the biological process of seed formation in fruit trees, which is accurate. It correctly identifies that flowers are pollinated, leading to the development of seeds within the ovules, and these seeds are then encased within the fruit. The answer also correctly interprets the image, noting the presence of both fruits and flowers on the orange tree.

However, **there is a slight misinterpretation in the final conclusion.** While the seeds are indeed found within the fruit, **the part of the tree that initially makes the seeds is the flower, as the seeds develop from the fertilized ovules within the flowers.**

**Solution after optimization:**

The image shows an orange tree with numerous fruits and flowers, indicating different stages of the reproductive process. Flowers are pollinated, leading to the fertilization of ovules, which develop into seeds. These seeds are then encased within the fruit as it forms.

In summary, the flowers of the orange tree are pollinated, leading to the development of seeds within the ovules of the flowers. These seeds are then encased within the fruit as the flower turns into the fruit. **Therefore, the part of the orange tree that initially makes the seeds is the flower. Answer: (A) the flowers**

## New setting 2: Optimizing compound AI systems

We expand **TEXTGRAD** for optimization in more complex systems involving multiple variables and longer chains of processes.

**Optimizing longer chains:** Our first study focuses on Chameleon [1], a sophisticated agent system involving multiple LLM modules using diverse tools. Chameleon previously reported state-of-the-art results across several reasoning-intensive tasks. It includes a module inventory of different tools and an LLM-based planner. Each module produces an output given the inputs from the previous models, and the final module generates the overall response. However, the lack of a feedback mechanism in the planning and tool execution processes can lead to error accumulation over time.

To address this, we use **TEXTGRAD** to optimize Chameleon by using textual gradients to adjust each module’s output at test time. The system first completes a standard forward pass, then iteratively optimizes each module and unrolls the forward pass, starting from the first and moving sequentially.

For evaluation, we use ScienceQA-IMG [3], a dataset requiring coordinated use of tools for visual perception, knowledge retrieval, and response generation. The module inventory includes tools for tasks like OCR (via EasyOCR [15]), image captioning, knowledge retrieval, solution generation, and answer generation. The results, as shown in Figure 2 Panel (e). **TEXTGRAD** achieves a 5.7% increase over the original Chameleon with two iterations of optimization. This improvement further extends to 7.7% with three iterations, indicating the benefits of multiple optimization iterations.

Overall, **TEXTGRAD** proves to be an effective method to significantly enhance the capabilities of compound systems like Chameleon in complex, multi-modal tasks that require coordinated use of diverse tools.

**Reducing hallucinations by jointly optimizing prompts and solutions:** In our second compound AI system application, we apply **TEXTGRAD** to reduce hallucinations in multimodal models, focusing on the Visual Question Answering (VQA) task. In VQA, images are presented alongside questions related to the content of the images. To guide the model in answering the questions, instructional prompts are provided, such as "Answer the question based on the image, let’s think step by step.". These instructional prompts help direct the model’s reasoning process. Then another language model evaluate the solution based on a loss function prompt. Here we use **TEXTGRAD** to jointly optimize the two sets of prompts—instructional prompt for the answer generation LLM and the loss prompt for the evaluation LLM—as well as the generated solution itself. All of these can be treated as variables in **TEXTGRAD** and are optimized jointly through our text-based chain rule. We show that this approach lead to an accuracy of 79% HQH VQA benchmark [2], a challenging task where multimodal language models often hallucinate, compared to an accuracy of 70% by gpt-4o with Chain-of-thought (CoT).

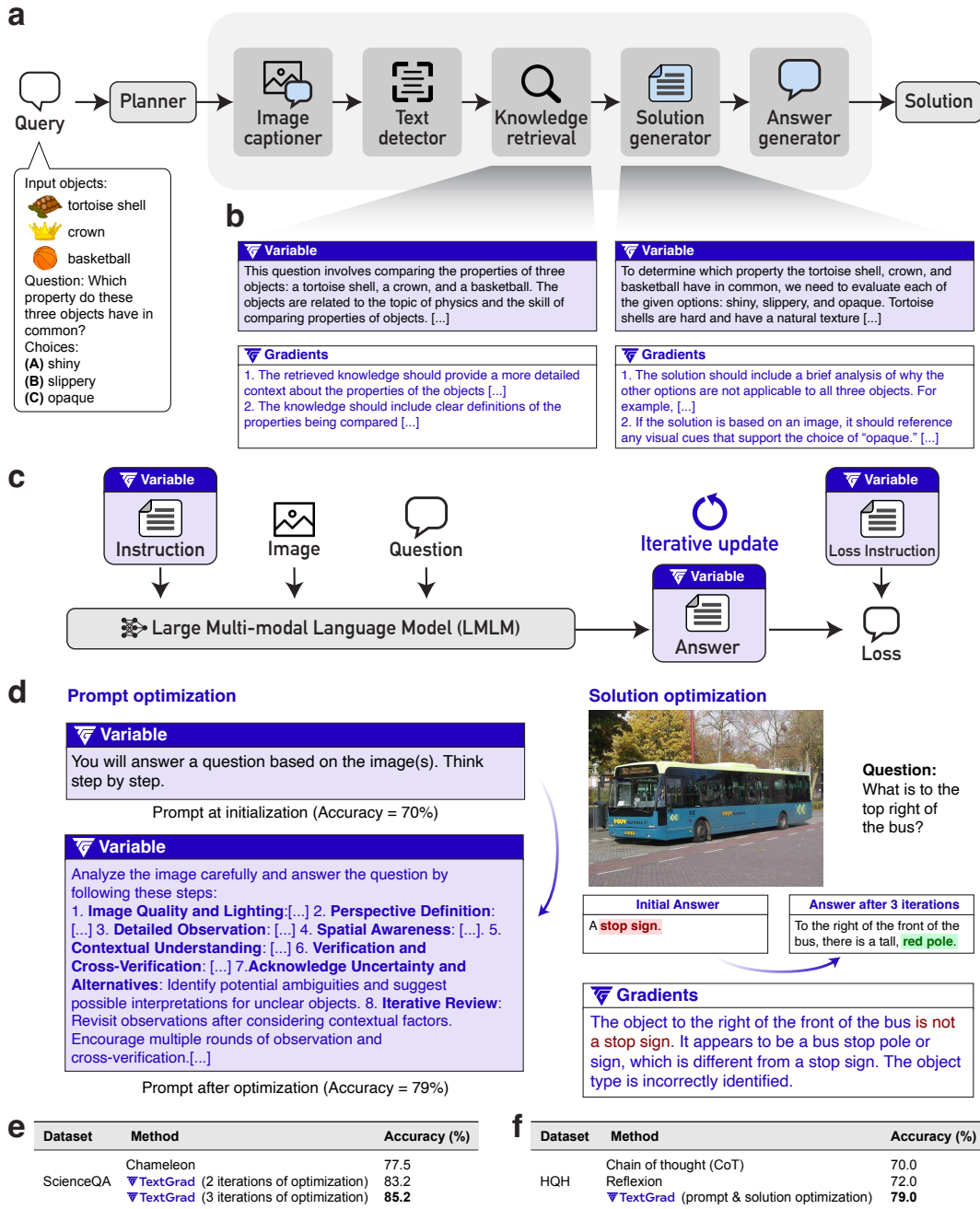


Figure 2: (New Figure 3 in the main paper) **Optimizing compound AI systems.** (a) **TEXTGRAD** enhances the solution generated by Chameleon, an agentic system that generates a plan followed by tool execution. We treat the output of each module as a separate variable to optimize and improve them. (b) Textual gradients are computed and backpropagated through the tool execution chain. (c) **TEXTGRAD** iteratively optimizes the answer to a multimodal question. To mitigate hallucinations of multimodal language models in visual question answering, we treat loss prompt, the instruction, and the solution as variables to jointly optimize and improve them. (d) With an example in HQH we show how **TEXTGRAD** jointly optimizes the prompts and solutions. **TEXTGRAD** yields accuracy improvements of 7.7% and 9% on ScienceQA (e) and HQH (f), respectively.

## Expanding the experiments in molecule optimization and treatment plan optimization

We included new baselines for molecule optimization, comparing 6 molecular optimization algorithms against **TEXTGRAD**. The comparison algorithms included optimization approaches from Monte Carlo Tree Search [16], Reinforcement Learning [17], Bayesian Optimization [18], and Genetic Algorithms [16], operating on both graph and text molecule representations. We benchmarked all methods on the task of ligand design against 58 protein targets in the DOCKSTRING benchmark, evaluated by simulated binding affinity using the Autodock Vina molecular docking software [19]. We compared performance across both the quality of the designed ligands (*in silico* binding affinity, e.g. Vina score) and oracle efficiency (number of docking simulations required), and determined that **TEXTGRAD** significantly outperformed all other methods, with the full results and sample trajectories shown in Figure 7. For treatment planning, we have explored a task-specific optimization approach using Bayesian Optimization (BO), similar to the work in [8], where a score function—a weighted sum of multiple plan quality metrics—was directly optimized. **TEXTGRAD** significantly outperforms the baseline in both efficiency and plan quality.

## Adding more baselines to existing experiments

We added new baselines to our experiments. In particular, we added OPRO [5], which uses LLMs as optimizers and was suggested by Reviewer 3, to our prompt optimization experiments and Reflexion [6] to all instance optimization tasks.

In Figure 1, Table 2, and Table 1 we report a new state-of-the-art baseline OPRO [5] and extend the state-of-the-art Reflexion [6] baseline to other instance optimization settings. Overall, **TEXTGRAD** outperforms both of these approaches in all of the settings we tested.

In addition to better performance, we would also like to highlight the generality of our framework. The codebase for Reflexion and OPRO have a separate implementation and/or prompting strategy for every task. For each task, there are separate implementations and prompts, making it nontrivial for users to apply these approaches to a new task.

With **TEXTGRAD**, a user does not need to modify the framework to apply to a new domain. It works out of the box for all the domains we tested without any modifications, and we spent a great deal of work to make sure optimization works as broadly as possible.

Table 2: (New Supplementary Table 5) Prompt optimization for reasoning tasks. With **TEXTGRAD**, we optimize a system prompt or the in-context examples for gpt-3.5-turbo using gpt-4o as the gradient engine that provides the feedback during backpropagation.

| Dataset                  | Method                                               | Accuracy (%) |
|--------------------------|------------------------------------------------------|--------------|
| Object Counting [20, 21] | CoT (0-shot) [10, 11]                                | 77.8         |
|                          | OPRO [5] (instruction-only, 0 demonstrations)        | 82.8         |
|                          | DSPy (BFSR, 8 demonstrations) [22]                   | 84.9         |
|                          | <b>TEXTGRAD</b> (3 demonstrations)                   | <b>91.9</b>  |
|                          | <b>TEXTGRAD</b> (instruction-only, 0 demonstrations) | <b>91.9</b>  |
|                          | <b>TEXTGRAD</b> (instruction and 3 demonstrations)   | <b>91.9</b>  |
| Word Sorting [20, 21]    | CoT (0-shot) [10, 11]                                | 76.7         |
|                          | OPRO [5] (instruction-only, 0 demonstrations)        | 77.8         |
|                          | DSPy (BFSR, 8 demonstrations) [22]                   | 79.8         |
|                          | <b>TEXTGRAD</b> (3 demonstrations)                   | 78.8         |
|                          | <b>TEXTGRAD</b> (instruction-only, 0 demonstrations) | 79.8         |
|                          | <b>TEXTGRAD</b> (instruction and 3 demonstrations)   | <b>80.8</b>  |
| GSM8k [23]               | CoT (0-shot) [10, 11]                                | 72.9         |
|                          | OPRO [5] (instruction-only, 0 demonstrations)        | 76.8         |
|                          | DSPy (BFSR, 8 demonstrations) [22]                   | 81.1         |
|                          | <b>TEXTGRAD</b> (3 demonstrations)                   | 80.1         |
|                          | <b>TEXTGRAD</b> (instruction-only, 0 demonstrations) | 81.1         |
|                          | <b>TEXTGRAD</b> (instruction and 3 demonstrations)   | <b>82.4</b>  |

#### Comment #1.4

*The reliance on LLMs, which are typically large and require significant computational resources, might limit the accessibility of TEXTGRAD in resource-constrained environments. Discussing potential mitigations or adaptations for more lightweight models could enhance the paper’s relevance.*

#### Response to #1.4

Thank you for raising this important point! We propose two distinct ways to test the accessibility of our framework. First, we test optimization using open-source models such as Llama [24] or Mixtral [25]. This informs us about whether our framework could be deployed using open-source models, and we show that it can. Next, we test the transferability and robustness of the prompts we optimized for closed-source models to other open-source models. We show a great deal of robustness for the transferability of **TEXTGRAD** optimized prompts, suggesting that users can pay a fixed cost to optimize prompts using closed-source models, to later be used by smaller models.

## Using open-source models with **TEXTGRAD**

We test prompt optimization using Llama-3.1-70B Instruct, which is a recently released open-source model. We use Llama-3.1-70B as an optimizer, and we test optimizing prompts for three different models: Llama-3.2-11B, Mixtral 8x22B, and Llama-3.1-70B itself. This is to demonstrate the generality of our approach.

In Table 3, we show the results for prompt optimization. In particular, we see that prompt optimization with **TEXTGRAD** consistently improves the performance of open-source models. We observe that the degree of improvement is larger for more capable models. As open-source LLMs rapidly progress, we expect **TEXTGRAD** to be even more useful to a broader audience.

