Evaluating Large Language Models for G-Code Debugging, Manipulation, and Comprehension

Anushrut Jignasu[a] Kelly Marshall[b] Baskar Ganapathysubramanian[a] Aditya Balu[a] Chinmay Hegde[b]
Adarsh Krishnamurthy[a]
Email: [a] (ajignasu | baskarg | baditya | adarsh)@iastate.edu , [b] (km3888 | chinmay.h)@nyu.edu

2024 IEEE LLM Aided Design Workshop (LAD) | 979-8-3503-7608-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/LAD62341.2024.10691700

***Abstract*—3D printing is a revolutionary technology that enables the creation of physical objects from digital models. However, the quality and accuracy of 3D printing depend on the correctness and efficiency of the numerical control programming language (specifically, G-code) that instructs 3D printers on moving and extruding material. Debugging G-code, a low-level programming language for 3D printing, is a challenging task that requires manual tuning and geometric reasoning. In this paper, we present the first extensive evaluation of numerous large language models (LLMs) for debugging G-code files for 3-axis 3D printing. We design effective prompts to enable pre-trained LLMs to understand and manipulate G-code and test their performance on various aspects of G-code debugging and manipulation, including detection and correction of common errors and the ability to perform geometric transformations. We compare different state-of-the-art LLMs and analyze their strengths and weaknesses. We also discuss the implications and limitations of using LLMs for G-code comprehension and suggest directions for future research.**

***Index Terms*—G-code, Large language models, Debugging, Geometric comprehension, Manufacturing 4.0.**

I. Introduction

The 3D printing revolution has democratized the easy, scalable, and efficient creation of physical objects from digital models. It has impacted applications in various domains, such as manufacturing, healthcare, construction, and the arts. However, the quality and accuracy of 3D printing depend on the correctness and efficiency of *G-code*, a low-level numerical control programming language that instructs 3D printers how to move and extrude material. Dedicated software can generate the G-code for a particular part from the computer-aided design (CAD) model. Still, the efficacy of the generated G-code to correctly 3D print parts depends on extensive manual tuning of the G-code generation software. Consequently, generating high-quality G-code requires considerable expertise and is often an iterative process with manual debugging of the G-code itself. Debugging and modifying G-code for 3D printing involves complex syntactic, semantic, and geometric challenges, with errors potentially leading to equipment damage or human harm. G-code errors, categorized into syntax, configuration, and geometry issues, necessitate precise debugging solutions.

a. Iowa State University, USA
b. New York University, USA
This work was supported by the National Science Foundation under grant CMMI-AM-2347623/2347624. We would like to thank the NVIDIA Corporation for providing GPUs for academic research.

Recent research has shown that natural language descriptions can be used for various tasks related to 3D printing, such as generating novel shapes [1]–[4], editing scenes [5], and reasoning about geometry in the volume space [6]. We present the first comprehensive evaluation of the state-of-the-art pre-trained LLMs for debugging and modifying G-code files for extrusion-based additive manufacturing (AM) without requiring fine-tuning or domain adaptation; our main technical challenge is demonstrating their generalization abilities by framing (a suitable chain of) prompts for G-code manipulation. The main contributions of the paper are the following:

(i) We show how careful prompt engineering can enable pre-trained large language models (LLMs) to understand G-codes for AM. We develop effective strategies for feeding large G-code files to context-length-limited LLMs.

(ii) We systematically evaluate different state-of-the-art LLMs on a suite of comprehension tasks, including G-code debugging, capturing geometric transformations, and suggesting corrections. We compare the performance of different LLMs and analyze their strengths and weaknesses.

(iii) Best practices for using LLMs for dealing with such low-level (assembly-like) languages. We also discuss the implications and limitations of using LLMs for G-code debugging and comprehension, and finally, directions for future research.

