

An MIT Exploration of Generative AI • From Novel Chemicals to Opera

Large Language Models for Design and Manufacturing

**Liane Makatura^{1,2} Michael Foshey^{1,3,4} Bohan Wang^{1,2} Felix Hähnlein⁵
Pingchuan Ma^{1,2} Bolei Deng^{1,3,4} Megan Tjandrasuwita^{1,2}
Andrew Spielberg⁶ Crystal Elaine Owens^{1,2} Peter Yichen Chen^{1,2}
Allan Zhao^{1,2} Amy Zhu⁵ Wil J. Norton^{1,2} Edward Gu^{1,2}
Joshua Jacob^{1,2} Yifei Li^{1,2} Adriana Schulz⁵ Wojciech Matusik^{1,3,4}**

¹Department of Electrical Engineering and Computer Science, ²CSAIL, MIT, ³CSAIL,

⁴Department of Mechanical Engineering, MIT,

⁵Paul G. Allen School of Computer Science & Engineering, University of Washington,

⁶John A. Paulson School of Engineering and Applied Sciences, Harvard University

MIT

Published on: Mar 27, 2024

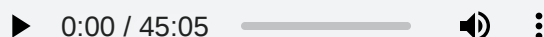
DOI: <https://doi.org/10.21428/e4baedd9.745b62fa>

License: [Creative Commons Attribution-NonCommercial 4.0 International License \(CC-BY-NC 4.0\)](https://creativecommons.org/licenses/by-nc/4.0/)

ABSTRACT

The progress in generative AI, particularly large language models (LLMs), opens new prospects in design and manufacturing. Our research explores the use of these tools throughout the entire design and manufacturing workflow. We assess the capabilities of LLMs in various tasks: converting text prompts into designs, generating design spaces and variations, transforming designs into manufacturing instructions, evaluating design performance, and searching for designs based on performance metrics. We identify and discuss the current strengths and limitations of LLMs, suggesting areas for potential enhancements. Additionally, we examine the ethical implications and propose strategies to mitigate risks associated with employing generative AI in design and manufacturing.

Keywords: design, manufacturing, CAD, CAM, CAE, large language models



 Listen to this article

1. Introduction

Computer-aided technologies (CAx), which encompass computational tools for design, analysis, and manufacturing of products, have significantly influenced various industries such as automotive, aerospace, architecture, electronics, biomedical, and digital media. However, the full potential of CAx workflows is often hindered by challenges such as the need for extensive domain-specific knowledge and the time required to utilize CAx software packages or to integrate these solutions into existing workflows. The advent of generative AI, particularly large language models (LLMs), seems poised to help us overcome these obstacles. By automating or semiautomating each stage of the CAx process, LLMs can serve as a software copilot with intuitive and user-friendly interfaces such as interactive chats with multimodal inputs. This would streamline the use of CAx tools and integrate them more efficiently into broader workflows.

Our analysis¹ examines the standard CAx workflow to identify opportunities for automation or acceleration through generative AI methods. We dissect the workflow into five distinct phases, assessing the potential for efficiency and quality enhancements in each phase by integrating generative AI tools. The components under review are: (1) generating a design, (2) constructing a design space and design variations, (3) preparing and documenting designs for manufacturing, (4) evaluating a design's performance, and (5) discovering high-performing designs within a given performance metric and design space.

Although it would be feasible to create specialized LLMs for CAX, our study highlights the benefits of using generic, pretrained models. We base our experiments on GPT-4,² a leading general-purpose LLM, to demonstrate its application in CAX workflows. This approach illustrates how LLMs can streamline and accelerate the design and production of complex objects. Our analysis further explores how LLMs can integrate with existing solvers, algorithms, tools, and visualizers to create a cohesive workflow. Additionally, we identify current limitations of LLMs in design and manufacturing, pointing out avenues for future enhancements in both the models and the augmented workflows. Finally, we address the ethical considerations and dual-use risks associated with these tools, along with potential strategies for mitigation.

2. Evaluation of Current LLM for Design and Manufacturing

Our evaluation’s primary goal is to explore the potential and challenges of integrating contemporary LLMs into the CAX workflow. Recognizing that every component of the CAX workflow—from design and design spaces to manufacturing instructions, performance metrics, and technical documentation—can be represented as compact programs, we employ LLMs across these diverse tasks. We conceptualize each phase of the CAX workflow as a translation layer, transforming input (which may be natural language or a domain-specific language, DSL for short) into an output DSL. Given LLMs’ proficiency in symbolic manipulations, they show promise in tackling these tasks, potentially enhancing our traditional methods. Our investigation covers each stage of the design and manufacturing process, including design generation from natural language, design space creation, design for manufacturing, performance prediction, and inverse design.

2.1 Generating Designs from Natural Language Specifications

We evaluate the ability of LLMs to create designs from natural language instructions, spanning various design contexts such as individual parts, hierarchical assemblies, and hybrid designs that integrate existing components. The LLM demonstrates proficiency in generating designs from high-level textual input, effectively handling a diverse range of representations and problem domains. For instance, we assess the LLM’s capability using OpenJSCAD,³ an open source JavaScript-based CAD library. In every scenario, the LLM successfully generates coherent and well-structured code, complete with semantically meaningful variables and comments. It also shows skill in interpreting and completing underspecified prompts by inferring and supplying plausible values for missing parameters. An example of this is asking the LLM to design a simple cabinet with a shelf, as depicted in [Figure 1](#). While conducting our experiments, we note some common challenges the LLM faces in generating designs in OpenJSCAD. However, it effectively rectifies most of these issues upon user guidance.