Table 3: (New Supplementary Table 3) Prompt optimization for reasoning tasks using open-source models. We optimize system prompts and in-context demonstrations for various models using **TEXTGRAD** with Llama-3.1-70B Instruct as the gradient engine that provides feedback during backpropagation and the optimizer.

| <b>Dataset</b>                            | <b>Method</b>                                      | <b>Accuracy (%)</b> |
|-------------------------------------------|----------------------------------------------------|---------------------|
| <b>Llama 3.2 11B Vision Instruct [24]</b> |                                                    |                     |
| Object Counting [20, 21]                  | CoT (0-shot) [10, 11]                              | 82.8                |
|                                           | <b>TEXTGRAD</b> (instruction and 3 demonstrations) | <b>89.4</b>         |
| Word Sorting [20, 21]                     | CoT (0-shot) [10, 11]                              | 63.6                |
|                                           | <b>TEXTGRAD</b> (instruction and 3 demonstrations) | <b>64.6</b>         |
| GSM8k [23]                                | CoT (0-shot) [10, 11]                              | <b>86.1</b>         |
|                                           | <b>TEXTGRAD</b> (instruction and 3 demonstrations) | <b>86.1</b>         |
| <b>Mixtral 8x22B Instruct [25]</b>        |                                                    |                     |
| Object Counting [20, 21]                  | CoT (0-shot) [10, 11]                              | 87.9                |
|                                           | <b>TEXTGRAD</b> (instruction and 3 demonstrations) | <b>89.8</b>         |
| Word Sorting [20, 21]                     | CoT (0-shot) [10, 11]                              | 71.7                |
|                                           | <b>TEXTGRAD</b> (instruction and 3 demonstrations) | <b>78.3</b>         |
| GSM8k [23]                                | CoT (0-shot) [10, 11]                              | <b>91.3</b>         |
|                                           | <b>TEXTGRAD</b> (instruction and 3 demonstrations) | <b>91.3</b>         |
| <b>Llama 3.1 70B Instruct [24]</b>        |                                                    |                     |
| Object Counting [20, 21]                  | CoT (0-shot) [10, 11]                              | 85.9                |
|                                           | <b>TEXTGRAD</b> (instruction and 3 demonstrations) | <b>93.1</b>         |
| Word Sorting [20, 21]                     | CoT (0-shot) [10, 11]                              | 87.9                |
|                                           | <b>TEXTGRAD</b> (instruction and 3 demonstrations) | <b>95.8</b>         |
| GSM8k [23]                                | CoT (0-shot) [10, 11]                              | 94.8                |
|                                           | <b>TEXTGRAD</b> (instruction and 3 demonstrations) | <b>95.1</b>         |

## **Transferring prompts learned using closed-source models to open-source models**

Next, we test the transferability of prompts optimized for gpt-3.5-turbo-0125. In particular, we take the prompts optimized for gpt-3.5-turbo-0125, and use them as prompts to various other open-source models.

In Table 4, we present the results. Overall, prompts optimized for closed-source models using closed-source models, transfer well to open-source models. In particular, we observe a greater degree of transferability for more capable models, and again we expect this degree of improvement to only get better as open-source models are becoming more capable.

Table 4: (New Supplementary Table 3) Transferability of optimized prompts. We test the transferability of prompts optimized for gpt-3.5-turbo using **TEXTGRAD** on different models. Overall, optimized prompts improve the performance of various open-source and closed models, with up to 10% absolute performance gains.

| <b>Model</b>             | <b>Dataset</b>  | <b>Method</b> | <b>Accuracy</b> |
|--------------------------|-----------------|---------------|-----------------|
| Gemini 1.5 - Flash       | GSM8K           | CoT           | 91.1            |
|                          |                 | TG - Transfer | <b>92.3</b>     |
|                          | Object Counting | CoT           | <b>86.9</b>     |
|                          |                 | TG - Transfer | 81.8            |
|                          | Word Sorting    | CoT           | 59.6            |
|                          |                 | TG - Transfer | <b>70.7</b>     |
| Qwen 2.5 - 14B Instruct  | GSM8K           | CoT           | 94.6            |
|                          |                 | TG - Transfer | <b>94.9</b>     |
|                          | Object Counting | CoT           | 85.9            |
|                          |                 | TG - Transfer | <b>86.9</b>     |
|                          | Word Sorting    | CoT           | 32.3            |
|                          |                 | TG - Transfer | <b>33.3</b>     |
| Qwen 2.5 - 32B Instruct  | GSM8K           | CoT           | 95.6            |
|                          |                 | TG - Transfer | <b>96.6</b>     |
|                          | Object Counting | CoT           | 83.8            |
|                          |                 | TG - Transfer | <b>92.9</b>     |
|                          | Word Sorting    | CoT           | 47.5            |
|                          |                 | TG - Transfer | <b>55.6</b>     |
| Llama 3.1 - 8B Instruct  | GSM8K           | CoT           | <b>86.8</b>     |
|                          |                 | TG - Transfer | <b>86.8</b>     |
|                          | Object Counting | CoT           | 86.9            |
|                          |                 | TG - Transfer | <b>94.9</b>     |
|                          | Word Sorting    | CoT           | <b>46.5</b>     |
|                          |                 | TG - Transfer | 45.5            |
| Llama 3.1 - 70B Instruct | GSM8K           | CoT           | <b>94.4</b>     |
|                          |                 | TG - Transfer | 94.0            |
|                          | Object Counting | CoT           | 94.9            |
|                          |                 | TG - Transfer | <b>96.0</b>     |
|                          | Word Sorting    | CoT           | 73.7            |
|                          |                 | TG - Transfer | <b>84.8</b>     |

**Comment #1.5**

October 9, 2024

17 of 54

*In addition, the way the paper is written in many places may confuse readers or make them feel that the paper is over-claiming. For example, if I understand correctly, the components being optimized through TEXTGRAD in an AI system are textual contents like prompts. However, the paper may give the readers the impression that it introduces a new way to optimize a broader set of components, including model parameters, in an AI system. For instance, the abstract states, "Inspired by this, we introduce TEXTGRAD, a powerful framework performing automatic 'differentiation' via text. TEXTGRAD backpropagates textual feedback provided by LLMs to improve individual components of a compound AI system."*

### Response to #1.5

We apologize for this confusion, and thank you for this comment. It is correct that we are only optimizing textual content so far. First, we firmly believe this does not limit the applicability of **TEXTGRAD** in broader contexts. Natural language feedback is very expressive, as demonstrated in the various applications of our framework. Once future systems are better equipped at optimizing variables from other modalities (such as the multimodal outputs of gpt-4o are released to public), our framework will be applicable to other domains.

To reduce the confusion here, we clarified these aspects in the abstract, introduction and in Section 2. We hope this addresses the your concern. In particular, here is the revised version of the relevant parts of the text:

*Inspired by this, we introduce **TEXTGRAD**, a powerful framework performing automatic "differentiation" via text. **TEXTGRAD** backpropagates textual feedback provided by LLMs to improve individual components of a compound AI system by optimizing natural language text variables.*

*In this framework, each AI system is transformed into a computation graph, where variables are inputs and outputs of complex (not necessarily differentiable) function calls that are in the form of unstructured variables [...] Notably, the text feedback does not modify the parameters of the underlying neural network and is thus compatible with black-box APIs.*

*Example textual gradient computation. **TEXTGRAD** generates feedback to and optimizes unstructured variables in the form of text; it does not compute numerical gradients.*

### Comment #1.6

*My suggestion: While the paper "TEXTGRAD: Automatic 'Differentiation' via Text" introduces an innovative and potentially impactful framework, I do not think it meets the publication standard of Nature, unless my concerns can be well addressed. Specifically, the issues related to scalability, computational efficiency, comparative analysis, and contextual limitations need to be thoroughly examined and resolved to ensure the robustness and broader applicability of the framework.*

### Response to #1.6

We sincerely appreciate the your thoughtful comments and suggestions. We hope that our extensive new results and analyses thoroughly address the concerns raised regarding scalability, computational efficiency, comparative analysis, and contextual limitations. Specifically, we have:

1. Conducted a comprehensive cost-accuracy analysis (Figure 1) to demonstrate **TEXTGRAD**'s efficiency and scalability.
2. Expanded our comparative analysis by including new state-of-the-art baselines such as OPRO and Reflexion (Tables 2,1).
3. Demonstrated **TEXTGRAD**'s benefits on open-source models (Table 3) and validated the transferability of optimized prompts (Table 4) to address accessibility concerns.
4. Added new experiments in multimodal solution optimization (Table 1) and more complex systems (Figure 2) to showcase **TEXTGRAD**'s broad applicability.
5. Provided more detailed explanations and clarifications throughout the manuscript to address potential misunderstandings.

We believe these additions significantly strengthen our work and demonstrate the robustness and broader applicability of **TEXTGRAD**. We hope these new results and analyses meet the high standards expected for publication in Nature.

**Comment #1.7**

*The code is a good one.*

**Response to #1.7**

We thank you for your positive feedback on our code! We strive to maintain high standards of code quality and documentation to ensure reproducibility and facilitate further research in this area.

---

## 2 Response to Reviewer 2

### Comment #2.1

*Yuksekgonul et al. introduces TextGrad as a strategy for "performing automatic differentiation via text". Claims are made in the abstract as to improvements in question and answering (51% to 55%), LeetCode-Hard coding problem solutions, and the design of druglike small molecules and high-specificity oncology treatment plans. The premise of the work is that SOTA machine learning approaches may require expertise to design, but can be posed as a computation graph that is modified/improved by feedback from an LLM. TextGrad is introduced as a new framework with analogy to how PyTorch can be used for parameter optimization in traditional supervised learning. There are two 'modes' of optimization: optimization of instances and optimization of prompts. These capabilities are enabled through engineered prompts that explain the high-level goal, include mention of past iterations to provide momentum in SGD, etc. Overall, this work tries to demonstrate the generality and broad relevance of the framework to several tasks across several domains.*

### Response to #2.1

We sincerely thank you for your comprehensive summary of our work! We appreciate the recognition of **TEXTGRAD** as a framework that uses LLM optimized computation graphs as a general paradigm for parameter optimization. This generality is a key novel contribution of this work. Unlike existing and related works for LLM driven optimization such as Reflexion or OPRO which require separate implementations for different tasks, **TEXTGRAD** users do not need to modify the framework to apply it to their problem domain. The same code and framework can be applied from code optimization to science reasoning, from prompt engineering to biomedical applications.

### Comment #2.2

*The manuscript is written clearly, formatted nicely, and is bold in its claims. However, I believe that with the exception of the standard benchmarks used to evaluate LLMs and prompt engineering (the first two demonstrations), the demonstrations on molecular design and radiotherapy treatment are shallow and do not show evidence of performance that exceeds what is already capable with other, sometimes task-specific and sometimes non-ML approaches. It is already clear that one \*can\* use LLMs to approach these tasks, but it is not clear that one \*should\* and this paper does not illustrate their value adequately, especially in the case of drug discovery.*

### Response to #2.2

We are pleased to hear that our manuscript was clearly written and that our evaluations on standard benchmarks were acknowledged as significant. For our demonstrations of **TEXTGRAD**'s applications to molecule and radiotherapy treatment, we **do not** intend to recommend **TEXTGRAD** as a replacement for existing, domain-expert designed approaches. We apologize if our manuscript was not clear in this regard, and we have made corrections to ensure that this is not misinterpreted. In particular, for molecular optimization we have highlighted our reliance on a small subset of *in silico* oracles, and cautioned the practical value of our generated molecules in the absence of experimental validation.

**TEXTGRAD** is designed to work *with* domain specific methods. This is a major motivation and benefit of **TEXTGRAD**'s treatment of optimization using a generalized computation graph, along with our PyTorch-like syntax. **TEXTGRAD** enables users to easily integrate existing or newly developed domain specific tools that are suited for their problem of interest. In our applications, we integrate **TEXTGRAD** with tools from computational chemistry and radiotherapy, using the outputs of docking simulations, molecular property evaluation scores, and physical simulators to optimize molecular structures and treatment plans.

### Comment #2.3

*Even setting aside that this is a toy task, no comparisons are made to show SOTA or near-SOTA performance. My comments below around the drug discovery example might appear harsh, but I believe it is irresponsible to have overstated claims about the role of AI, let alone LLMs, for such tasks that are significantly more nuanced than what is discussed or evaluated. I cannot be supportive of publication based on this section, and hope that other experts for the other demonstrations can speak to the significance of those as well.*

### Response to #2.3

We sincerely appreciate your thoughtful comments and suggestions. We have introduced extensive new results and analysis to address specific concerns related to the domain specific applications of drug discovery and radiotherapy. Specifically, we have:

- Conducted experiments demonstrating improved performance of **TEXTGRAD** compared to existing molecule optimization algorithms.
- Conducted a sensitivity analysis study to analyze the specificity and value of **TEXTGRAD** gradients.
- Explored the steerability of **TEXTGRAD**'s optimization in multi-objective molecular property optimization.
- Conducted experiments demonstrating the benefit of **TEXTGRAD** compared to a Bayesian optimization approach for treatment planning.

In addition, throughout the text, we have provided additional commentary to better contextualize both the general contributions of **TEXTGRAD** and its applications and limitations to specific domains like molecule optimization and radiotherapy treatment planning.

Based on your feedback, we have also removed the molecule optimization section from the main paper. It is now presented in the Appendix as a supplementary, proof-of-concept case study. This better reflects the primary contribution of **TEXTGRAD** as a versatile optimization framework that achieves state-of-the-art results across diverse applications like coding, visual reasoning, question-answering, etc.

To this end, we have also added new applications of **TEXTGRAD**. We explore optimizing even more complex systems including systems that use tools and retrieval, and also optimizing prompts and instances simultaneously to reduce hallucinations in compound

systems (Figure 2). We optimize the solution chains from Chameleon [1], an agentic system that uses tools to solve visual question answering problems, and improve its performance from 77.5% to 85.2%. Further, in the hallucination benchmark HQH [2], we reduce the failure rate of gpt-4o from 30% to 21%.

We have also added applications of **TEXTGRAD** to multimodal domains such as scientific question answering in vision-language domains (Table 1), where **TEXTGRAD** improves the accuracy for visual question answering from 63.8% to 66.1% on ScienceQA [3], and from 91.6% to 92.4% on MathVista [4].

#### **Comment #2.4**

*The description of the strategy relies on pseudo-mathematical statements about partial derivatives that are not true derivatives, i.e., empirical applications of chain rule-like reasoning. Conceptually, what is reported bears a strong resemblance to ideas in self-refinement and other cited works like Reflexion. The relationship to other prompt optimization frameworks is discussed in the Related Work, including ProTeGi, which uses the exact same textual gradient analogy. This work packages these ideas into a convenient framework that may not represent an entirely novel conceptual advance, but is presented in an intuitive manner.*

#### **Response to #2.4**

We truly appreciate you pointing out these details of our work! Our method is related to works such as Reflexion or ProTeGi; however, we do think that **TEXTGRAD** provides multiple conceptual advancements. First of all, none of the previous works provide a method that can work with compound systems. Our backpropagation algorithm is uniquely targeting compound computation graphs.