II. Related Work

Large Language Models. Interest in the use of Large Language Models (LLMs), particularly in the research community, has increased exponentially in the last several months (as of summer 2023). The generative capabilities of these LLMs have yielded promising applications spanning a wide range of domains. Prominent LLMs include proprietary models like OpenAI's GPT series [7], [8] as well as LLMs which have been open-sourced for use by the general public such as Llama [9] and Starcoder [10]. Further work has extended LLM's applicability through the use of Reinforcement Learning from Human Feedback [11], which fine-tunes language models on reward signals provided by human evaluators. This has spawned interactive versions of the aforementioned base models as well as some chat-specific competitors such as Google's Bard/Gemini [12] and Anthropic's Claude [13]. These chatbots allow users to interface with an LLM's knowledge base and obtain useful answers that align with their preferences rather than simply receiving a prediction of the most likely text.

TABLE I: *Overview of capabilities we evaluate on for various LLMs[1]. Color coding represents the average of performance across the shapes. depicts success, depicts partial success, and shows failure. Note: Debugging was only evaluated for the S-shape.*

Capability	`GPT-3.5`	`GPT-4`	`Claude-2`	`Llama-2-70b`	`Starcoder`
Debugging					
Simple Transformations					
Complex Transformations					
Comprehension					

Large Language Models for Additive Manufacturing. Just as humans have evolved with language being the primary mode of information exchange, most AM techniques require a way to understand the model geometry, and this usually comes in the form of Geometric code (G-code). The use of LLMs in manufacturing is nascent and largely unexplored as of early-2024. The closest work to our approach is a recent preprint by Makatura [14], which makes a deep dive into how a specific LLM (ChatGPT) can be used in design and manufacturing applications. Another recent work by Badini [15] has looked into utilizing a specific LLM — ChatGPT — for AM process parameter optimization. They address printing-related issues for extrusion-based additive manufacturing and assess the utility of ChatGPT to modify the process parameters to reduce common errors such as warping, bed detachment, and stringing. However, it should be noted that they do not use the entire G-code file as input for modifying these parameters; these parameters are stored in the G-code header.

Prompt Engineering Given the nuanced complexities of human language, where words can take on context-specific meanings, effective prompting is essential to guide LLMs' outputs in a more reliable and predictable manner. "Prompt engineering" involves designing task-specific language prompts to condition the LLM during inference and techniques range from automated methods [16], [17] to manual approaches [18]. However, the scalability of gradient-based prompt-tuning techniques remains an open issue [19]–[21]. Research has explored the utility of fundamental contextual statements as reusable prompt patterns [22], generating multiple prompts for a single task and aggregating the responses, employing weak supervision techniques for final predictions [23], and decomposed prompting for more refined outputs [24]. Additionally, the sensitivity of LLMs to the phrasing and structure of prompts has been analyzed to develop more effective single-prompt strategies [25], [26]. In our evaluations below, we leverage the technique of "chain-of-thought" [27] reasoning, a form of in-context learning used to precondition an LLM for G-code-related tasks.

Our contributions are distinct and complementary to these existing works. We focus on assessing the capabilities of a diverse set of currently available LLMs for performing error correction, manipulating geometric shapes, and comprehending entire G-code files. Along the way, we address a key technical (and novel) challenge: dealing with limited context window lengths. We discuss details in the next section.

Use these definitions for in-context learning. For context, I have provided example commands for a 3D printer.
M104 S205 sets nozzle temperature to 205 degrees Celsius
G28 homes all axes
G1 Z5 F5000 moves the Z axis up 5mm at a speed of 5000mm/min
M109 S200 sets bed temperature to 200 degrees Celsius
G21 sets all units to millimeters
G90 uses absolute coordinates
M82 uses absolute distance for extrusion
G92 E0 sets extrusion to 0
M107 turns off the fan
M106 S255 turns on the fan at full speed. S128 would turn it on at half speed
G1 X90 Y90 F7800 moves the print head in X and Y to 90mm at a speed of 7800mm/min
G1 E2 F2400 extrudes 2mm of filament at a speed of 2400mm/min
G1 F1800 sets the feedrate to 1800mm/min