Let's use OpenJSCAD to design a cabinet ... with 3/4" wood sheets, and final exterior dimensions of 30 inches tall, 20 inches wide, and 18 inches deep. The stationary part of the cabinet should be comprised of 6 boards: bottom, top, back, two sides, and one shelf centered inside the cabinet. (... omitted by authors: OpenJSCAD modeling hints; see Appendix A.1 ...)



```
(... omitted by authors: import statements and variable setup ...)
pieces.push(colorize([1, 0, 0], translate([0, 0, -height/2 + thickness/2],
    cuboid({size: [width, depth, thickness]})))) // bottom
pieces.push(colorize([0, 1, 0], translate([0, 0, height/2 - thickness/2],
    cuboid({size: [width, depth, thickness]})))) // top
pieces.push(colorize([0, 0, 1], translate([0, -depth/2 + thickness/2, 0],
    cuboid({size: [width, thickness, height]})))) // back
pieces.push(colorize([1, 1, 0], translate([-width/2 + thickness/2, 0, 0],
    cuboid({size: [thickness, depth, height]})))) // left
pieces.push(colorize([0, 1, 1], translate([width/2 - thickness/2, 0, 0],
    cuboid({size: [thickness, depth, height]})))) // right
pieces.push(colorize([1, 0, 1], translate([0, -thickness/2, 0],
    cuboid({size: [innerWidth, thickness, innerDepth]})))) // shelf
(... omitted by authors: rest of design ...)
```



This is a great first guess! However, there are two issues. First, the shelf is oriented vertically, rather than horizontally. Second, there are many cases where the boards are protruding into one another. (... omitted by authors: description of the desired relative extent of each board ...)



```
(... omitted by authors: preceding code was unchanged ...)
pieces.push(colorize([0, 0, 1], translate([0, -depth/2 + thickness/2, 0],
    cuboid({size: [width, thickness, innerHeight]})))) // back
pieces.push(colorize([1, 1, 0], translate([-width/2 + thickness/2, thickness/2, 0],
    cuboid({size: [thickness, innerDepth, innerHeight]})))) // left
pieces.push(colorize([0, 1, 1], translate([width/2 - thickness/2, thickness/2, 0],
    cuboid({size: [thickness, innerDepth, innerHeight]})))) // right
pieces.push(colorize([1, 0, 1], translate([0, 0, 0],
    cuboid({size: [innerWidth, innerDepth, thickness]})))) // shelf
(... omitted by authors: subsequent code was unchanged ...)
```

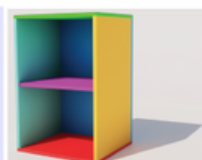


Figure 1

Large language model's (LLM) cabinet design process. Simple cabinet design, from initial prompt to the final result, in which all elements have the correct orientation and no components overlap. The LLM reaches the final result in eight messages (four each of prompt/response).

To test the limits of LLM-driven design, we asked the LLM to design a functional quadcopter, which requires the integration of prebuilt elements such as motors, propellers, and batteries. After sourcing these components, the challenge lies in designing a frame that accommodates their dimensions. We assigned this task to an LLM, focusing primarily on the frame design to securely hold the selected components. The initial frame design proposed by the LLM was impractical, as it was directly attached to the motor cylinder and lacked adequate support for components like the battery, controller, and signal receiver. With minor adjustments, the LLM refined the design to a practical version. This final design was then subjected to further evaluation in a simulator or under real-world conditions, as illustrated in [Figure 2](#).



Figure 2

A quadcopter designed with the aid of a large language model. First row: CAD model (motors - red, propellers—yellow, battery—dark gray, frame—blue, controller—dark yellow, receiver—green box), frame model, close-up of the frame with motor connectors. Second row: sourced parts, manufactured frame, assembled quadcopter.

Although the LLM sometimes produces flawed initial designs, it effectively corrects errors after a few user interactions. This capability for iterative design is particularly beneficial for constructing complex structures. Users can begin with a simple prompt and gradually increase complexity to achieve the desired outcome. The LLM excels in incorporating modules and hierarchical structures through natural language, showcasing its skill in managing high-level structure and discrete composition. It reliably creates accurate primitives in various design languages. Additionally, the LLM’s output is characterized by its readability and maintainability. The generated code features descriptive variable names, helpful comments, and proper modularity. Users also have the option to request custom hierarchical refactoring, which aligns well with the preferences of human programmers and designers.

2.2 Design Spaces via Natural Language and Extrapolation

A design space encompasses a range of potential designs. A common approach to creating a design space is through parametric designs, where design parameters may be continuous or discrete. However, having a parametric design alone is insufficient for exploring different design variations, either manually or automatically. It is essential to determine specific values for the design parameters. This can be achieved by setting lower and upper bounds for each parameter, allowing each to take any value within these limits. A design space, therefore, is defined by the parametric design combined with these parameter bounds, representing the entire set of possible design variations.

The initial step in creating a design space is to verify the LLM's ability to generate parametric designs. To ensure the generation of good parametric designs, we instruct the LLM to use high-level design parameters and minimize the number of variables. Interestingly, the LLM often introduces additional variables for readability on its own initiative, even without specific instructions. We have observed that incorporating these guidelines into our prompts consistently leads to the creation of parametric designs. Alongside this, establishing parameter bounds is crucial for defining a design space. When prompted for lower and upper bounds, the LLM suggests values based on the typical proportions of the object being designed. Although the absolute scale is arbitrary, the proposed bounds are semantically reasonable and proportionate to each other.

We also explored the LLM's capacity to infer a parametric design space from a specific preexisting design provided by the user. The input designs for the LLM are provided in a text-based DSL such as OpenJSCAD, with varying degrees of semantic annotations and explanatory comments. Enriching LLM inputs with more semantic information noticeably enhances the quality of the resulting design space. For instance, including the name of the object being modeled in the design proves beneficial for generating a parametric design. This approach reveals design parameters that are semantically more relevant for altering the design, as illustrated in [Figure 3](#). We find that the LLM's extensive semantic knowledge base can be utilized to generate parameters, bounds, and constraints not only for text-based designs but also for preexisting designs, as shown in [Figure 4](#).

Can you expose high-level design parameters for this openscad design? Use as few variables as possible

```
union() {
  difference () {
    cube([19, 19, 15 + 2]);
    (... omitted by authors: intermediate construction operations ...)
    for (i = [0:5-1])
    (... omitted by authors: rest design ...)
```



// High-level design parameters

```
cube_size = 19;
cube_height = 15;
extra_height = 2;
num_layers = 5;
// Other parameters ...
union() {
  difference () {
    cube([cube_size, cube_size, cube_height + extra_height]);
    (... omitted by authors: intermediate construction operations ...)
    for (i = [0: num_layers-1])
    (... omitted by authors: rest design ...)
```

Can you expose high-level design parameters for this openscad design of a chair with slats? Use as few variables as possible

```
length = 19; // Length of the chair
width = 19; // Width of the chair
height = 15; // Height of the chair base
slat_count = 5; // Number of slats
union() {
  difference () {
    cube([length, width, height + thickness]);
    (... omitted by authors: intermediate construction operations ...)
    for (i = [0: slat_count -1])
    (... omitted by authors: rest design ...)
```

Figure 3

Semantic cues for chair design parameters. When asked to parametrize a chair design, the large language model gives different parametrization with or without context cues.