Secondly, through all of our benchmarks in prompt optimization (Table 2), multimodal solution optimization (Table 1) or reducing hallucinations (Figure 2), we observe that **TEXTGRAD** consistently performs better than Reflexion, OPRO, and DSPy.

In addition to better performance, we would also like to highlight the generality of our framework. The codebase for Reflexion and OPRO have a separate implementation and/or prompting strategy for every task. For each task, there are separate implementations and prompts, making it nontrivial for users to apply these approaches to a new task.

With **TEXTGRAD**, a user does not need to modify the framework to apply to a new domain. It works out of the box for all the domains we tested without any modifications, and we spent a great deal of work to make sure optimization works as broadly as possible.

#### **Comment #2.5**

*Modest improvements in performance are observed for a few different tasks (e.g., Table 2); I am not the best reviewer to judge the significance of these differences. Table 3 shows performance matching DSPy on two tasks and moderately better performance on a third.*

#### **Response to #2.5**

We thank you for recognizing the earlier improvements. We further note the recognition of significance from the other two reviewers for the existing results.

We added a new set of results including baselines we added with i) performance gains including Figure 2,1,4, ii) robustness in Table 4, and iii) cost-accuracy in Figure 1.

For instance, first, we find that **TEXTGRAD** can optimize more complex compound AI systems such as Chameleon [1], an agentic system, or that we can optimize in-context examples to exceed the performance of baselines such as DSPy across all of the tasks (Table 2). Second, **TEXTGRAD**-optimized prompts are transferable to open-source models, and we can use open-source models as optimizers in **TEXTGRAD** (3). Third, optimization with **TEXTGRAD** is often inexpensive, costing  $< \$1$  for prompt optimization and  $< \$0.1$  for solution optimization experiments.

Overall, we hope the robustness, feasibility, and performance benefits are better highlighted in these new set of experiments.

#### **Comment #2.6**

*It is not clear how some of the hyperparameters for the model are chosen, e.g., how many iterations to use for refinement (sometimes 3, sometimes 12). Does performance deteriorate if too many iterations are used?*

#### **Response to #2.6**

Thank you for this question! We use optimization trajectories to determine how many iterations to perform. For instance, below are the learning curves for Prompt Optimization:

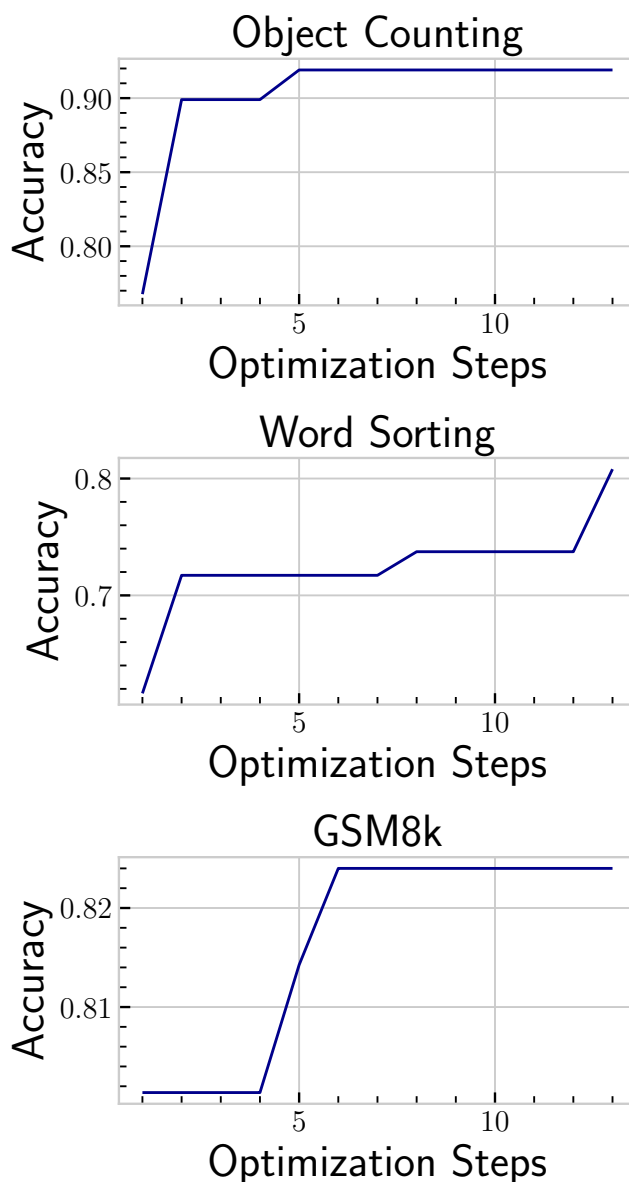


Figure 3: Learning curves for prompt optimization. Here, x-axis is the number of optimization iterations, and y-axis is the accuracy over the entire test set. We observe that optimization with **TEXTGRAD** is stable and the performance does not deteriorate over time.

In the case of prompt optimization, since we did not observe performance improvement in the validation set beyond 12 iterations, we capped it at 12.

#### Comment #2.7

*To comment on the chemistry and drug discovery related tasks from the perspective of a domain expert, there is a lot I am concerned about. The significance of a Vina score and QED value are overstated. QED is an extraordinarily simple and uninformative metric that has been overused in computational benchmarks, but is trivial to maximize with a score of 0.948 using almost any molecular generative model.*

## Response to #2.7

We appreciate the your perspective and detailed analysis. We outline our choice and treatment of the QED and Vina metrics as follows.

**Selection:** We agree with you that binding affinity and druglikeness as measured by Vina and QED respectively do not accurately reflect all the considerations and challenges in practical drug discovery campaigns. However, as you acknowledged, both the Vina and QED scores have been included as objective functions in numerous molecular optimization benchmarks [7, 19, 26]. Since the main contribution of TextGrad is as a general purpose framework for LLM-driven optimization, we elected to provide a proof-of-concept with standard objectives and datasets whenever possible.

For the molecule optimization task, we adopted both the objective function and benchmark dataset from the DOCKSTRING molecule design benchmark [19]. In our manuscript, we acknowledged that these metrics are not comprehensive, and have provided additional text to ensure that readers are aware of the limitations of our evaluation (in bold):

*In drug discovery, researchers seek to discover or design molecules that maximize a variety of objectives with regards to synthesizability, efficacy, and safety. [27, 28]. To demonstrate TEXTGRAD’s applications to multi-objective optimization, we apply TEXTGRAD to drug molecule optimization, and show how our framework can interface with computational tools and optimize chemical structures towards simultaneously improving their binding affinity and druglikeness as measured by in-silico oracles. . . . Though there are many more considerations for successful molecules, in our experiments, we restrict our objectives to minimizing the Vina score and maximizing the QED.*

**Difficulty:** We agree that the QED score on its own is straightforward to optimize and indeed has a ceiling of 0.948 [7]. However, the task presented was not towards single objective QED optimization, but rather the multi-objective task of improving both the QED and Vina scores. As recent works have shown, designing molecules that dock well is challenging for generative models [19, 29]. Since increasing the Vina score is not guaranteed to simultaneously increase the QED score or vice versa, simultaneously maximizing both is a non-trivial problem.

We acknowledge that the inclusion of the QED score can cause difficulties in judging the performance of TEXTGRAD, and in a further section, we have compared TEXTGRAD with existing molecule optimization algorithms for Vina only optimization.

**Significance:** As mentioned above, we agree that binding affinity and druglikeness are not the only important properties in drug development and have included this context in our manuscript. We further evaluate the quality of the TEXTGRAD generated molecules using *in-silico* safety properties, namely predicted Clinical Toxicity and Mutagenesis using the ADMET-AI algorithm [30] (Figure 4). Since these criteria were not included in the DOCKSTRING benchmark, we did not include them in our optimization, but perform the analysis post-generation. In the excerpted text preceding and following this response, we acknowledge that there are still further criteria of interest, and note that both selecting and designing informative metrics for successful drug discovery is an area

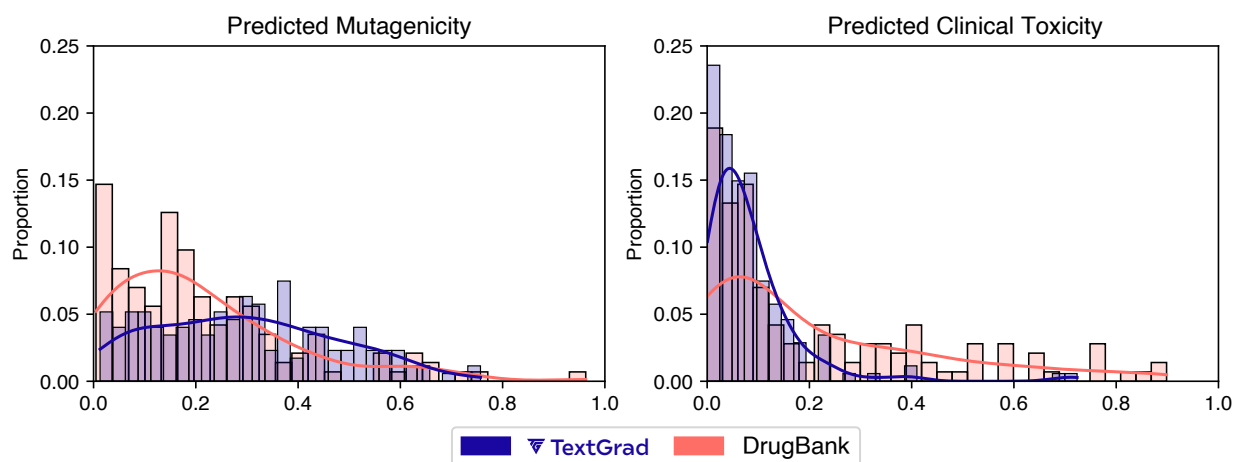


Figure 4: **(New Supplementary Figure 4) Safety Profiles** We evaluate the predicted harmfulness of **TEXTGRAD** molecules using the ADMET-AI model and compare them to clinically approved drugs. For both mutagenicity and clinical toxicity, 1.0 indicates a highly likelihood for harm and 0.0 a low likelihood. We observe that **TEXTGRAD** proposes molecules a similar predicted distributions to clinical counterparts, despite neither metric being directly optimized.

of ongoing research.

#### Comment #2.8

*QED does not inherently favor molecules with "better absorption properties" and docking scores don't inherently prefer "many functional groups".*

#### Response to #2.8

We thank you for your correction and apologize for our imprecise characterization of these metrics. Our intention was to indicate that these two objectives are distinct, and to provide intuition that optimizing the Vina Score does not automatically imply an improved QED score, and vice versa. We have removed the inaccurate text, and the corrected text is recorded below.

*$\Delta G$  can be estimated using "docking" simulations of protein-ligand binding [31, 32]. In our experiments, we employ the Vina score from the Autodock Vina tool, a widely used physics-based docking simulator [33]. A more negative Vina score indicates that the drug is predicted to be more likely to bind to its intended target.*

...

*One metric for "druglikeness" is the Quantitative Estimate of Druglikeness (QED) score, a weighted composite score of chemical characteristics such as molecular weight, lipophilicity, polar surface area, among others.*

...

*Though there are many more considerations for successful molecules, in our experiments, we restrict our objectives to minimizing the Vina score and maximizing the QED. Since each score evaluates different considerations in molecular design, optimizing one does not necessarily imply improvement in the other. Thus, optimizing molecules that simultaneously improve both objectives*

*is non-trivial.*

#### Comment #2.9

*Comparing a Vina score of a generated molecule to a clinically approved molecule is not evidence that a better molecule has been found.*

#### Response to #2.9

We thank you for this important clarification. We agree with you, and we do not claim that the molecules that **TEXTGRAD** has optimized are better than clinically approved molecules. We apologize if this was unclear, and we have included additional text to highlight the limitations of optimizing *in silico* objective functions. We include the relevant text below and highlight the addition in bold.

*For all 58 targets, **TEXTGRAD** consistently generates molecules with improved binding affinity and druglikeness irrespective of the initial starting fragment (Appendix F.4). For the 29 protein targets with clinically approved drugs, we observe that **TEXTGRAD** generates molecules with highly competitive affinity and druglikeness when compared to clinical molecules evaluated using the same objective function (Figure 2 (b)). ... we caution that the practical value of the molecules **TEXTGRAD** has generated are limited. In particular, we optimize over and evaluate only a few properties, all evaluated using *in silico* objective functions (e.g. Vina, QED, ADMET scores), which are not reliable substitutes for experimental and clinical evaluation.*

#### Comment #2.10

*The trajectory does show steady improvement from the starting molecule, but the seed molecules are artificially poor binders. If the optimization is continued for more iterations, are even better molecules identified?*

#### Response to #2.10

We thank you for your insightful question. The trajectory does improve, but we encounter diminishing returns. In order to provide the best gain for the least optimization cost (both **TEXTGRAD** calls and docking simulations), and to allow us to explore more combinations of targets and initial molecules, we employ early stopping. Briefly, we score each molecule produced using a scoring function, and if the mean of the top- $K = 5$  scores does not improve by more than  $TOL = 0.1$  for more than  $patience = 5$  iterations, the run is terminated, or until the `max_epoch` is reached. Since our objective is both Vina and QED optimization, we use the summarizing function described in Appendix F3 to generate a single scalar value, reproduced here for convenience:

$$s_{\text{overall}}(\text{molecule}, \text{protein}) = \text{Vina}(\text{molecule}, \text{protein}) + \frac{10}{\text{vina\_qed\_ratio}}(1 - \text{QED}(\text{molecule})) \quad (1)$$

In our original experiments, we set `max_epoch = 10`, where the low budget allowed us to explore a wide variety of targets ( $n = 58$ ) and initial molecules ( $n = 3$ ) for a to-

tal of 174 unique trajectories or 1740 molecules generated. In an additional set of experiments outlined below where we compared **TEXTGRAD** to existing molecule optimization algorithms, we increased `max_epoch` = 50, and used the scoring function described in the further section (e.g. Vina only). The scoring function and all parameters `K`, `TOL`, `patience`, `max_epoch` are all tunable. In general, a higher value of all the parameters (except `TOL`) will allow for longer explorations, and we invite users to select these parameters to suit their optimization budgets.