Fig. 1: *Initial prompt to each LLM*

III. Methods

Our work aims to study the ability of LLMs to perform a range of operations — simple debugging, geometric transformations (such as translation, scaling, shearing, and rotation), and finally, geometry comprehension. Basic debugging evaluates the ability of the LLM to easily locate syntax errors. Geometric transformations change an object's position, orientation, and size in a predefined coordinate system. An LLM's ability to perform such transformations is a precursor to its ability to spatially understand, reason about, and manipulate an input geometry represented as G-code. Finally, we also use the LLM to reason about the geometry from the G-code alone. We utilize three canonical shapes for experimentation: a cube, a cylinder, and an S-shape. Each of the shapes was generated using Solidworks [28], sliced using PrusaSlicer [29], and visualized in Ultimaker's Cura [30].

We selected a diverse subset of the best available closed- and open-source LLMs as of late 2023 for evaluation. GPT-3.5 and GPT-4 from OpenAI [8], Anthropic's Claude-2 [13], Meta's open-source Llama-2-70b [9] fine-tuned for chat (Llama-2-70b-chat-hf) by Huggingface, and the BigCode community's open-source Starcoder-Starchat-Beta [10] fine-tuned for chat and hosted on Huggingface. We adhere to default settings for all LLMs except for Starcoder's Starchat, where we set the *maximum new tokens* to 1024 to allow us to generate G-code

[1]We also evaluated Bard [12], but the results are not included here since its incorporation into Gemini.

TABLE II: *Overview of geometric transformation capabilities of the LLMs[2] on different shapes. Color coding represents the performance. depicts success, depicts partial success, and shows failure.*

Shape	Capability	`GPT-3.5`	`GPT-4`	`Claude-2`	`Llama-2-70b`	`Starcoder`
Cube	Translation					
	Scale					
	Shear					
	Rotation					
Cylinder	Translation					
	Scale					
	Shear					
	Rotation					
S-shape	Translation					
	Scale					
	Shear					
	Rotation					

sequences without token length limits. Furthermore, we make a concerted effort to use the same prompts for evaluation across all LLMs. However, owing to inherently different model architectures, attention mechanisms, different training datasets, and varying token lengths of the LLMs, such variance inherently affects the LLM's response to identical prompts, necessitating the occasional use of different prompts to achieve the same task across different models.

All our dialogues are performed with atleast one of the following traits: **(i)** We begin each conversation with the same prompt (Fig. 1). **(ii)** We provide chunks of G-code, owing to the varying token lengths. Empirically, we have found that a chunk length of 500 lines works for GPT-3.5 and GPT-4. For Claude-2, Llama-2-70b, and Starcoder, we were able to provide the entire G-code as input. **(iii)** Depending on the evaluation metric, we provide G-code for the first layer on a conditional basis. **(iv)** We provide user feedback to solutions that are incomplete or omit key parts of the g-code. This is necessary to observe the support for iteration noted in [14]. For these prompts, we maintain as much uniformity as possible between tasks and models.

Translation transformation moves the object by a fixed distance (10mm in **X** and 20mm in **Y**) without changing its orientation or size. The process involved specific prompts asking the LLM to adjust the G-code for these movements. For ***Scaling***, we applied a uniform scaling factor of **2** and ask the LLM to scale the coordinates accordingly. ***Shearing*** transformations were aimed at skewing the object by a factor of **0.5**, and ***Rotation*** involved turning the objects **45 degrees** (counter-clockwise) with respect to the Z axis.

[2]We also evaluated Bard [12], but the results are not included here since its incorporation into Gemini.

The second aspect of our work focuses on the LLMs' comprehension abilities. We define basic comprehension as the ability of the LLM to understand the underlying geometry from the G-code and consequently identify the 3D model being manufactured. We begin by providing G-code chunks of varying lengths, depending on the LLMs token limits, and asking them to combine all chunks and deduce the shape and dimensions of the manufactured object. We used a range of prompts to test the LLMs, ensuring a thorough evaluation of their comprehension abilities. Furthermore, we extend our evaluation of manufacturing-specific comprehension abilities by providing a configuration file containing various parameter values used by slicing software (in this case, PrusaSlicer [29]).