Can you give bounds for each design parameter?



Bounds for each parameter could be something like this:

handle_thickness: This parameter could range from 0.5 (for a thin handle) to 3 (for a thick handle). So the bounds are [0.5, 3].
 (... omitted by authors: explanation of other bounds ...)
 Note that these bounds are somewhat arbitrary and based on common mug sizes. You could adjust them based on your specific needs or the scale of your 3D printing.

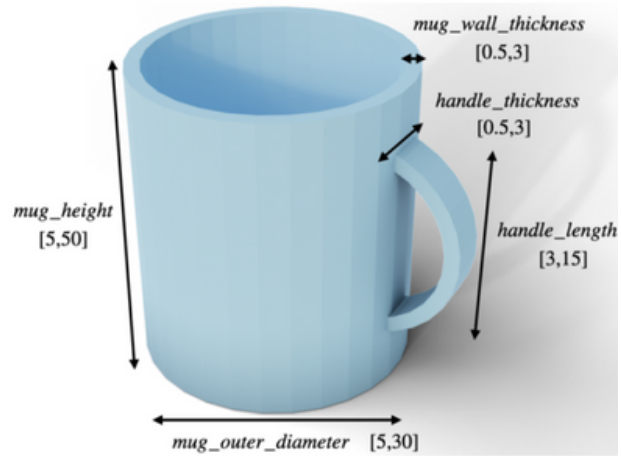


Figure 4

Mug design parameter bounds and constraints. The large language model gives parameter bounds based on common mug sizes.

Design spaces derived from a single design enable the exploration of various shapes through parameter adjustment. However, for more substantial structural modifications, designers might consider combining elements from two or more designs within the same object class. This process of design interpolation presents its own challenges. To test the LLM’s capabilities in this area, we presented it with designs of a bicycle and a quad bike, which differ in wheel number and fork construction. The bicycle has a fork encasing the wheel, while the quad bike’s wheels attach to the frame with horizontal bars. Tasked with designing a tricycle, the LLM successfully determined the appropriate wheel arrangement and adapted the quad bike’s frame for wheel alignment. However, it did not fully incorporate the bicycle’s fork design, suggesting room for improvement with additional user guidance ([Figure 5](#)). This experiment highlights the LLM’s potential in merging design concepts and coding proficiency. We observe that the LLM can interpolate existing designs by extracting and adapting subdesigns based on their program representations. Interestingly, even when designs are not presented in a modular fashion, it tries to recognize and abstract submodules in input designs.

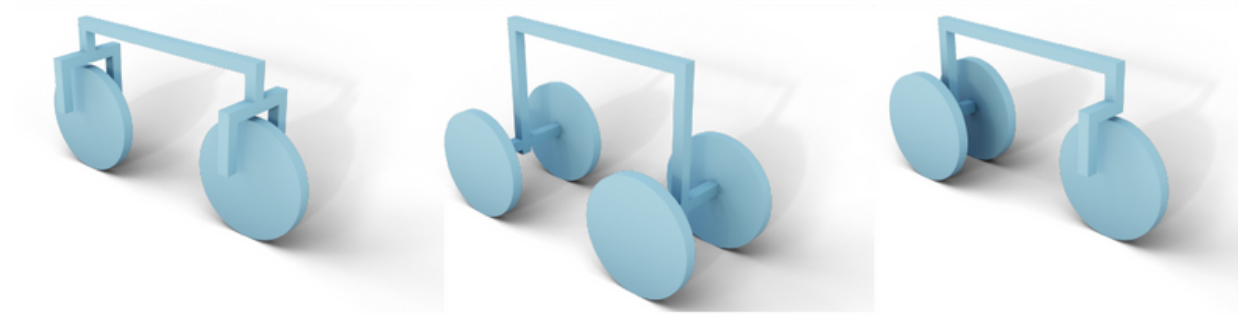


Figure 5

Bike design interpolation. *Left:* Input bicycle design. *Middle:* Input quad-bike design. *Right:* Interpolated output tricycle design.

Design spaces are crucial for creating design variations, yet determining significant variation parameters can be an iterative and labor-intensive process. To test the LLM’s effectiveness in this area, we conducted an experiment. We provided the LLM with a parametric design of a Lego brick and tasked it to define parameter bounds and constraints. Subsequently, we asked it to generate ten different parameter settings for viable Lego bricks, each with a unique name, as illustrated in [Figure 6](#). The LLM successfully adhered to the established bounds and constraints, resulting in distinct 3D models with plausible names. These outcomes suggest that LLMs can efficiently identify semantically meaningful design variations within a given design space.