#### Comment #2.11

*The claim that TextGrad produces competitive results without a prior training set is also highly flawed, as the base LLM has been trained on data including knowledge of molecular structures, approved drugs, and various other aspects of molecular design. This goes beyond memorization of approved drugs for specific protein targets and is not mitigated by an evaluation of Tanimoto similarity to ChEMBL molecules.*

#### Response to #2.11

We agree with you and apologize for the confusing text. We believe you are referring to the following sentence in our original draft:

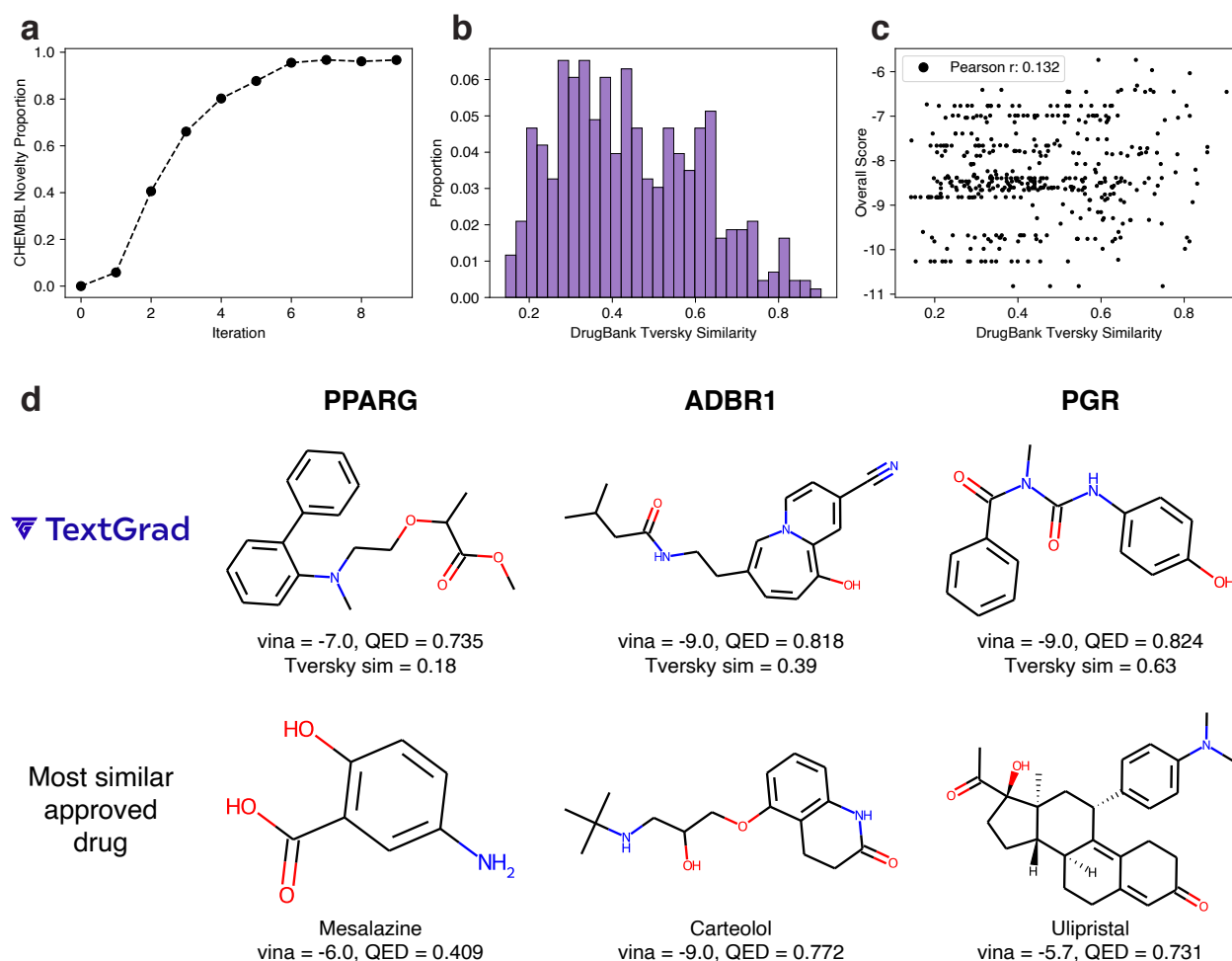
*... by combining traditional chemoinformatics tools with the general knowledge and reasoning capabilities of LLMs, **TEXTGRAD** produces competitive results even without a prior training set.*

We mistakenly failed to include the preceding text “finetuning” before “training set” and apologize for our error.

Our intention was to highlight that from a user’s perspective, applying **TEXTGRAD** to a problem domain has significantly less requirements than other LLM or traditional approaches. In the instance optimization setting, **TEXTGRAD** does not require the user to provide additional datasets for supervised finetuning or in-context-learning like other LLM approaches. For applications to molecular optimization, an additional compound library may be helpful, but is not necessary whether for initialization or supervision. **TEXTGRAD** only requires that the user provide an evaluation function and a starting molecule, which as we have shown, can be arbitrarily selected and does not need to be a high performing initial guess. We apologize that this was poorly and inaccurately communicated, and have corrected the text as follow.

*By combining specialized tools from computational chemistry with the general knowledge and reasoning capabilities of LLMs, **TEXTGRAD** enables molecule optimization from arbitrary starting molecules, without the need for additional fine-tuning datasets, or high-performing initializations from curated compound libraries.*

Regarding the memorization of approved drugs, we are aware that the base LLM selected by the user as the backend for **TEXTGRAD** is likely to have been trained on molecular structures, including approved drugs and information from compound libraries. However, we showed that the Tanimoto and Tversky similarity to ChEMBL and approved drugs respectively is low, and furthermore, the Pearson correlation between the Tversky similarity to approved drugs and the objective function **TEXTGRAD** optimizes is  $< 0.15$  (Figure 5). Together, these results suggest that while **TEXTGRAD** is designed



**Figure 5: (New Supplementary Figure 3) Structural Novelty** In panel (a), we observe increasing novelty over optimization updates, where a generated molecule is considered “novel” if there does not exist any existing molecule in ChEMBL with a Tanimoto similarity score greater than 0.8. In panels (b,c,d), we estimate substructure similarity using the Tversky metric between **TEXTGRAD** molecules and clinically approved drugs for each target. While there are a wide range of similarity scores, we observe that similarity with known drugs has little correlation with molecule performance, suggesting that **TEXTGRAD**’s generation process is weakly influenced by “memorized” knowledge of clinically approved structures.

to harness the capabilities of LLMs, the molecules it produces during the optimization trajectory are not simply extracted from memory.

#### Comment #2.12

*The explanations provided by the model that are claimed to be informative/interpretable are shallow and could be summarized in the most basic of introductory articles about medicinal chemistry "moves" and basic concepts in binding; moreover, they do not seem specific to the protein target of interest and instead are generic comments about what kind of interactions might be relevant for binding in general.*

#### Response to #2.12

Thank you for this perspective! The gradients that **TEXTGRAD** produces are conditioned on the loss function and taken with respect to the optimized variable, and thus contain information regarding both the objective and the variable to be optimized. Since the loss function in our experiments is a molecule's QED and its Vina score when docked against a particular protein (the full prompt is shown in Appendix F2), while the variable to be optimized is the molecule's SMILES string, we expect that the content of **TEXTGRAD**'s gradient to be largely composed of comments related to binding and medicinal chemistry. We acknowledge that the interpretability and depth of these gradients are subjective, and may not be highly informative to domain experts. In particular, here is the revised version of the relevant parts of the text:

***TEXTGRAD**'s natural language gradients enable users to transparently analyze the optimization process and the impetus for the changes between each molecule in the trajectory.*

We also performed new experiments to quantify the information content and specificity of these gradients. Concretely, we compare two optimization trajectories.

- **Original:** In the original **TEXTGRAD** framework, an initial molecule is modified by computing the loss function against a particular protein and updating its structure based on the gradient computed with respect to the loss. This process is repeated for multiple iterations, with the all gradients computed against the same protein.
- **Random:** In the random framework, an initial molecule is modified by randomly selecting a gradient from a different molecule docked against a different protein. This process is repeated for the same number of iterations, but with random gradients selected at each iteration.

We evaluate both sequences of molecules using the loss function (QED and Vina Score) against the original protein. If the gradients produced by **TEXTGRAD** are non-target specific, we expect that the randomized trajectory to perform as well as the original trajectory.

We compute "original" trajectories for all 58 targets in the DOCKSTRING benchmark, each starting from 3 initial starting fragments (benzene, pentane, acetamide) for a total of 174 original trajectories. For each original trajectory, we generate 10 random trajectories using the same initial fragment and evaluated against the same target, for a total of 1740 random trajectories. Each trajectory, both random and original, was optimized for 10 iterations. While the optimization length was fixed to allow for a comprehensive comparison

across multiple targets and for multiple randomizations, we observed that the trends for both the random and original trajectories are readily apparent within 10 iterations.

We observe that random trajectory produces lower performing molecules relative to original trajectory across both QED and Vina metrics. Using a paired t-test across protein targets and initial molecules, we compute the difference between the QED of the final molecule generated in the original trajectory and the mean QED value of the final molecules across the 10 repetitions in the random trajectories. For a total of  $n = 174$  paired trajectory comparisons, we report that the random trajectories have statistically significant lower QED scores than the original trajectories with a p-value of less than  $1e-4$ . We repeat this comparison for the Vina score, and report similar results, with the random trajectories having significantly higher (worse) Vina scores ( $p < 1e-4$ ). We also observe that random trajectories have a higher incidence of invalid molecules, producing on aggregate 1.5x more invalid SMILES strings than the original trajectory. Using the same statistical procedure for invalidity as the Vina and QED comparison, we report that the significance is  $p < 1e-2$ .

#### Comment #2.13

*The evaluation of this molecular optimization task lacks any discussion of other approaches, whether based on generative design or based on virtual screening of known molecules from compound libraries.*

#### Response to #2.13

We thank you for your suggestion, and have added further comparisons to molecular optimization algorithms. Since **TEXTGRAD** is a general framework for optimization that only requires a loss function and an arbitrary initial molecule, we believe that virtual screening represents a complementary, orthogonal approach as it does not perform optimization and whose performance is dependent on the compound library provided by the user. We provide a comprehensive outline of our comparisons below.

**Algorithms:** We compare **TEXTGRAD** to 6 molecular optimization algorithms, described briefly below.

- **Graph MCTS:** Based on Monte-Carlo Tree Search (MCTS), Graph MCTS applies MCTS to an atom-level molecular graph [16].
- **REINVENT:** Based on Reinforcement Learning (RL). REINVENT uses a policy based RL approach to tune RNNs to generate SMILES strings [17].
- **REINVENT-Selfies:** Similar to REINVENT, but with SELFIES strings instead of SMILES strings [7].
- **GP BO:** Combines Bayesian Optimization with a Gaussian Process kernel to sample molecular structures [18].
- **Smiles GA:** A genetic algorithm (GA) that explores molecular space through mutations and crossovers. In Smiles GA, the action space is taken over SMILES strings [34].

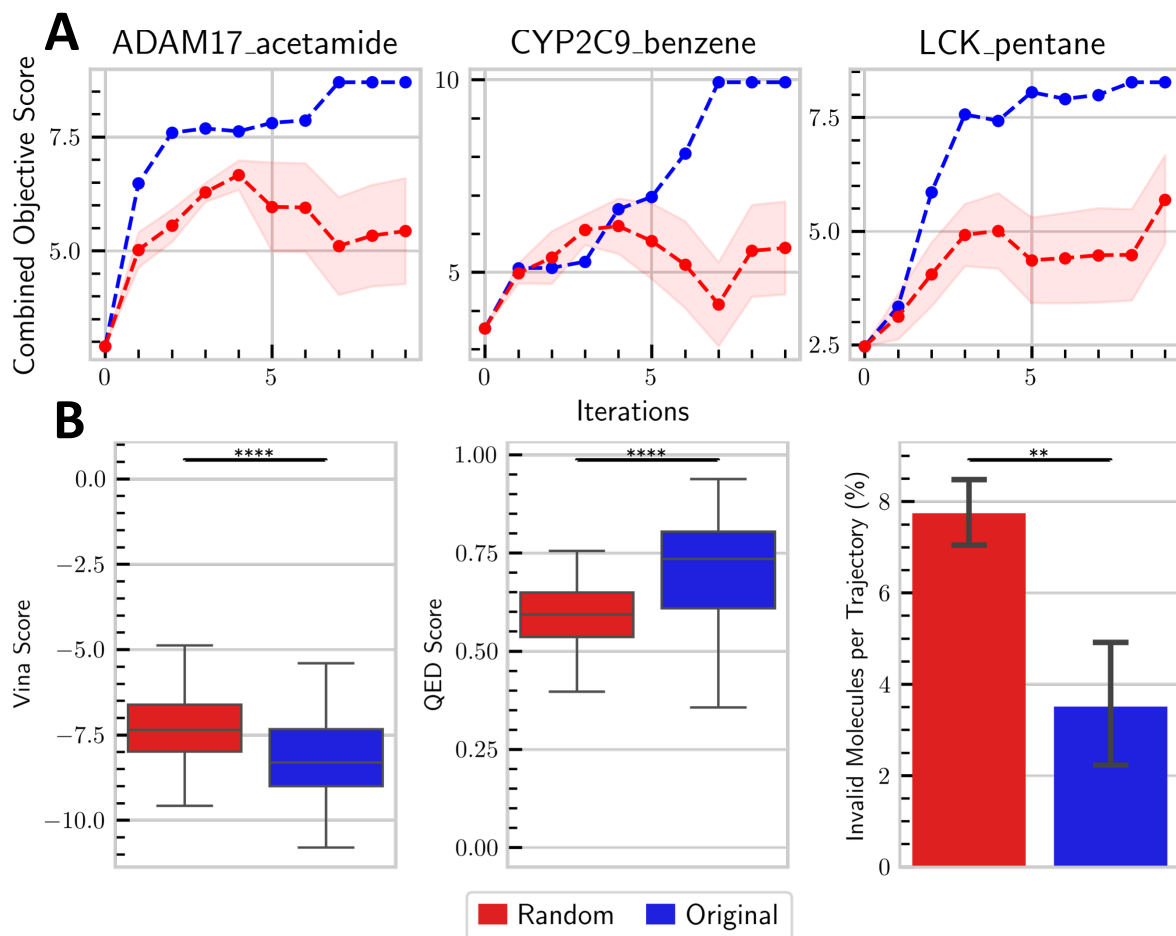


Figure 6: **(New Supplementary Figure 6) Gradient Sensitivity Analysis** **(A)** Sample performance trajectories under original and randomly selected gradients for different protein targets and initial molecules, using the combined objective score defined by Equation 1, with a `vina_qed_ratio` = 10, identical to the value in the loss prompt. **(B)** We observe that random trajectories underperform the original across Vina, QED, and percentage of valid molecules. Statistical significance computed using a paired t-test across protein-initial molecule pairs.