IV. Results and Discussion

Our results for the LLMs capabilities are summarized in Table I. *Simple Transformations* include translation and scaling, *Complex Transformations* include shearing and rotation, and *Comprehension* is about the LLM's ability to understand the underlying geometry. We summarize results in Table II.

A. Simple Debugging

We test various LLM chatbots on a simple 2D S-Shape (204 lines; suitable for all context lengths) for line-by-line debugging and replace the "G1" command in one line with "G13", an invalid G-code command. For each model, we first give it only the G-code file and ask it to look for an error. If the model understands the task but is unable to determine the cause of the error, we then provide an additional prompt saying that we are looking for an invalid command. GPT-4 performs the best out of all the models, finding the error immediately and pointing it out. Claude-2 also performs well, getting the correct answer after receiving the additional hint. GPT-3.5 is unable to solve the problem. Neither Starcoder nor Llama-2-70b are able to process the input and recognize that it is a G-Code file. An example transcript with GPT-4 is shown in Fig. 2.

Hey I think there's an erroneous line in this G-Code file, could you help me find it?

User provides edited S-Shape G-Code

Upon a quick scan of your provided G-code, the suspicious line is:

```
G13 X104.506 Y100.205 E3.67803
```

The command G13 is not standard for most G-code interpreters used in 3D printing or CNC machining. The likely intended command is G1, which is a linear move.
You should probably change this line to:

```
G1 X104.506 Y100.205 E3.67803
```

...

Fig. 2: *Prompting GPT-4 to find a bug in the G-Code.*

(omitted by authors: See Fig. 1 for initial input prompt)
can you tell me what shape is being represented by this gcode?
User provides entire G-code for a cube

Based on analyzing the GCODE commands, it looks like this is printing a square shape with some infill. Here are the key things I noticed:

...
The first layer is at Z0.35 and prints a perimeter of a square from X97.175 to X102.825 and Y97.175 to Y102.825.
...
The Z height reaches 6.35mm on the last layer, so with a first layer height of 0.35mm, it's printing a total of 18 layers.

So in summary, the shape is a square with solid infill, printing 18 layers to a height of 6.35mm. The square size starts at 5.65 x 5.65mm on the first layer and shrinks slightly on each layer. Let me know if you need any other details explained!

Fig. 4: *Prompting Claude-2 to translate G-Code for a cube.*

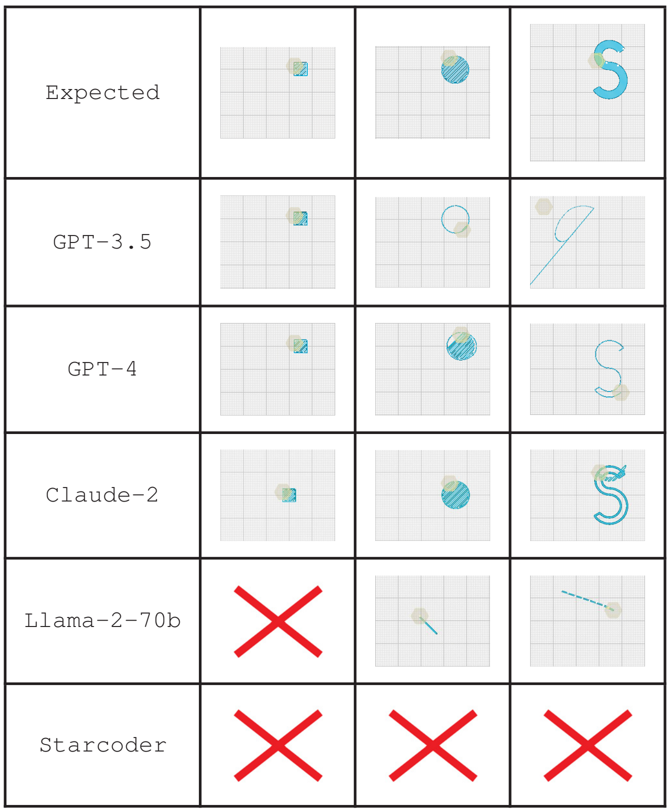

Fig. 3: *G-code visualization for translation operation on all LLMs. Expected G-code (top row). GPT-3.5 (second row), GPT-4 (third row), Claude-2 (fourth row), Llama-2-70b (fifth row), Starcoder (last row).*