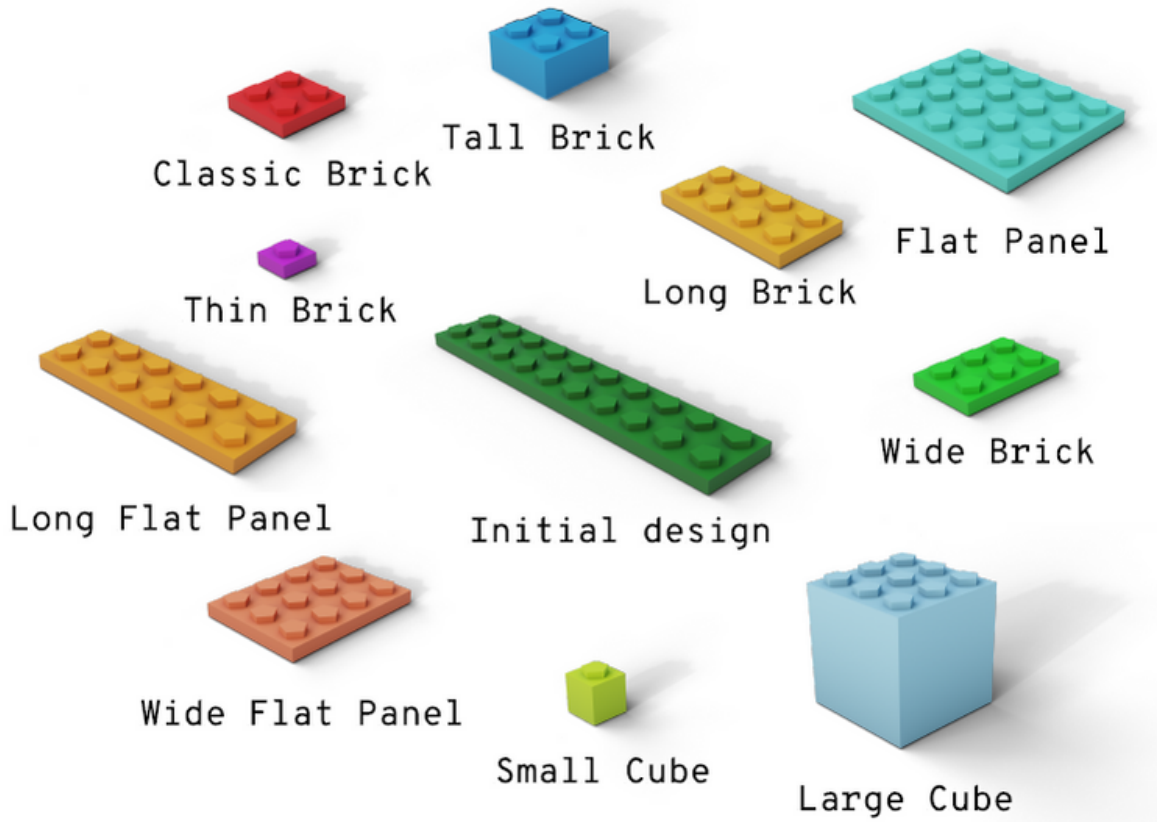


Figure 6

Lego design space exploration. The large language model (LLM) generates ten different design variations for the parametric Initial design. The label under each model corresponds to the name given by the LLM.

2.3 Design for Manufacturing

LLMs have a range of applications in design for manufacturing (DfM), offering potential enhancements in the design and production of parts and assemblies. They serve as a repository of manufacturing expertise, leveraging their pattern recognition and language interpretation capabilities during the design and manufacturing phases. Additionally, LLMs' programming skills and text pattern analysis enable them to generate and modify design and manufacturing files. Traditionally, DfM has depended on human expertise and CAD software to review and adapt designs for better manufacturability. By integrating LLMs into this workflow, the DfM process could become more streamlined, leading to more consistent, scalable, and efficient decision-making processes.

We outline several ways in which this new manufacturing expertise bank, as represented by LLMs, can be applied in design and manufacturing (refer to [Figure 7](#)). First, the LLM can be used to identify the most suitable manufacturing techniques for a part, based on its specific features. Additionally, it can suggest and apply design modifications to enhance manufacturability, thereby streamlining production processes. This

concept can also be expanded to include part sourcing, where the model’s reasoning abilities are used to select potential suppliers, taking into account the part’s intended function and performance. Lastly, the LLM could assist in creating manufacturing instructions for various processes.

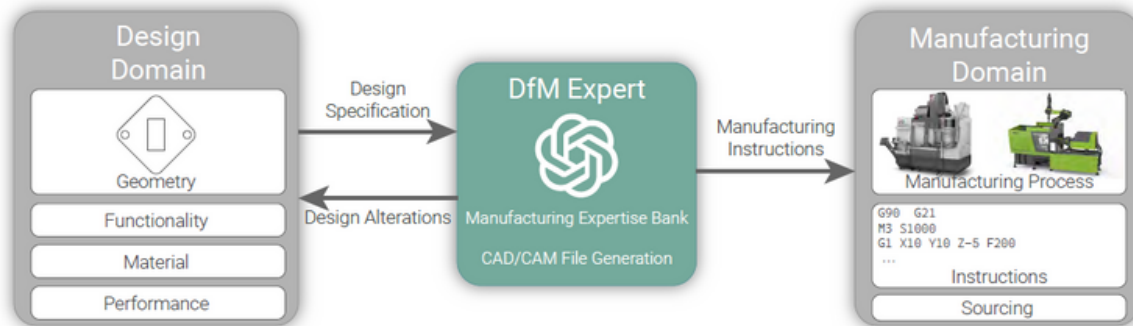


Figure 7

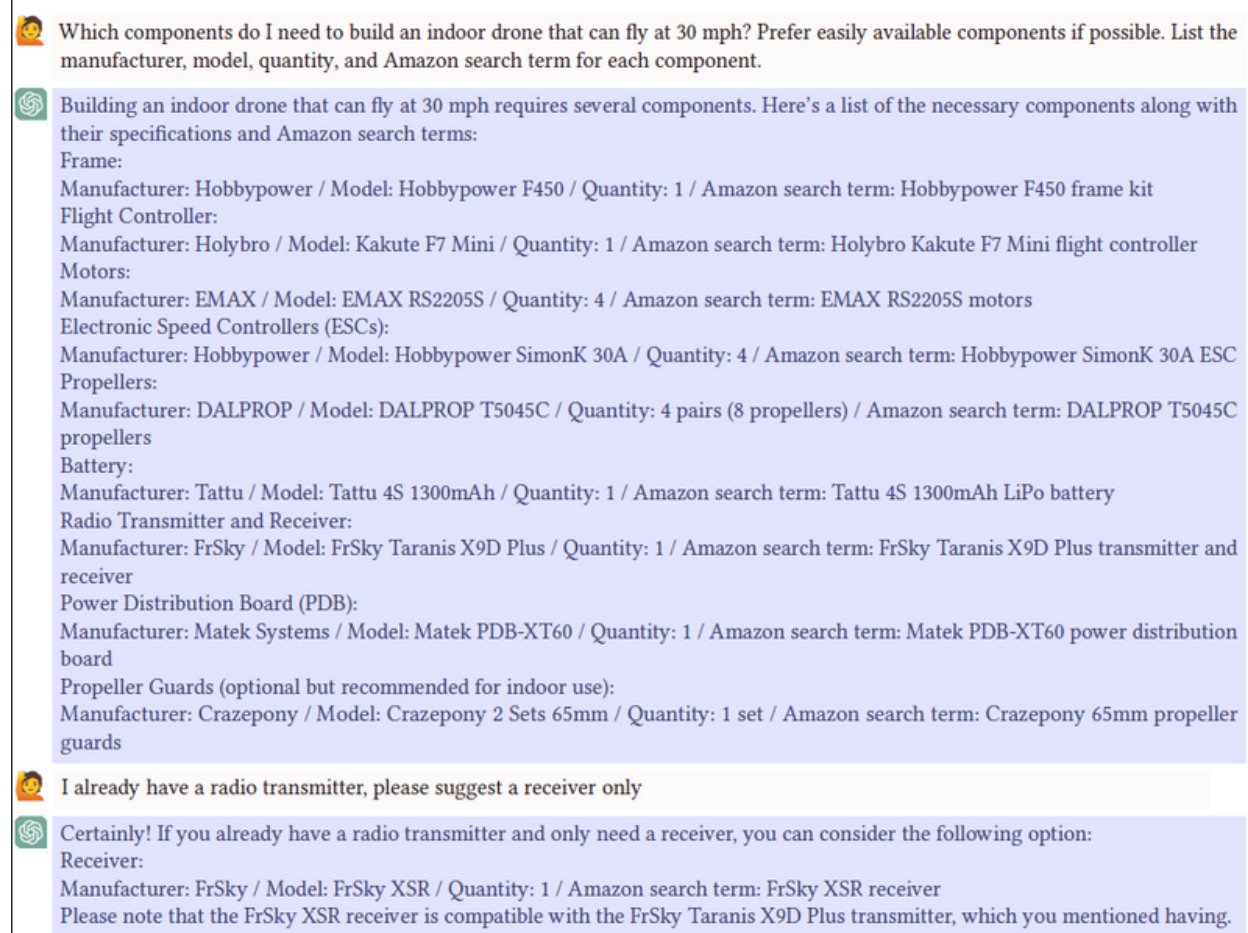
Integration of a large language model (LLM) into design for manufacturing (DfM): An LLM can be used to augment the DfM Process when designing a part.