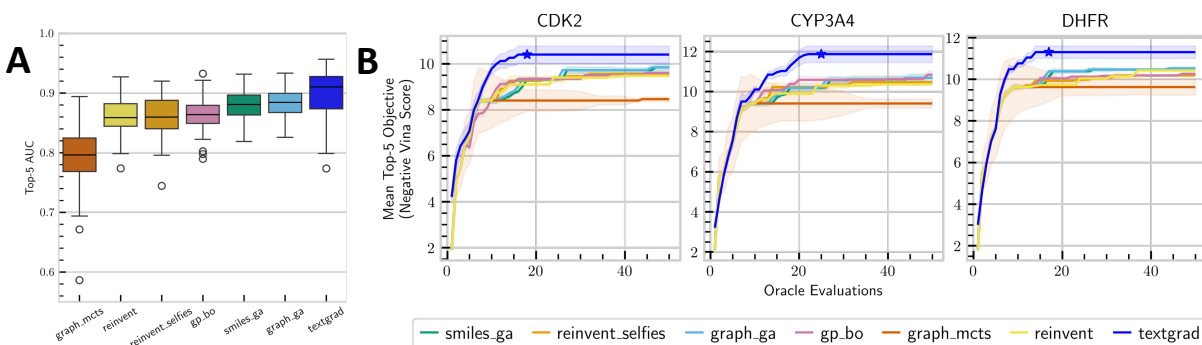
- **Graph GA:** Similar to Smiles GA, except that the action space is defined over molecular graphs [16].

Implementations were taken from the Practical Molecular Optimization (PMO) benchmark suite [7] without modification and with default hyperparameters, with the exception of the Graph MCTS algorithm which accepts an initial molecule (default CC). For Graph MCTS, we replaced the default initial molecule with the same molecule used by **TEXTGRAD** (pentane, acetamide, or benzene). For a full treatment of their implementation and algorithmic details, we refer the reviewer to the PMO benchmark and their original publications.

**Objective:** Incorporating your earlier feedback, for this comparison, we modified our original multi-objective of optimizing both the Vina and QED scores to only the Vina score. For the reasons outlined earlier, we still believe that our original task of multi-objective Vina/QED optimization is relevant and non-trivial. However since the benchmark algorithms above only accept single-value objective functions, combining the Vina and QED metrics would necessitate including a weighted QED term. Given your earlier concerns, we did not want the QED term to influence our comparison of **TEXTGRAD** against other models. For implementation reasons, the objective score was converted to a negative Vina Score (e.g. better binding  $\propto$  positive score) as the PMO benchmark adopts a maximization objective.

**Evaluation Metrics:** To compare the performances of different molecular optimization algorithms, we use the scaled AUC top- $K$  metric introduced by the Practical Molecular Optimization benchmark [7], with  $K = 5$ . Briefly, the top- $K$  trajectory measures the mean performance of the top  $K$  molecules generated at for a given oracle budget (in our setting, number of docking simulations). This is preferred over a single “best” ( $K = 1$ ) molecule as it rewards algorithms that generate multiple viable candidates. The AUC of the top- $K$  trajectory quantifies both the performance of the molecules generated as well the sample efficiency of the algorithm. In our setting, the objective function is the negative docking score (e.g. maximization) which is an unbounded objective that is variable between proteins. In order to compare algorithm performance across multiple proteins, we scale each trajectory to a  $[0, 1]$  range by empirically deriving maximal and minimal AUCs for each protein across the set of algorithms compared. Specifically, for a given protein, we set the maximum AUC Top- $K$  to be the AUC if the highest point in the top- $K$  trajectory across all algorithms was discovered at the first oracle call, while the minimum to be the AUC if the lowest point in the top- $k$  trajectory of all algorithms persisted for all oracle calls.

**Set-up:** We evaluated the 6 benchmark algorithms and **TEXTGRAD** over all 58 unique protein targets in the DOCKSTRING benchmark dataset. For each protein target and algorithm, we repeated the optimization run three times for different randomization seeds for REINVENT, REINVENT-Selfies, Graph GA, Smiles GA, and GP BO. In the case of **TEXTGRAD** and Graph MCTS, the runs were repeated for different initial molecules (benzene, pentane, acetamide). Each run was performed for a maximum oracle budget (e.g. number of docking simulations) of 50. For the benchmark algorithms, we ran all tra-



**Figure 7: (New Supplementary Figure 7) TEXTGRAD Comparisons against Baseline Methods (A)** Scaled mean top-5 AUCs for each method across all 58 protein targets. **(B).** Sample trajectories. For each algorithm, we selected the best performing trajectory to visualize, as measured by scaled Top-5 AUC, for the protein target listed, with the shaded error bars representing the standard error across the three repetitions (seeds/initial molecules). The blue star indicates the iteration at which **TEXTGRAD**’s early stopping condition was triggered. All objective scores afterwards are imputed by replacement with the last reached objective score.

jectories for the full 50 oracle calls. For **TEXTGRAD**, in order to limit the number of LLM calls, we employed the early stopping procedure outlined previously. Briefly, if the mean top- $K = 5$  objective did not improve by more than  $TOL = 0.1$  for more than  $patience = 5$  iterations, the run was terminated. To calculate the AUC of these early stopped runs, we simply extend the last mean top- $K$  objective score for the remaining oracle calls. Since the mean top- $K$  is a non-decreasing function, this prevents us from overestimating **TEXTGRAD**’s performance. In total, this experiment performed a comprehensive evaluation between the 7 algorithms across 58 proteins with 174 trajectories, and generating a total of 8700 molecules.

**Results:** We outline our results in Figure 7, where the performance across all 58 proteins is shown in panel A, and sample trajectories in panel B. We conducted a paired t-test between the AUC Top- $K$  of **TEXTGRAD** for each protein against all other methods. The difference between **TEXTGRAD** and all other methods was significant, with the greatest difference between **TEXTGRAD** and Graph MCTCS and the closest between **TEXTGRAD** and Graph GA.

While these results demonstrate the promising potential of **TEXTGRAD** to optimize molecules, we don’t want to claim that it is the best method for this particular task. We have framed all of these experiments as a proof-of-concept case study of molecule optimization in the Appendix, as it is not critical to the main message of **TEXTGRAD**.

#### Comment #2.14

*Section F.2 claims that the vina\_qed\_ratio can be tuned to change the tradeoff but no results are shown to indicate that the scalarization of the two objectives is quantitatively respected by the model.*

## Response to #2.14

We thank you for your suggestion and have included additional results showing the effect of the `vina_qed_ratio` parameter.

**Setup:** Using the prompt provided in Appendix F.3 (copied below for convenience), we set the `vina_qed_ratio` to 50, 25, 1 respectively. This sequence of values represents an increase in the prioritization of the QED score relative to the Vina. Intuitively, for the value of 50, we essentially instruct **TEXTGRAD** to disregard the QED score, while at 1, we inform **TEXTGRAD** that the QED and Vina objectives are equally important. Starting from pentane, we optimize molecules for 10 iterations against all 58 proteins in the DOCKSTRING benchmark, for a total of 58 unique trajectories and 580 molecules.

**Results:** We observe that tuning the `vina_qed_ratio` alters the distributions of the generated molecules (Figure 8). We observed that an increase in QED prioritization led to a corresponding increase QED values. Conversely, a decrease in Vina prioritization led to a decrease in Vina performance (e.g. higher Vina scores). Using a paired t-test, we observed that the changes in both Vina and QED were statistically significant between the parameter settings of 50 and 1.

### Molecule Optimization Prompt

Given a docking and a druglikeness score, and a molecule as a SMILES string provide a short criticism to improve the druglikeness of this molecule and its binding affinity with the protein {protein\_name}. For docking, lower is better (less than -10 is considered good) and for druglikeness, 1 is the best and 0 is the worst (greater than 0.8 is considered good). In terms of prioritization, the docking score is {vina\_qed\_ratio} times as important as the druglikeness score. Make sure your criticism is very succinct and to the point.

```
# if smiles_string is valid
SMILES: {smiles_string}, Docking: {Vina}, Druglikeness: {QED}
```

```
# if smiles_string is invalid
SMILES: {smiles_string}, This molecule is invalid.
```

## Comment #2.15

*I believe that defending claims of utility for drug design requires a stricter standard than claims of improving performance on standardized Q&A/LeetCode benchmarks.*

## Response to #2.15

Thank you for this important remark. We agree with your perspective and believe that drug development is a more nuanced topic with less standardized benchmarks than traditional LLM evaluations on question answering and code improvement. To this end, we have removed the molecule optimization section from the main paper and now present it as a supplementary case study in the Appendix. The main contribution of **TEXTGRAD** is

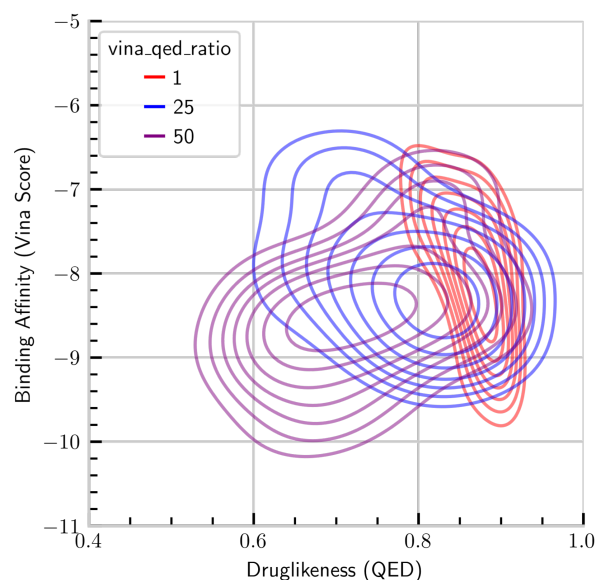


Figure 8: **(New Supplementary Figure 5) Vina/QED Steerability** Distributions of molecule properties under different prioritization instructions. We observe a statistically significant worsening (increase) in the Vina scores upon reducing the Vina importance, and a corresponding significant improvement in the QED scores.

a versatile framework to optimize diverse tasks that people use LLMs to do. In place of molecule optimization, we have added several additional tasks to demonstrate the value of **TEXTGRAD** across diverse settings. This includes showing that **TEXTGRAD** can optimize more complex compound AI systems such as Chameleon [1], an agentic system, **TEXTGRAD** can optimize in-context examples to exceed the performance of baselines such as DSPy across all of the tasks (Table 2), **TEXTGRAD** can reduce hallucinations in multimodal visual question answering, and we can use open-source models as optimizers in **TEXTGRAD** (3), among many other new results.

#### Comment #2.16

*For the application of radiotherapy treatment plan optimization, what is the performance of a task-specific optimizer that approaches the problem as a numerical optimization? Or any other nested optimization that directly tries to optimize the performance metrics reported in Figure 3de? Though I am not an expert in this application, I suspect there are plausible baselines that have been omitted. Details are not provided as to the origin of the "human-optimized" plans.*

#### Response to #2.16

Thank you for this important question! We would like to clarify the strength of our clinical "human-optimized" baseline and the reasons behind its use. These plans were developed by experienced Stanford dosimetrists and radiation oncologists using the Eclipse treatment planning system from Varian Medical Systems, which is a widely used software in the clinical workflow. The plans were manually created as Volumetric Modulated Arc Therapy (VMAT) plans and clinically approved for patient treatment, which qualifies

| Method                | Mean Dose (Gy)      | D <sub>95</sub> (Gy) |
|-----------------------|---------------------|----------------------|
| Clinical Goal         | 70.20               | 70.20                |
| Radiation Oncologist  | +1.97 (0.36)        | -0.10 (0.15)         |
| Bayesian Optimization | +4.84 (1.65)        | +0.00 (0.07)         |
| <b>TEXTGRAD</b>       | <b>+0.51 (0.09)</b> | <b>+0.00 (0.00)</b>  |

Table 5: (New Supplementary Table 1) Radiotherapy treatment plan optimization for prostate cancer patients. **TEXTGRAD** optimized plans have better dose metrics (mean dose and D<sub>95</sub>) for the planning target volume (PTV) than plans obtained with Bayesian Optimization.

them as standard clinical practices. We have added more details to the paper.

This human-optimized baseline represents a very strong comparison because it reflects real-world treatment planning, which involves multidisciplinary expertise from dosimetrists, radiation oncologists, and technicians. In creating these plans, clinicians manually balance multiple treatment quality metrics, such as target coverage, organ-at-risk sparing, and plan delivery efficiency. These metrics are weighted according to the experts’ clinical judgment, which is highly context-specific. Moreover, the complexity of these manual processes and the balancing of competing quality metrics make human-optimized plans a robust standard for comparison. We therefore believe that it is a strong demonstration that **TEXTGRAD** is able to get closer to the ideal clinical dosages while reducing organ exposure compared to these clinician-optimized treatment plans.