B. Geometric Transformations

We conduct dialogues that ask LLMs in our test suite to perform translation, scaling, shearing, and rotation on the provided G-code. Due to the varying token lengths across all the LLMs we test, the geometric transformations may or may not be applied to the entire G-code. To assess the effect of this potential bottleneck we also test their performance on G-code for a single layer. This led to more engaging and positive results than prompts asking for evaluation on the entire G-code.

Fig. 3 showcases G-code generated by various LLMs for the translation operation. An example chat with Claude-2 for a cube is shown in Fig. 4.

C. Comprehension

Our examination of the LLMs in this work reveals several limitations attributed to differences in underlying architectures, training data, tokenization schemes, and token length constraints. A pervasive issue across most of the LLMs we tested is their inability to completely parse an entire G-code file in a single-shot manner. Our empirical observations suggest that this shortcoming adversely affects the models' capability to reason about the geometric intricacies represented in the G-code. While the models showed some aptitude in understanding two-dimensional counterparts, i.e., single layers of G-code, they struggled with the three-dimensional portion, i.e., the entire G-code. Our interaction with Claude-2 indicates that a longer context length significantly enhances output quality. It would be premature to definitively claim that context length limitations impede an LLM's spatial reasoning capability of the underlying geometry; it is plausible that this is a contributing factor and makes for an interesting direction for future research.

V. Conclusions

In this work, we rigorously assessed various large language models (LLMs), both closed- and open-source, for their proficiency with Gcode in debugging, manipulation, and geometric reasoning tasks. The outcomes demonstrate notable variances in performance for our pre-defined tasks. GPT-4 emerges as the most proficient, closely followed by Claude-2, while the open-source LLMs such as Llama-2-70b and Starcoder showed poor performance, highlighting a critical research gap. Furthermore, our study reveals a general deficiency in the nuanced comprehension of complex geometries from G-code data, often due to limited context windows influencing parsing abilities. These limitations point the way toward future directions for follow-up research. A full length version of our paper is on *ArXiv* and our results can be found on *GitHub*.

References

[1] A. Sanghi, H. Chu, J. G. Lambourne, Y. Wang, C.-Y. Cheng, M. Fumero, and K. R. Malekshan, "Clip-forge: Towards zero-shot text-to-shape generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 603–18 613.

[2] K. O. Marshall, M. Pham, A. Joshi, A. Jignasu, A. Balu, and A. K. C. Hegde, "Zeroforge: Feedforward text-to-shape without 3d supervision," *arXiv preprint arXiv:2306.08183*, 2023.

[3] A. Jain, B. Mildenhall, J. T. Barron, P. Abbeel, and B. Poole, "Zero-shot text-guided object generation with dream fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 867–876.

[4] C.-H. Lin, J. Gao, L. Tang, T. Takikawa, X. Zeng, X. Huang, K. Kreis, S. Fidler, M.-Y. Liu, and T.-Y. Lin, "Magic3d: High-resolution text-to-3d content creation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 300–309.

[5] A. Haque, M. Tancik, A. A. Efros, A. Holynski, and A. Kanazawa, "Instruct-nerf2nerf: Editing 3d scenes with instructions," *arXiv preprint arXiv:2303.12789*, 2023.

[6] J. Kerr, C. M. Kim, K. Goldberg, A. Kanazawa, and M. Tancik, "Lerf: Language embedded radiance fields," *arXiv preprint arXiv:2303.09553*, 2023.

[7] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[8] OpenAI, "GPT-4 technical report," 2023.

[9] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "LLaMA: Open and efficient foundation language models," 2023.