To assess the LLM’s proficiency in selecting the best manufacturing processes, we conducted a test involving four different cases. Each case featured variations in geometry, material, tolerance, and quantity, described using an OpenJSCAD file. The LLM’s task was to determine the most suitable manufacturing process based on predefined priorities. In some cases, we intentionally limited the options to evaluate the LLM’s decision-making under constraints. The objective was to gauge the accuracy of the LLM in aligning its choices with the specified priorities. In three of the cases, the LLM’s selections were deemed optimal by an expert reviewer. However, in the remaining case, its recommended process was inappropriate for the given material. Initially, the LLM presented a range of options, necessitating additional prompts to identify the most appropriate choice.

We evaluated the LLM’s ability to optimize manufacturing designs by inputting a CNC machining project into the LLM using an OpenJSCAD file. The project involved a 10-inch diameter disk with bolt holes and a central blind square pocket. The LLM identified manufacturing complexities and adjusted the OpenJSCAD specification to address them. Although the LLM initially misunderstood some geometric features, it correctly identified several issues. Following corrections, the LLM addressed the concerns with the part’s size and machining area. In the final phase of the project, the LLM improved the wall thickness and adjusted the radius of the internal pocket for more efficient machining with a 1/4" endmill.

By drawing on extensive datasets, LLMs possess specialized knowledge in part manufacturing, which enables them to accurately identify part names and describe their function. In designing a quadcopter, we initially requested a comprehensive parts list from the LLM, including items like batteries, frames, and propellers (Figure 8). The response was both thorough and precise. Seeking further details, we asked for numerical specifications for components like batteries, and the LLM provided realistic estimates. Depending on the design requirements, whether for a heavy-duty copter or an indoor model, the LLM recommended appropriate

parts. However, specific requests for particular part names or manufacturers sometimes led to incompatible or nonexistent part suggestions. Through iterative questioning, the accuracy of these suggestions improved. Although the LLM faced challenges in generating precise, fully compatible parts lists, it excelled in offering broad, relevant suggestions and advice on part compatibility. The LLM proved valuable for domain-specific knowledge and in creating detailed parts lists, but it remains prudent to verify exact numbers and specifications independently.



 Which components do I need to build an indoor drone that can fly at 30 mph? Prefer easily available components if possible. List the manufacturer, model, quantity, and Amazon search term for each component.

 Building an indoor drone that can fly at 30 mph requires several components. Here's a list of the necessary components along with their specifications and Amazon search terms:

Frame:
 Manufacturer: Hobbypower / Model: Hobbypower F450 / Quantity: 1 / Amazon search term: Hobbypower F450 frame kit

Flight Controller:
 Manufacturer: Holybro / Model: Kakute F7 Mini / Quantity: 1 / Amazon search term: Holybro Kakute F7 Mini flight controller

Motors:
 Manufacturer: EMAX / Model: EMAX RS2205S / Quantity: 4 / Amazon search term: EMAX RS2205S motors

Electronic Speed Controllers (ESCs):
 Manufacturer: Hobbypower / Model: Hobbypower SimonK 30A / Quantity: 4 / Amazon search term: Hobbypower SimonK 30A ESC


Propellers:
 Manufacturer: DALPROP / Model: DALPROP T5045C / Quantity: 4 pairs (8 propellers) / Amazon search term: DALPROP T5045C propellers


Battery:
 Manufacturer: Tattu / Model: Tattu 4S 1300mAh / Quantity: 1 / Amazon search term: Tattu 4S 1300mAh LiPo battery

Radio Transmitter and Receiver:
 Manufacturer: FrSky / Model: FrSky Taranis X9D Plus / Quantity: 1 / Amazon search term: FrSky Taranis X9D Plus transmitter and receiver

Power Distribution Board (PDB):
 Manufacturer: Matek Systems / Model: Matek PDB-XT60 / Quantity: 1 / Amazon search term: Matek PDB-XT60 power distribution board

Propeller Guards (optional but recommended for indoor use):
 Manufacturer: Crazepony / Model: Crazepony 2 Sets 65mm / Quantity: 1 set / Amazon search term: Crazepony 65mm propeller guards

 I already have a radio transmitter, please suggest a receiver only

 Certainly! If you already have a radio transmitter and only need a receiver, you can consider the following option:

Receiver:
 Manufacturer: FrSky / Model: FrSky XSR / Quantity: 1 / Amazon search term: FrSky XSR receiver

Please note that the FrSky XSR receiver is compatible with the FrSky Taranis X9D Plus transmitter, which you mentioned having.


Figure 8

We illustrate the use of the large language model (LLM) to generate a parts list for a copter from Amazon. While the LLM successfully identified specific parts, it also made factual errors. For instance, it included Crazepony propeller guards in the list, despite Crazepony not selling such items. After iterative refinement, our final list of ten parts contained four major inaccuracies. Two of these inaccuracies were parts attributed to manufacturers that didn't actually offer them. Another error was a redundancy: the recommended flight controller kit already included a power distribution board, rendering a separate purchase unnecessary. The fourth error involved a misidentified part: an item the LLM classified as both a transmitter and receiver was, in fact, only a transmitter. A subsequent correction led the LLM to accurately recommend an appropriate receiver. After addressing these errors, we placed an order based on the modified parts list.