Regarding other optimization techniques, while there are task-specific optimization systems such as those mentioned in [35–37], their adoption in clinical practice remains limited due to the challenges of balancing multiple non-differentiable quality metrics. There is no unified objective score that consolidates all relevant metrics, and developing such a score would require explicit trade-offs between conflicting goals. This makes direct numerical optimization difficult in practice. Furthermore, many of these metrics are non-differentiable, limiting the application of gradient-based optimization methods in this setting.

To address your suggestion, we experimented with a task-specific optimization approach using Bayesian Optimization (BO), following the methodology of [8]. In this implementation, we created a score function as a weighted sum of multiple plan quality metrics and applied Optuna [38] to perform BO over this objective. The results of these experiments are detailed in Table 5. While BO was able to optimize the score function reasonably well, it didn’t perform as well as **TEXTGRAD**. Moreover BO converged much more slowly than **TEXTGRAD**, typically requiring an average of 21.3 iterations to achieve suboptimal plans. In contrast, **TEXTGRAD** generated high-quality plans within just 10 iterations. This highlights the efficiency and effectiveness of our approach compared to the baseline.

We have further demonstrated the applicability of **TEXTGRAD** to optimize radiation treatment plans for patients with a different type of cancer—head-and-neck cancer. In this setting, **TEXTGRAD** optimized plans achieved mean dosages that are closer to the clinical target than oncologist-optimized plans —average deviation of 2.3 in planning tar-

get volumes (PTVs) from the target dose for **TEXTGRAD** compared to average deviation of 3.4 for the oncologist plans.

---

### 3 Response to Reviewer 3

#### Comment #3.1

##### *A. Summary of the key results*

*The paper introduces TextGrad, a framework analogous to backpropagation and automatic differentiation, designed for optimizing text prompts or instances through backpropagated textual feedback. This general and effective approach is applied across diverse scenarios: optimizing code snippets, molecular structures, question-answering, and designing radiation oncology treatment plans. The results demonstrate that the proposed method significantly improves the performance of LLM agents. Additionally, the authors present a library that enables developers to easily apply the approach using programming syntax similar to PyTorch.*

#### Response to #3.1

We thank you for your comprehensive summary of our work. We appreciate your recognition of **TEXTGRAD** as a framework analogous to backpropagation and automatic differentiation, designed for optimizing text prompts or instances through backpropagated textual feedback. We are pleased that you have highlighted the broad applicability of our approach across diverse scenarios and acknowledged the significance of our PyTorch-like library for easy implementation.

#### Comment #3.2

##### *B. Originality and significance: if not novel, please include reference*

*The use of textual feedback as an optimizer is not new and has been discussed in the literature (see, for example, the paper list <https://github.com/AGI-Edgerunners/LLM-Optimizers-Papers>). To me, the method of using textual feedback as a gradient is similar to approaches found in [1] and [2], as cited in this paper. However, to the best of my knowledge, none of the prior works provides a general framework for implementing this idea, nor do they leverage PyTorch's programming syntax to allow developers to easily construct complex, domain-specific applications. The authors demonstrate that their library enables this approach to be applied across a range of diverse and knowledge-intensive fields. In the same way that deep learning has benefited from libraries that simplify implementation, making it accessible to various fields, the proposed framework simplifies the use of LLM as an optimizer. I can see the potential impact of this framework being significant. [1] Large Language Models as Optimizers [2] Automatic Prompt Optimization with "Gradient Descent" and Beam Search*

#### Response to #3.2

We truly appreciate your thoughtful assessment of the originality and significance of our work! We agree that the use of textual feedback as an optimizer has been explored in previous literature, and we acknowledge the similarities with approaches like those in OPRO and ProTeGi. We are pleased that you recognize the unique contributions of **TEXTGRAD**, particularly:

- Leveraging PyTorch-like programming syntax to allow developers to easily construct complex, domain-specific applications.

- Demonstrating the framework’s applicability across a range of diverse and knowledge-intensive fields.
- Providing a general framework for implementing textual feedback as an optimization method.

We agree with the your assessment that, much like how deep learning libraries have simplified implementation and made it accessible to various fields, **TEXTGRAD** simplifies the use of LLMs as optimizers. We are encouraged by your view on the potential significant impact of our framework.

Thank you for highlighting these key aspects that differentiate **TEXTGRAD** from prior work and underscore its potential to advance the field!

### **Comment #3.3**

*C. Data & methodology: validity of approach, quality of data, quality of presentation Overall, the proposed approach is well-motivated and clearly articulated. The authors validate the framework across a diverse range of applications, effectively demonstrating its utility. The experimental design is both solid and reasonable, and the results strongly support the authors’ claims. I have several suggestions and questions about the experiment settings, please see Section F.*

### **Response to #3.3**

We appreciate your positive assessment of our methodology. We’re pleased they found our approach well-motivated, clearly articulated, and effectively demonstrated across diverse applications. We’re grateful for your view that our experimental design is solid and our results strongly support our claims. Below we’ve addressed your additional suggestions about experiment settings in our responses to the specific comments.

### **Comment #3.4**

*D. Appropriate use of statistics and treatment of uncertainties Certain factors may introduce variance in the evaluation process. Additionally, some experimental results rely on models released by OpenAI, which may evolve or be removed over time. In such cases, future attempts to reproduce the results might not be possible. Therefore, it would be great to address these uncertainties and discuss their potential impact on reproducibility.*

### **Response to #3.4**

Thank you for this comment! To ensure the reproducibility of the results, we take three key steps:

#### **Details on the models used:**

For all the models we used, we refer to the specific endpoints. For instance, we report that we use the gpt-4o-2024-05-13 version of gpt-4o, and gpt-3.5-turbo-0125 version of gpt-3.5-turbo. These are dated snapshots, meaning that future or more up-to-date deployments of these models will not change these endpoints. Further, we ensure we run

our experiments with temperature 0 to make results as close to deterministic as possible. For our new experiments with open-source models, we are using public models so experiments should be easier to reproduce.

### **Releasing implementations of our experiments:**

In our open-source repository under the evaluation folder, we release scripts to reproduce our experiments.

### **Releasing optimization trajectories:**

Finally, in our repository, we release the optimization trajectories for various experiments. For instance, we release all steps of the optimization for solution optimization, the discovered prompts, and the discovered code snippet. This makes it easier and more transparent for the research community to understand the behavior of **TEXTGRAD**.

### **Comment #3.5**

*E. Conclusions: robustness, validity, reliability Overall, the experiment results demonstrate the effectiveness of the proposed framework. With testing the approach in various datasets and tasks, the authors also demonstrate the robustness of the approach. See more comments and discussion in Section F.*

### **Response to #3.5**

We thank you for recognizing the effectiveness and robustness of our framework across various datasets and tasks. We appreciate your acknowledgment of our efforts to demonstrate **TEXTGRAD**'s capabilities in real-world applications.

### **Comment #3.6**

*F. Suggested improvements: experiments, data for possible revision Please see comments/suggestions in the following: 1. I'm curious whether the proposed approach can be applied to optimizing few-shot examples in an in-context learning setting.*

### **Response to #3.6**

Thank you for this interesting question! To demonstrate the broad applicability of **TEXTGRAD**, we expanded our prompt optimization experiments to jointly optimize both in-context demonstrations and also the instruction.

In particular, optimizing in-context demonstration falls nicely under the **TEXTGRAD** abstractions. At optimization time, we backpropagate feedback to in-context demonstrations, and update them using Textual Gradient Descent. For instance, for Object Counting, the initialization and the optimized version of in-context demonstrations look like follows:

**Demonstrations at initialization:**

Input: I have 1 banana, 4 strawberries, 1 apple, 2 peaches, 1 plum, 1 blackberry, and 2 raspberries. How many fruits do I have? Answer: 12

Input: I have 1 blackberry, 1 peach, 1 nectarine, 1 plum, 1 raspberry, 1 orange, 1 strawberry, 1 banana, 2 apples, and 4 grapes. How many fruits do I have? Answer: 14

Input: I have 1 apple, 3 bananas, 1 strawberry, 1 peach, 3 oranges, 1 plum, 1 raspberry, 2 grapes, 1 nectarine, and 1 blackberry. How many fruits do I have? Answer: 15

**Demonstrations after TEXTGRAD optimization:**

Input: I have a banana, four strawberries, an apple, two peaches, a plum, a blackberry, and two raspberries. How many fruits do I have? Ground Truth Response: To find the total number of fruits, we add up the quantities of each type of fruit mentioned in the input. Banana: 1, Strawberries: 4, Apple: 1, Peaches: 2, Plum: 1, Blackberry: 1, Raspberries: 2. Adding these quantities together, we get  $1 + 4 + 1 + 2 + 1 + 1 + 2 = 12$ . Answer: 12

Input: I have a blackberry, a peach, a nectarine, a plum, a raspberry, an orange, a strawberry, a banana, two apples, and four grapes. How many fruits do I have? Ground Truth Response: To find the total number of fruits, we add up the quantities of each type of fruit mentioned in the input. Blackberry: 1, Peach: 1, Nectarine: 1, Plum: 1, Raspberry: 1, Orange: 1, Strawberry: 1, Banana: 1, Apples: 2, Grapes: 4. Adding these quantities together, we get  $1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 2 + 4 = 14$ . Answer: 14

Input: I have an apple, three bananas, a strawberry, a peach, three oranges, a plum, a raspberry, two grapes, a nectarine, and a blackberry. How many fruits do I have? Ground Truth Response: To find the total number of fruits, we add up the quantities of each type of fruit mentioned in the input. Apple: 1, Bananas: 3, Strawberry: 1, Peach: 1, Oranges: 3, Plum: 1, Raspberry: 1, Grapes: 2, Nectarine: 1, Blackberry: 1. Adding these quantities together, we get  $1 + 3 + 1 + 1 + 3 + 1 + 1 + 2 + 1 + 1 = 15$ . Answer: 15

Overall results are shown in Table 6. We observe that optimizing in-context examples are often helpful for optimization, and gives us improved performance for 2/3 prompt optimization tasks. We truly appreciate this suggestion! We added further details on this to the main paper.

Table 6: (New Supplementary Table 5) Prompt optimization for reasoning tasks. With **TEXTGRAD**, we optimize a system prompt or the in-context examples for gpt-3.5-turbo using gpt-4o as the gradient engine that provides the feedback during backpropagation.

| Dataset                  | Method                                               | Accuracy (%) |
|--------------------------|------------------------------------------------------|--------------|
| Object Counting [20, 21] | CoT (0-shot) [10, 11]                                | 77.8         |
|                          | OPRO [5] (instruction-only, 0 demonstrations)        | 82.8         |
|                          | DSPy (BFSR, 8 demonstrations) [22]                   | 84.9         |
|                          | <b>TEXTGRAD</b> (3 demonstrations)                   | <b>91.9</b>  |
|                          | <b>TEXTGRAD</b> (instruction-only, 0 demonstrations) | <b>91.9</b>  |
|                          | <b>TEXTGRAD</b> (instruction and 3 demonstrations)   | <b>91.9</b>  |
| Word Sorting [20, 21]    | CoT (0-shot) [10, 11]                                | 76.7         |
|                          | OPRO [5] (instruction-only, 0 demonstrations)        | 77.8         |
|                          | DSPy (BFSR, 8 demonstrations) [22]                   | 79.8         |
|                          | <b>TEXTGRAD</b> (3 demonstrations)                   | 78.8         |
|                          | <b>TEXTGRAD</b> (instruction-only, 0 demonstrations) | 79.8         |
|                          | <b>TEXTGRAD</b> (instruction and 3 demonstrations)   | <b>80.8</b>  |
| GSM8k [23]               | CoT (0-shot) [10, 11]                                | 72.9         |
|                          | OPRO [5] (instruction-only, 0 demonstrations)        | 76.8         |
|                          | DSPy (BFSR, 8 demonstrations) [22]                   | 81.1         |
|                          | <b>TEXTGRAD</b> (3 demonstrations)                   | 80.1         |
|                          | <b>TEXTGRAD</b> (instruction-only, 0 demonstrations) | 81.1         |
|                          | <b>TEXTGRAD</b> (instruction and 3 demonstrations)   | <b>82.4</b>  |

### Comment #3.7

*2. Please correct me if I'm wrong — while the proposed framework is general, the applications discussed in the paper don't seem to involve complex backpropagation chains and multiple variables, as shown in Figure 1(b). I acknowledge that the authors have demonstrated the framework's effectiveness in several real-world tasks, but it would be interesting to test it in more complex agent-based tasks, such as WebArena, which require more intricate actions and tool use.*

### Response to #3.7

Thank you for the insightful question. We added new experiments with more complex optimization problems. These new results are summarized in Fig. 9.

**Optimizing longer chains:** We study Chameleon [1], a sophisticated agent system involving multiple LLM modules using diverse tools. Chameleon previously reported state-of-the-art results across several reasoning-intensive tasks. It includes a module inventory of different tools and an LLM-based planner. Each module produces an output given the inputs from the previous models, and the final module generates the overall response. However, the lack of a feedback mechanism in the planning and tool execution processes can lead to error accumulation over time.

To address this, we use **TEXTGRAD** to optimize Chameleon by using textual gradients to adjust each module’s output at test time. The system first completes a standard forward pass, then iteratively optimizes each module and unrolls the forward pass, starting from the first and moving sequentially.

For evaluation, we use ScienceQA-IMG [3], a dataset requiring coordinated use of tools for visual perception, knowledge retrieval, and response generation. The module inventory includes tools for tasks like OCR (via EasyOCR [15]), image captioning, knowledge retrieval, solution generation, and answer generation. The results, as shown in Figure 2 Panel (e). **TEXTGRAD** achieves a 5.7% increase over the original Chameleon with two iterations of optimization. This improvement further extends to 7.7% with three iterations, indicating the benefits of multiple optimization iterations.

Overall, **TEXTGRAD** proves to be an effective method to significantly enhance the capabilities of compound systems like Chameleon in complex, multi-modal tasks that require coordinated use of diverse tools.