[10] R. Li, L. B. Allal, Y. Zi, N. Muennighoff, D. Kocetkov, C. Mou, M. Marone, C. Akiki, J. Li, J. Chim, Q. Liu, E. Zheltonozhskii, T. Y. Zhuo, T. Wang, O. Dehaene, M. Davaadorj, J. Lamy-Poirier, J. Monteiro, O. Shliazhko, N. Gontier, N. Meade, A. Zebaze, M.-H. Yee, L. K. Umapathi, J. Zhu, B. Lipkin, M. Oblokulov, Z. Wang, R. Murthy, J. Stillerman, S. S. Patel, D. Abulkhanov, M. Zocca, M. Dey, Z. Zhang, N. Fahmy, U. Bhattacharyya, W. Yu, S. Singh, S. Luccioni, P. Villegas, M. Kunakov, F. Zhdanov, M. Romero, T. Lee, N. Timor, J. Ding, C. Schlesinger, H. Schoelkopf, J. Ebert, T. Dao, M. Mishra, A. Gu, J. Robinson, C. J. Anderson, B. Dolan-Gavitt, D. Contractor, S. Reddy, D. Fried, D. Bahdanau, Y. Jernite, C. M. Ferrandis, S. Hughes, T. Wolf, A. Guha, L. von Werra, and H. de Vries, "StarCoder: May the source be with you!" 2023.

[11] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," 2022.

[12] Google, "Google Bard," 2023. [Online]. Available: https://bard.google.com/u/2/

[13] Anthropic, "Claude," Aug 2023. [Online]. Available: https://www.anthropic.com/product

[14] L. Makatura, M. Foshey, B. Wang, F. HähnLein, P. Ma, B. Deng, M. Tjandrasuwita, A. Spielberg, C. E. Owens, P. Y. Chen *et al.*, "How can large language models help humans in design and manufacturing?" *arXiv preprint arXiv:2307.14377*, 2023.

[15] S. Badini, S. Regondi, E. Frontoni, and R. Pugliese, "Assessing the capabilities of chatgpt to improve additive manufacturing troubleshooting," *Advanced Industrial and Engineering Polymer Research*, 2023.

[16] T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, "Autoprompt: Eliciting knowledge from language models with automatically generated prompts," *arXiv preprint arXiv:2010.15980*, 2020.

[17] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners," *arXiv preprint arXiv:2012.15723*, 2020.

[18] L. Reynolds and K. McDonell, "Prompt programming for large language models: Beyond the few-shot paradigm," in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–7.

[19] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," *arXiv preprint arXiv:2104.08691*, 2021.

[20] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "Gpt understands, too," *AI Open*, 2023.

[21] G. Qin and J. Eisner, "Learning how to ask: Querying lms with mixtures of soft prompts," *arXiv preprint arXiv:2104.06599*, 2021.

[22] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, "A prompt pattern catalog to enhance prompt engineering with chatgpt," *arXiv preprint arXiv:2302.11382*, 2023.

[23] S. Arora, A. Narayan, M. F. Chen, L. J. Orr, N. Guha, K. Bhatia, I. Chami, F. Sala, and C. Ré, "Ask me anything: A simple strategy for prompting language models," *arXiv preprint arXiv:2210.02441*, 2022.

[24] T. Khot, H. Trivedi, M. Finlayson, Y. Fu, K. Richardson, P. Clark, and A. Sabharwal, "Decomposed prompting: A modular approach for solving complex tasks," *arXiv preprint arXiv:2210.02406*, 2022.

[25] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen, "What makes good in-context examples for gpt-3?" *arXiv preprint arXiv:2101.06804*, 2021.

[26] Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh, "Calibrate before use: Improving few-shot performance of language models," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 697–12 706.

[27] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 824–24 837, 2022.

[28] SolidWorks Corp., "SolidWorks," 2023. [Online]. Available: www.solidworks.com

[29] PrusaSlicer, "Prusa," 2012-2023. [Online]. Available: https://www.prusa3d.com/page/prusaslicer_424/

[30] Ultimaker, "Ultimaker cura," 2011-2023. [Online]. Available: https://ultimaker.com/software/ultimaker-cura/