Computer-aided manufacturing (CAM) employs software to transform digital designs into precise manufacturing instructions, streamlining the production process. It serves as a crucial link between digital design and physical manufacturing by automating tasks such as planning and toolpath generation. CAM enhances production efficiency, accuracy, and communication. In this context, we explored the use of LLMs in conjunction with open source CAM software to create both machine-readable and human-readable manufacturing instructions.


Additive design, essential in 3D printing, typically involves complex spatial reasoning and multiple design iterations. We posit that LLMs can simplify this process by comprehending intricate specifications, creating designs, simulating results, and devising innovative solutions that improve both functionality and aesthetics. Our method consists of two phases. Initially, we translate the concept from natural language into an intermediate 3D shape using triangle meshes. Subsequently, this shape is converted into G-code, suitable for specific 3D printing hardware. This conversion, requiring deep knowledge of fabrication processes, is facilitated using Slic3r,⁴ a professional G-code generation software, integrated with Python for optimal G-code output. The LLM plays a crucial role in ensuring effective communication throughout these stages, fostering a smooth flow of data and instructions. This LLM-augmented process aims to enhance the efficiency and quality of additive manufacturing outcomes.

We also conducted an experiment to explore the potential of LLMs for generating assembly instructions that are human and machine-readable as standard operating procedures. Our experiment involved assembling a wooden box, using a specified set of tools and materials. As shown in Figure 9, we asked the LLM to generate human and machine-readable instructions, which involves creating a set of functions to specify different tasks and generating corresponding sequences to execute those tasks. The LLM's output details system-agnostic actions for a robot to perform. We then instructed the LLM to convert these machine-readable instructions into a comprehensive, human-readable standard operating procedure. This procedure offers a step-by-step guide to the assembly process, making it easy for human operators to understand and follow. The aim was to evaluate the LLM's ability to bridge communication between robots and humans in assembly tasks, by generating instructions that cater to both.

 Create a Python script for a household robot that assembles a wooden box with specific dimensions (5x5x5 inches with a thickness of 0.5 inches). The robot has to perform a set of operations including applying wood glue along certain lines, clamping the pieces together in a specific order, drilling a hole in the center of the box with a 1/8 inch drill bit, and finally placing a #8 size, 1.5 inch long wood screw into the drilled hole. The robot should be able to move to a specific location before each operation and should print a statement describing the action it has performed, including the coordinates where the action took place. The operations should be ordered as follows: glue and clamp the bottom panel, left and right sides, back, and finally the top. Then, it should change the drill bit, drill a hole in the center of the box, change the screw, and place the screw in the hole. Specify the tools in the script including 'wood glue', 'power drill with 1/8 inch drill bit', 'bar clamp', and '#8 size, 1.5 inch long wood screw'.

```
# The dimensions of the panels
width, height, depth, thickness = 5, 5, 5, 0.5
# Function to apply glue
def apply_glue( start , end):
    # Move to start point
    go_to_location( start )
    pick_up(' wood glue')
    # Apply glue while moving to end point
    go_to_location(end)
    use_item(' wood glue')
    return "Wood glue applied along line from ({}, {}, {}) to ({}, {}, {})".format(* start , *end)
...
# Gluing and clamping the panels in the correct order
print(apply_glue((0, 0, 0), (width, 0, 0))) # Bottom Panel
print(place_clamp(width/2, 0, 0)) # Bottom Panel
print(apply_glue((0, 0, 0), (0, height, 0))) # Side Panel (Left)
print(place_clamp(0, height/2, 0)) # Side Panel (Left)
print(apply_glue((0, height, 0), (width, height, 0))) # Side Panel (Right)
print(place_clamp(width, height/2, 0)) # Side Panel (Right)
print(apply_glue((width, 0, 0), (width, height, 0))) # Back Panel
print(place_clamp(width/2, height, 0)) # Back Panel
print(apply_glue((0, 0, depth), (width, 0, depth))) # Top Panel
print(place_clamp(width/2, 0, depth)) # Top Panel
# Drilling holes and placing screws
change_drill_bit('1/8 inch drill bit')
print( drill (width/2, height/2, thickness)) # Center of the box
change_screw('#8 size , 1.5 inch long wood screw')
print(place_screw(width/2, height/2, thickness)) # Center of the box
```

 Can you convert these instructions into a human-readable form as a standard operating procedure?

 **Preparation:**

...

Gluing and Assembly of Panels:

Applying Glue: Apply wood glue in a thin, consistent line along the edges where the panels will join. This includes the edges of the bottom panel and both side panels.

Assembly: Start by placing the bottom panel flat on your work surface. Align each side panel perpendicularly with the edges of the bottom panel, creating a U shape. Make sure the corners are square and the edges flush. Then, apply glue to the edges of the back panel and align it with the bottom and side panels. Finally, apply glue to the top panel and align it with the side panels and back panel.

Clamping: After each panel is placed, secure it with a bar clamp. Ensure clamps are tight, but not too tight to avoid bowing the wood. Leave the glue to dry according to the manufacturer's instructions.

Drilling Pilot Holes:

Drill Bit Installation: Replace the existing bit in your power drill with a 1/8 inch drill bit.

Drilling: Drill pilot holes at all eight corners where the panels join together. The pilot holes should be centered on the edge and go through both pieces of wood. This will help prevent the wood from splitting when the screws are inserted.

Screwing in Screws:

Screw Selection: Use #8 size, 1.5-inch long wood screws. These screws are robust enough to secure the box structure.

Screwing: Screw them into the pilot holes at each corner to secure the box structure. Make sure each screw is flush with or slightly recessed into the wood surface.

...

Figure 9

Instructions for assembling a wooden box using specific tools and dimensions. This figure shows a Python script for a household robot that assembles a wooden box with specific dimensions. The robot follows a sequence of operations using tools such as wood glue, a power drill with a 1/8 inch bit, a bar clamp, and a #8 size, 1.5-inch-long wood screw. The second part shows the corresponding human-readable instructions.