**Optimizing complex chains:** In our second compound AI system application, we apply **TEXTGRAD** to reduce hallucinations in multimodal models, focusing on the Visual Question Answering (VQA) task. In VQA, images are presented alongside questions related to the content of the images. To guide the model in answering the questions, instructional prompts are provided, such as "Answer the question based on the image, let’s think step by step.". These instructional prompts help direct the model’s reasoning process. Then another language model evaluate the solution based on a loss function prompt. Here we use **TEXTGRAD** to jointly optimize the two sets of prompts—instructional prompt for the answer generation LLM and the loss prompt for the evaluation LLM—as well as the generated solution itself. All of these can be treated as variables in **TEXTGRAD** and are optimized jointly through our text-based chain rule. We show that this approach lead to an excellent accuracy of 79% HGH VQA benchmark [2], a challenging task where multimodal language models often hallucinate, compared to an accuracy of 70% by gpt-4o with CoT.

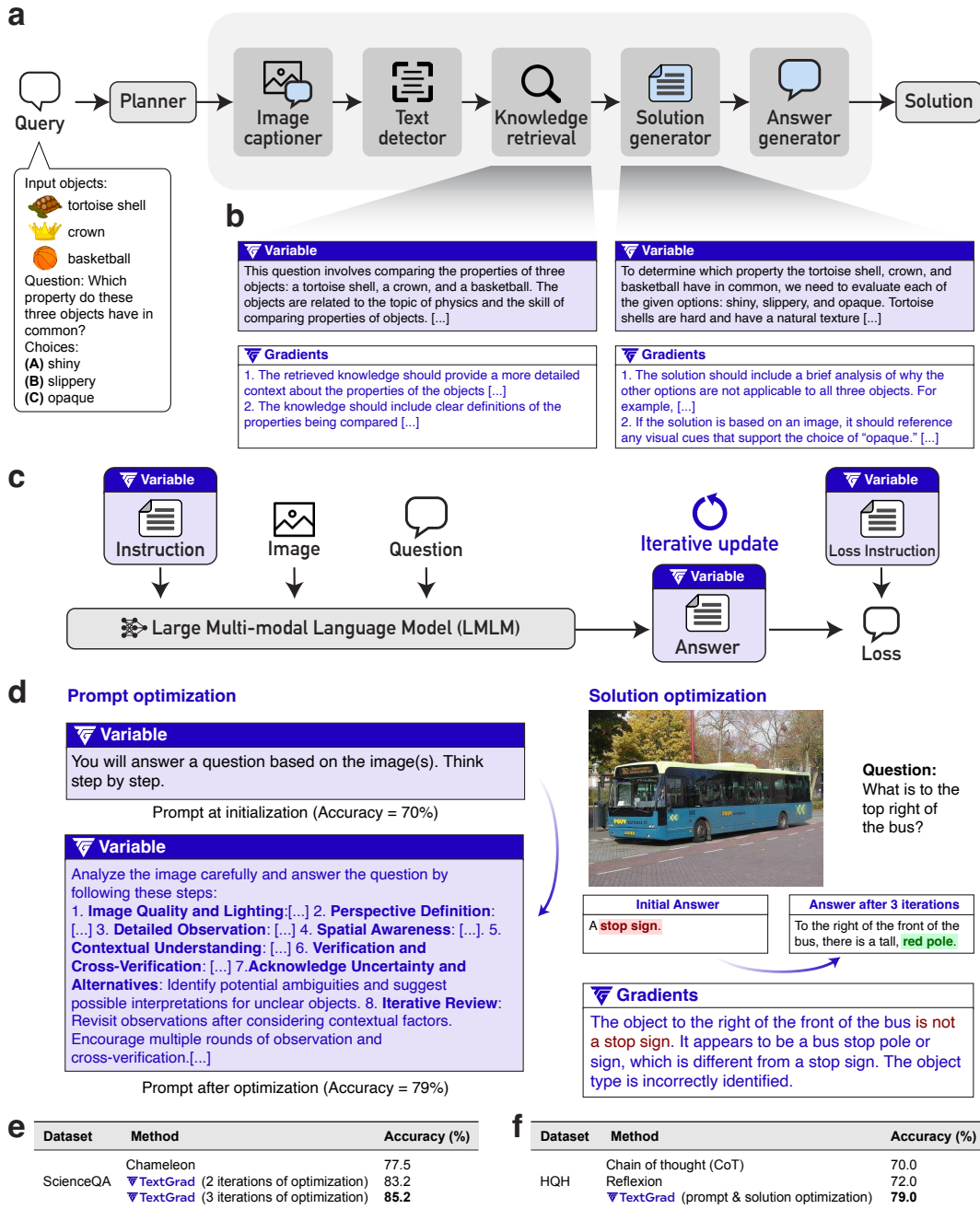


Figure 9: (New Figure 3 in the main paper) **Optimizing compound AI systems.** (a) **TEXTGRAD** enhances the solution generated by Chameleon, an agentic system that generates a plan followed by tool execution. We treat the output of each module as a separate variable to optimize and improve them. (b) Textual gradients are computed and backpropagated through the tool execution chain. (c) **TEXTGRAD** iteratively optimizes the answer to a multimodal question. To mitigate hallucinations of multimodal language models in visual question answering, we treat loss prompt, the instruction, and the solution as variables to jointly optimize and improve them. (d) With an example in HQH we show how **TEXTGRAD** jointly optimizes the prompts and solutions. **TEXTGRAD** yields accuracy improvements of 7.7% and 9% on ScienceQA (e) and HQH (f), respectively.

**Comment #3.8**

*3. I wonder how robust the prompts or instances identified by this framework are. Specifically, how does the performance hold up when applying prompts or instances obtained from one LLM (e.g., GPT-4) to another (e.g., Gemini)?*

**Response to #3.8**

Thank you for this important question! We test the transferability of prompts optimized for gpt-3.5-turbo-0125. In particular, we take the prompts optimized for gpt-3.5-turbo-0125, and use them as prompts to various other open-source models.

In Table 7, we present the results. Overall, prompts optimized for closed-source models using closed-source models transfer well to open-source models. We observe a greater degree of transferability for more capable models, and we expect this degree of improvement to only get better as open-source models become stronger.

Table 7: (New Supplementary Table 3) Transferability of optimized prompts. We test the transferability of prompts optimized for gpt-3.5-turbo using **TEXTGRAD** on different models. Overall, optimized prompts improve the performance of various open-source and closed models, with up to 10% absolute performance gains.

| <b>Model</b>             | <b>Dataset</b>  | <b>Method</b> | <b>Accuracy</b> |
|--------------------------|-----------------|---------------|-----------------|
| Gemini 1.5 - Flash       | GSM8K           | CoT           | 91.1            |
|                          |                 | TG - Transfer | <b>92.3</b>     |
|                          | Object Counting | CoT           | <b>86.9</b>     |
|                          |                 | TG - Transfer | 81.8            |
|                          | Word Sorting    | CoT           | 59.6            |
|                          |                 | TG - Transfer | <b>70.7</b>     |
| Qwen 2.5 - 14B Instruct  | GSM8K           | CoT           | 94.6            |
|                          |                 | TG - Transfer | <b>94.9</b>     |
|                          | Object Counting | CoT           | 85.9            |
|                          |                 | TG - Transfer | <b>86.9</b>     |
|                          | Word Sorting    | CoT           | 32.3            |
|                          |                 | TG - Transfer | <b>33.3</b>     |
| Qwen 2.5 - 32B Instruct  | GSM8K           | CoT           | 95.6            |
|                          |                 | TG - Transfer | <b>96.6</b>     |
|                          | Object Counting | CoT           | 83.8            |
|                          |                 | TG - Transfer | <b>92.9</b>     |
|                          | Word Sorting    | CoT           | 47.5            |
|                          |                 | TG - Transfer | <b>55.6</b>     |
| Llama 3.1 - 8B Instruct  | GSM8K           | CoT           | <b>86.8</b>     |
|                          |                 | TG - Transfer | <b>86.8</b>     |
|                          | Object Counting | CoT           | 86.9            |
|                          |                 | TG - Transfer | <b>94.9</b>     |
|                          | Word Sorting    | CoT           | <b>46.5</b>     |
|                          |                 | TG - Transfer | 45.5            |
| Llama 3.1 - 70B Instruct | GSM8K           | CoT           | <b>94.4</b>     |
|                          |                 | TG - Transfer | 94.0            |
|                          | Object Counting | CoT           | 94.9            |
|                          |                 | TG - Transfer | <b>96.0</b>     |
|                          | Word Sorting    | CoT           | 73.7            |
|                          |                 | TG - Transfer | <b>84.8</b>     |

**Comment #3.9**

October 9, 2024

47 of 54

*4. I wonder how "correct" the textual feedback provided by the LLMs in this framework is. If the feedback is inaccurate or varies over time, this could potentially impact the optimization process.*

**Response to #3.9**

Thank you for this question. We report the learning curves for prompt optimization. In particular, in Figure 10, we report the accuracy over the test set on the y-axis, and the number of optimization iterations we performed on the x-axis. We often observe that performance improves over time. We use a proposal-validation loop for **TEXTGRAD** prompt optimization, where given a new proposal for a variable, we test it on the validation set and decide to accept the proposal based on the performance.

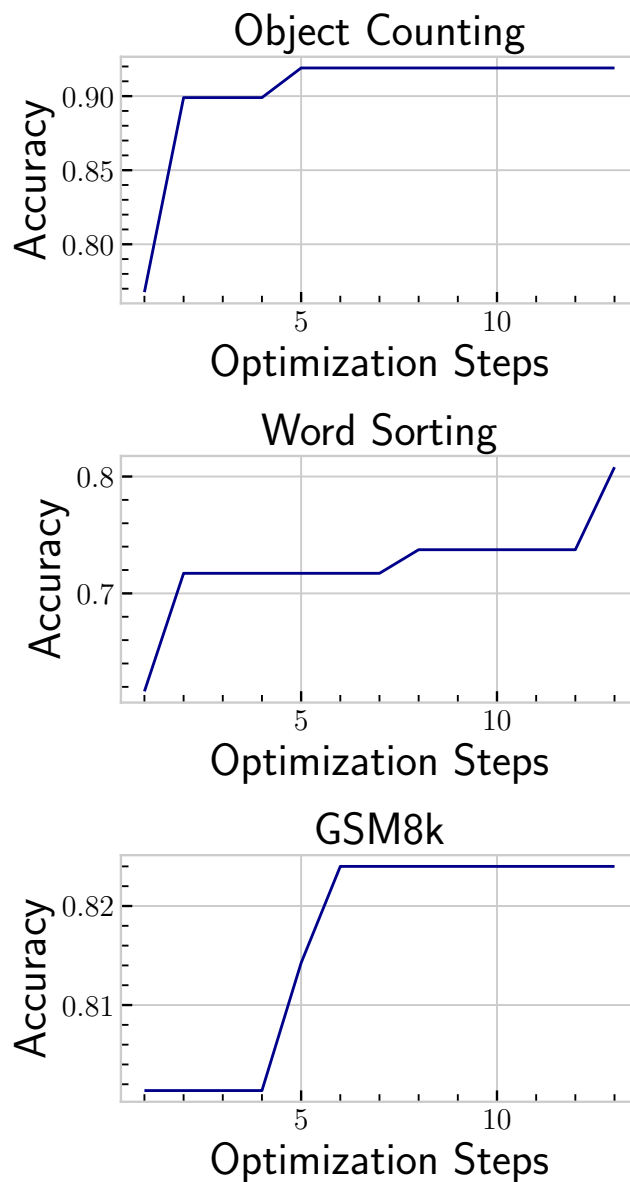


Figure 10: Learning curves for prompt optimization. Here, x-axis is the number of optimization iterations, and y-axis is the accuracy over the entire test set. We observe that optimization with **TEXTGRAD** is stable and the performance does not deteriorate over time.

#### Comment #3.10

*5. There are several self-reflection and self-correction approaches in the literature (e.g., see the survey: <https://arxiv.org/abs/2308.03188>). I am curious how the proposed approach compares to these.*

*Overall, the paper provides appropriate references to related work. However, I would like to see more discussion with related concurrent work such as "LLMs as optimizer" as it seems to me the idea is quite similar at first glance.*

#### Response to #3.10

Thank you for making this point! We agree that **TEXTGRAD** has important relations to both concepts in the existing literature. We have added new experiments and discussions to highlight the benefit of **TEXTGRAD** compared to these related works.

**Performance comparisons:** We compare to the state-of-the-art reflection method, Reflexion [6], and also to the LLMs as optimizer approach, OPRO [5]. Results can be found in Table 8 and Table 6. **TEXTGRAD** consistently outperforms these approaches in both language tasks and vision-language tasks.

Table 8: (New Table 2) Solution optimization for zero-shot question answering with gpt-4o. **TEXTGRAD** outperforms the baselines consistently across different tasks.

| Dataset                           | Method                    | Accuracy (%) |
|-----------------------------------|---------------------------|--------------|
| <b>Natural Language Tasks</b>     |                           |              |
| Google-proof QA [12]              | CoT [10, 11]              | 51.0         |
|                                   | Reflexion [6]             | 51.0         |
|                                   | GPT-4o best reported [13] | 53.6         |
|                                   | <b>TEXTGRAD</b>           | <b>55.0</b>  |
| MMLU-Machine Learning [14]        | CoT [10, 11]              | 85.7         |
|                                   | Reflexion [6]             | 85.7         |
|                                   | <b>TEXTGRAD</b>           | <b>88.4</b>  |
| MMLU-College Physics [14]         | CoT [10, 11]              | 91.2         |
|                                   | Reflexion [6]             | 94.1         |
|                                   | <b>TEXTGRAD</b>           | <b>95.1</b>  |
| <b>Multimodal Reasoning Tasks</b> |                           |              |
| MathVista [4]                     | CoT [10, 11]              | 63.8         |
|                                   | Reflexion [6]             | 64.0         |
|                                   | <b>TEXTGRAD</b>           | <b>66.1</b>  |
| ScienceQA-IMG [3]                 | CoT [10, 11]              | 91.6         |
|                                   | Reflexion [6]             | 90.2         |
|                                   | <b>TEXTGRAD</b>           | <b>92.4</b>  |

**Generality:** In addition to better performance, we would also like to highlight the generality of our framework. The codebase for Reflexion and OPRO have a separate implementation and/or prompting strategy for every task. For each task, there are separate implementations and prompts, making it nontrivial for users to apply these approaches to a new task.

With **TEXTGRAD**, a user does not need to modify the framework to apply to a new domain. It works out of the box for all the domains we tested without any modifications, and we spent a great deal of work to make sure optimization works as broadly as possible.

Finally, we add further discussion on the relation to Yang *et al.* [5]. Here is the change to the writing:

*Third, OPRO [5] is one of the first papers to use LLMs as optimizers. OPRO uses an optimizer LLM to generate solution candidates, and uses the downstream task performance (called ‘score’) to iteratively select and improve available candidates. Importantly, since OPRO relies on a scalar supervision signal to select and improve solution candidates, it is not possible without further innovation to use it in settings such as instance optimization, or settings with multiple components where we use backpropagation. Furthermore, the implementation of the framework depends on the task, unlike the generality of our framework where we do not make any modifications to the implementation for new tasks. Finally, both DSPy, ProTeGi, and OPRO focused on prompt optimization as the key application, while a significant advance of **TEXTGRAD**, as demonstrated through our diverse applications, is in instance optimization.*

### **Comment #3.11**

*H. Clarity and context: lucidity of abstract/summary, appropriateness of abstract, introduction and conclusions The paper is well-written. The abstract and introduction provide strong motivation and clear intuition for the work. It articulates the details of the proposed framework with illustrative examples and provides excellent real-world applications that demonstrate how the framework can be used.*

### **Response to #3.11**

We sincerely appreciate your positive feedback on the clarity and presentation of our paper. We’re grateful that they found the abstract and introduction to provide strong motivation and clear intuition for our work. We’re pleased that our explanations of the framework, along with the illustrative examples and real-world applications, effectively demonstrated **TEXTGRAD**’s utility and potential impact.

---

## References

1. Lu, P., Peng, B., Cheng, H., Galley, M., Chang, K.-W., Wu, Y. N., Zhu, S.-C. & Gao, J. Chameleon: Plug-and-play compositional reasoning with large language models. *Advances in Neural Information Processing Systems* **36** (2024).
2. Yan, B., Zhang, J., Yuan, Z., Shan, S. & Chen, X. Evaluating the Quality of Hallucination Benchmarks for Large Vision-Language Models. *arXiv preprint arXiv:2406.17115* (2024).
3. Lu, P., Mishra, S., Xia, T., Qiu, L., Chang, K.-W., Zhu, S.-C., Tafjord, O., Clark, P. & Kalyan, A. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems* **35**, 2507–2521 (2022).
4. Lu, P., Bansal, H., Xia, T., Liu, J., Li, C., Hajishirzi, H., Cheng, H., Chang, K.-W., Galley, M. & Gao, J. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts in *International Conference on Learning Representations* (2024).
5. Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D. & Chen, X. Large Language Models as Optimizers in *The Twelfth International Conference on Learning Representations* (2024). <https://openreview.net/forum?id=Bb4VGOWELI>.
6. Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K. & Yao, S. *Reflexion: language agents with verbal reinforcement learning* in *Advances in Neural Information Processing Systems* **36** (2023). [https://proceedings.neurips.cc/paper\\_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf](https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf).
7. Gao, W., Fu, T., Sun, J. & Coley, C. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in neural information processing systems* **35**, 21342–21357 (2022).
8. Wang, Q., Wang, R., Liu, J., Jiang, F., Yue, H., Du, Y. & Wu, H. High-dimensional automated radiation therapy treatment planning via Bayesian optimization. *Medical Physics* **50**, 3773–3787 (2023).
9. AI@Meta. Llama 3 Model Card. [https://github.com/meta-llama/llama3/blob/main/MODEL\\_CARD.md](https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md) (2024).
10. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., *et al.* Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* **35**, 24824–24837 (2022).
11. Kojima, T., Gu, S. S., Reid, M., Matsuo, Y. & Iwasawa, Y. Large language models are zero-shot reasoners. *Advances in neural information processing systems* **35**, 22199–22213 (2022).
12. Rein, D., Hou, B. L., Stickland, A. C., Petty, J., Pang, R. Y., Dirani, J., Michael, J. & Bowman, S. R. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022* (2023).
13. OpenAI. *Hello GPT-4o* Accessed: 2024-05-18. 2024. <https://openai.com/index/hello-gpt-4o/>.

14. Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D. & Steinhardt, J. *Measuring Massive Multitask Language Understanding in International Conference on Learning Representations* (2021). <https://openreview.net/forum?id=d7KBjmI3GmQ>.
15. AI, J. *EasyOCR* 2024. <https://jaided.ai/easyocr/>.
16. Jensen, J. *A graph-based genetic algorithm and generative model/Monte Carlo tree search for the exploration of chemical space. Chem Sci* 10 (12): 3567–3572 2019.
17. Olivecrona, M., Blaschke, T., Engkvist, O. & Chen, H. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics* 9, 1–14 (2017).
18. Tripp, A., Simm, G. N. & Hernández-Lobato, J. M. *A fresh look at de novo molecular design benchmarks in NeurIPS 2021 AI for Science Workshop* (2021).
19. García-Ortegón, M., Simm, G. N., Tripp, A. J., Hernández-Lobato, J. M., Bender, A. & Bacallado, S. DOCKSTRING: easy molecular docking yields better benchmarks for ligand design. *Journal of chemical information and modeling* 62, 3486–3502 (2022).
20. Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q., Chi, E., Zhou, D. & Wei, J. *Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them in Findings of the Association for Computational Linguistics: ACL 2023* (Association for Computational Linguistics, Toronto, Canada, July 2023). <https://aclanthology.org/2023.findings-acl.824>.
21. Srivastava, A. *et al.* Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*. ISSN: 2835-8856. <https://openreview.net/forum?id=uyTL5Bvosj> (2023).
22. Khattab, O., Singhvi, A., Maheshwari, P., Zhang, Z., Santhanam, K., A, S. V., Haq, S., Sharma, A., Joshi, T. T., Moazam, H., Miller, H., Zaharia, M. & Potts, C. *DSPy: Compiling Declarative Language Model Calls into State-of-the-Art Pipelines in The Twelfth International Conference on Learning Representations* (2024). <https://openreview.net/forum?id=sY5N0zY50d>.
23. Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C. & Schulman, J. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168* (2021).
24. Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., *et al.* The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).
25. Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., Casas, D. d. l., Hanna, E. B., Bressand, F., *et al.* Mixtral of experts. *arXiv preprint arXiv:2401.04088* (2024).
26. Huang, K., Fu, T., Gao, W., Zhao, Y., Roohani, Y., Leskovec, J., Coley, C. W., Xiao, C., Sun, J. & Zitnik, M. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *arXiv preprint arXiv:2102.09548* (2021).
27. Nicolaou, C. A. & Brown, N. Multi-objective optimization methods in drug design. *Drug Discovery Today: Technologies* 10, e427–e435 (2013).

28. Hoelder, S., Clarke, P. A. & Workman, P. Discovery of small molecule cancer drugs: successes, challenges and opportunities. *Molecular oncology* **6**, 155–176 (2012).
29. Cieplinski, T., Danel, T., Podlowska, S. & Jastrzebski, S. Generative models should at least be able to design molecules that dock well: A new benchmark. *Journal of Chemical Information and Modeling* **63**, 3238–3247 (2023).
30. Swanson, K., Walther, P., Leitz, J., Mukherjee, S., Wu, J. C., Shivnaraine, R. V. & Zou, J. ADMET-AI: A machine learning ADMET platform for evaluation of large-scale chemical libraries. *bioRxiv*, 2023–12 (2023).
31. Kontoyianni, M. Docking and virtual screening in drug discovery. *Proteomics for drug discovery: Methods and protocols*, 255–266 (2017).
32. Agarwal, S. & Mehrotra, R. An overview of molecular docking. *JSM chem* **4**, 1024–1028 (2016).
33. Trott, O. & Olson, A. J. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry* **31**, 455–461 (2010).
34. Yoshikawa, N., Terayama, K., Sumita, M., Homma, T., Oono, K. & Tsuda, K. Population-based de novo molecule generation, using grammatical evolution. *Chemistry Letters* **47**, 1431–1434 (2018).
35. Kisling, K., McCarroll, R., Zhang, L., Yang, J., Simonds, H., Du Toit, M., Trauernicht, C., Burger, H., Parkes, J., Mejia, M., *et al.* Radiation planning assistant-a streamlined, fully automated radiotherapy treatment planning system. *JoVE (Journal of Visualized Experiments)*, e57411 (2018).
36. Huang, C., Nomura, Y., Yang, Y. & Xing, L. Meta-optimization for fully automated radiation therapy treatment planning. *Physics in Medicine & Biology* **67**, 055011 (2022).
37. Yang, Y. & Xing, L. Clinical knowledge-based inverse treatment planning. *Physics in Medicine & Biology* **49**, 5101 (2004).
38. Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. *Optuna: A Next-generation Hyperparameter Optimization Framework in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2019).

## Referee #1 (Remarks to the Author):

*The authors have made commendable efforts to address and incorporate the feedback provided by all reviewers. I found the reading process quite enlightening, as it offered new insights and knowledge. The work presented is certainly both interesting and valuable.*

*However, after careful consideration, particularly in light of the points raised by reviewer 3, I still feel that the manuscript may not fully align with the publication standards of Nature. While this assessment is inherently subjective, it is important to note that a framework such as Auto-Grad, which can be likened to Pytorch in its domain, might not require the platform of Nature for its dissemination and recognition. At its current stage, a highly influential application or very broad usage might be necessary to meet the criteria for publication in Nature.*

Thank you for your thoughtful feedback; your suggestions have truly helped to strengthen the paper. We greatly appreciate your recognition of our efforts to address and incorporate the reviewers' comments and acknowledgment of the insights and value in our work.

## Referee #2 (Remarks to the Author):

*The authors have made numerous changes to the manuscript that I believe has strengthened it and clarified its contributions. I'm supportive of its publication.*

- 1. I appreciate the additional evaluations using a smaller base model and the discussion of cost required for optimization.*
- 2. I appreciate the additional contextualization with respect to Reflexion, OPRO, and DSPy. The emphasis on (and benefit of) generality is noted.*
- 3. I appreciate the decision to move the molecular optimization example to the Appendix and the additional caveats of the analysis (i.e., the existence of many additional considerations in molecular design). I know my initial comments were very picky about the language, so I am grateful that they were considered seriously. The inclusion of comparisons from the PMO benchmark provide better evidence of strong empirical performance in this proof of concept.*

Thank you! We sincerely appreciate your positive feedback and support for the publication of our work. Thank you very much for helping us make our manuscript stronger and more thoughtful.

*When skimming some of the examples in the appendices, I noticed in D.2 that the GPQA Diamond example has mis-formatted superscripts; might be worth checking others.*

Thank you for pointing this out. We have corrected the misformatted superscripts in the GPQA Diamond example in the Supplement. Additionally, we reviewed the entire manuscript to ensure that all formatting, including superscripts, is consistent and error-free.

### **Referee #3 (Remarks to the Author):**

*I would like to thank the authors for thoroughly addressing all the feedback and for including new experiments on MathVista and ScienceQA, as well as exploring the setting of optimizing compound AI systems.*

Thank you! We sincerely appreciate your positive feedback and support for the publication of our work.

*The results on MathVista and ScienceQA are promising, with TextGrad demonstrating improvements over strong published baselines in both cases. These experiments address my concerns.*

We are glad to hear that our results on MathVista and ScienceQA, showcasing TextGrad's improvements over strong published baselines, addressed your concerns. Thank you for acknowledging the promise of our approach in these experiments.

*However, the claim regarding the reduction of hallucinations in the multimodal language model for VQA is not convincing to me. First, the authors mention that the experiment is only on "the spatial relation type, using 200 images from this subset". In such a case, the authors should only claim that TextGrad improves spatial reasoning in vision-language models. Second, it is unclear why the authors chose to conduct experiments on HQH rather than other well-established VQA or spatial relation benchmarks. Third, the definition of hallucination in HQH seems problematic. For example, in Figure 3d, the initial answer appears to involve an object recognition error rather than what is typically considered hallucination. In vision-language, hallucination is not a well-defined term, but it is generally understood as VL models generate captions that include objects not present in the input images. However, HQH seems to consider all types of errors as hallucinations. I would recommend avoiding the use of such a vague term without further clarification.*

Thank you for your helpful feedback. We agree with your points and have revised our claims accordingly. Specifically, based on your suggestion, we reduced the claim of "removing hallucinations" to "improving the spatial reasoning ability" of vision-language models. We emphasized that this experiment focuses on spatial reasoning errors and provided more precise language to describe these phenomena in the revised paper. We chose the HQH dataset because performance on several popular benchmarks such as VQA (Agrawal et al 2015) and GQA (Hudson et al 2019) have nearly saturated, especially as many of these datasets used COCO images, which have been included in the training of multimodal language models. In contrast, HQH, released in 2024, still shows evolving performance with GPT models, making it a useful benchmark for evaluating the impact of TextGrad.

*Overall, the paper is interesting and demonstrates that TextGrad can be effectively applied across various scenarios. Therefore, I lean into accepting the paper. However, I encourage the authors to dive deeper into each experimental scenario, providing intuition on how TextGrad contributes to performance improvements and ensuring that all claims are supported by the results.*

We appreciate your positive feedback and are glad that you find the paper interesting and the application of TextGrad effective across various scenarios. We agree with your suggestion to dive deeper into each experimental scenario. In the revised manuscript, we provided additional intuition on how TextGrad contributes to performance improvements, elaborating on the mechanisms underlying its success. Furthermore, we ensured that all claims are supported by the results, with clear connections between our methods, experiments, and findings.

Thank you for your thoughtful recommendation, which has helped us strengthen the clarity and depth of our work.