2.4 Design to Performance

Evaluating a design typically involves analyzing performance metrics related to its geometry and materials, encompassing aspects like mechanical performance, dynamic functionality, and geometric compliance. The evaluation can focus on a single criterion or multiple criteria to determine if the design meets specific requirements. Results of this assessment can range from a single number to a spectrum of outcomes. More complex evaluations involve comparing different designs for optimization or final production decisions. These evaluations include objective, semi-subjective, and subjective criteria. Objective criteria are measurable features such as weight, size, load capacity, impact resistance, speed, battery life, vibration resistance, and price. Semisubjective criteria require some level of insight or estimation, including ergonomics, product lifespan, sustainability, portability, safety, and accessibility, which may vary based on the evaluator or context. Subjective criteria depend heavily on the evaluator's perspective, covering aspects like comfort, aesthetics, customer satisfaction, novelty, and perceived value.

In the engineering design process, after creating a design or design space, key geometric features like size, weight, and strength are typically evaluated to ensure the design's functionality. We demonstrate this process using a quadcopter as an example. We input the quadcopter's specifications into the LLM, including battery details, weight, and dimensions. The LLM is then tasked with generating functions to assess hover time, travel distance, and velocity/acceleration. It uses these inputs to calculate the inertia tensor, voltage–torque relationships, and other kinematic and dynamic factors. Furthermore, the LLM determines essential physical parameters for these calculations, such as maximum thrust, battery capacity, aerodynamic properties, control system efficiency, payload, environmental conditions, and operational constraints. This case study not only illustrates the LLM's capability in conducting detailed evaluations but also highlights its ability to recommend appropriate products, such as specific batteries from a particular vendor, for the quadcopter.

The LLM is capable of suggesting design considerations and metrics with a high degree of sophistication. It remains effective even when handling ambiguous requests, missing details, or complex performance metrics, consistently producing reasonable first-order approximation functions. Typically, the evaluation functions generated by the LLM are functional and free of coding errors. The LLM has demonstrated the ability to assess every property we tested. In instances where errors occur, additional questioning and interaction help refine the code, leading to improvements in the output.

The evaluation of subjective properties, which heavily rely on lexical input, presents an interesting challenge for LLMs. To test the LLM's ability to assess subjective criteria based on abstract input parameters, we tasked it with creating a list of key sustainability criteria and then evaluating chair designs against these criteria, scoring each category from one to ten. In this test case, we fed the LLM text from product pages, including each chair's name, design description, material, appearance, and some quantitative data like dimensions, weight, and cost, as shown in [Figure 10](#). When asked to provide sustainability scores, the LLM generated reasonable numbers, offering justifications based on the text descriptions. Additionally, the LLM demonstrated

its capacity to categorize items and rank a set of designs, even without providing explicit intermediate evaluations.

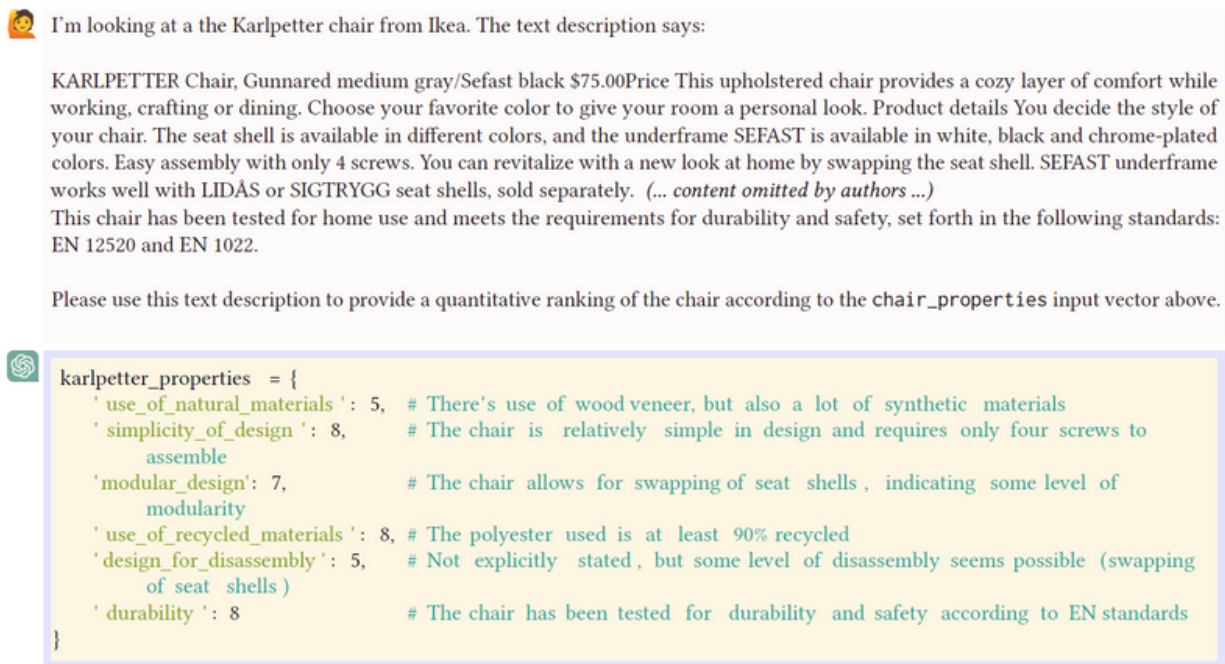


Figure 10

Evaluation of chair aesthetics. Evaluation by the large language model to score chairs based on a text description of the chair based on metrics of sustainability.

2.5 Inverse Design

Although generative AI algorithms can produce design candidates, they do not inherently assure high performance for these candidates. Inverse design aims to create designs that are as close to optimal as possible within certain constraints. Specifically, given a design space and performance metrics (which may include optimization targets or constraints), inverse design addresses this question: Which design within our space offers the best performance without breaching any constraints?

While LLMs are not inherently capable of independently searching for optimal solutions to novel problems, they can provide educated initial guesses and generate optimization code for users to execute. Consequently, we concentrated our efforts on using LLMs to create effective code for problems, taking into account factors like parameterization support (e.g., continuous vs. discrete domains), the nature of performance objectives, and fabrication constraints. In this scenario, we explore the potential of LLMs in proposing strategies that could alleviate some of the common challenges engineers face in the optimization process.

We explore whether the LLM can generate a practical cost function and an optimized design when given an example design, a parameterized design space, and a textual objective description. An example of this

application is in furniture design: Can an LLM optimize a cabinet design to achieve a specific volume while minimizing construction costs? Initially, we provide the LLM with a parametric cabinet design in OpenJSCAD, including parameter bounds. We then instruct it to create functions for calculating the cabinet's volume and material costs. After confirming the accuracy of these functions, the LLM is tasked to produce a Python script. This script is designed to minimize material costs while adhering to the specified volume constraint. The script and its results, including renders of the optimized cabinet, are displayed in [Figure 11](#) and [Figure 12](#).

**Figure 11**

Optimizing material cost of cabinet with respect to volume. The performance and cost functions are generated by prompting the large language model (LLM). To find the minimum of the material cost function with respect to volume, the LLM calls the minimize function (scipy.optimize) with an appropriate solver. In addition, the provided code extracts the relevant bounds and a suitable initial guess from the prompt.

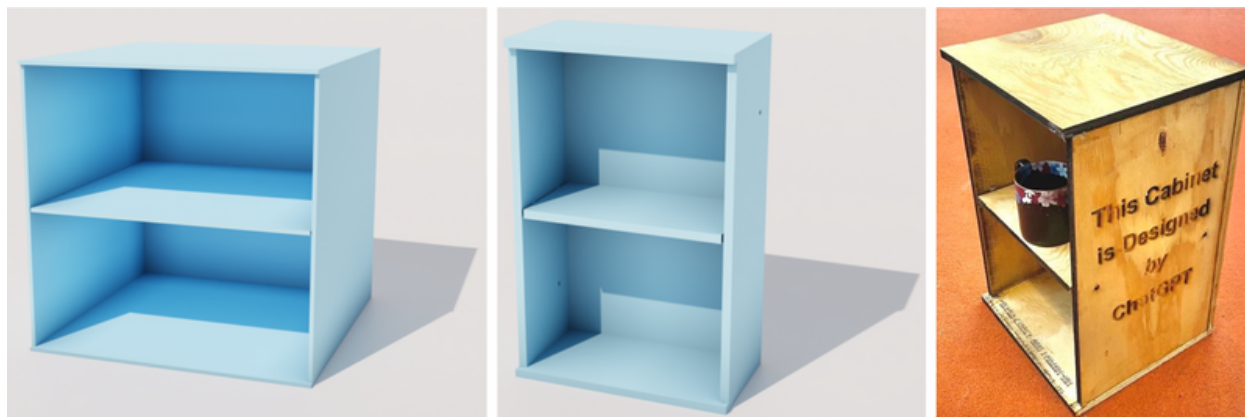


Figure 12

Single objective with constraint optimization. *Left:* A cabinet with initial height, width, and depth of 55 in. *Middle:* Using an off-the-shelf solver provided by `scipy.optimize`, the large language model (LLM) provides a Python script that optimizes the cabinet's height, width, and depth given a desired volume while minimizing the material cost. We show an optimized cabinet that meets the volume constraint of 5,000 cubic inches of storage capacity. *Right:* A manufactured variant of the cabinet design, which was generated by asking the LLM to convert the CAD model into a set of primitives laid out in a valid, laser-cuttable Drawing Exchange Format (DXF) file.

The LLM shows notable proficiency in generating design spaces, objectives, and constraints, proving to be a valuable resource in inverse design systems. Its strengths include selecting suitable design optimization algorithms for a variety of problems and rationalizing these choices. Additionally, LLMs can independently generate code for critical aspects of problem formulation, such as choosing parameters, establishing parameter ranges, and defining objective functions. This capability significantly reduces the routine workload in setting up problems, including establishing loose bounds and essential constraints. Even in less-than-ideal scenarios, the LLM serves as a useful aid, providing a robust foundation for tackling inverse problems.

3. Gaps in Using Generative AI for Design and Manufacturing and Potential Solutions

Generative AI techniques have the potential to establish efficient design and manufacturing workflows. However, after thoroughly assessing the current capabilities of LLMs across various tests, we have also identified several limitations. In this section, we outline these key constraints and offer potential solutions to address them.

Reasoning Challenges: LLMs struggle with analytical reasoning and complex computations, which are crucial in design and manufacturing. This often leads to challenges like limited spatial reasoning capabilities. One way to mitigate these issues is through the implementation of domain-specific languages. Common in computer science, DSLs encapsulate recurring knowledge, rules, and abstractions, effectively bridging knowledge gaps. It would also be helpful to leverage APIs capable of handling complex computations. LLMs' ability to create

high-level abstractions can be harnessed to generate inputs for these APIs, allowing computational solvers to process them effectively.

Correctness and Verification: LLMs sometimes generate inaccurate results or provide inadequate justifications for their solutions, and they lack self-verification capabilities. To address this beyond human oversight, automated verification can be implemented using APIs that perform checks and validations. Utilizing LLMs' iterative strengths, it may be beneficial to establish a feedback loop that continues until a satisfactory solution is achieved.

Scalability: When faced with larger or more intricate tasks, LLMs' performance can deteriorate, often due to difficulties in handling multiple tasks at once. To overcome this, it is effective to decompose large tasks into smaller, manageable subtasks. For example, instead of asking the tool to assess multiple performance metrics at once, it is more practical to evaluate them separately. To construct complex models, it is advantageous to adopt an incremental design approach in which each component is individually designed and verified before integrating them into the final model. LLMs' ability to support modularity can be leveraged to construct complex models step-by-step, following a series of instructions.

Iterative Editing: When modifications are required in a design, specifying these changes directly to the LLM often leads to unsatisfactory results. The issue arises because the LLM, upon receiving a prompt for changes, tends to regenerate the entire design, neglecting details from previous prompts. This complicates the process of editing designs. To address this, one approach is to incorporate all prompts into a single, comprehensive specification, though this method can be problematic due to the scalability issues mentioned earlier. An alternative solution is to adopt explicit modularization, allowing for individual parts to be designed and reused separately, which can lead to more effective and manageable design changes.

Context Information: LLMs are heavily reliant on the presence of detailed context information, and its performance dramatically improves with the addition of more comprehensive domain descriptions. The LLM's ability to provide context for its actions makes it particularly valuable in sequential workflows. This feature is advantageous when integrating LLM-generated content into subsequent tasks, as these tasks can leverage the contextual information provided in the outputs from earlier tasks. However, there are potential downsides: insufficient context provision by the user, or the user's desire to create unconventional designs, can lead to challenges if the LLM is unable to move beyond its inherent biases related to a specific domain.

Unprompted Responses: LLMs frequently infer unspecified details in prompts, either by autocompleting specifications or making decisions based on limited information. This capability is intriguing in design contexts, as it allows for the completion of partial specifications. However, this proactivity can sometimes be excessive, steering the design process in certain directions and potentially limiting creativity.

4. Ethical and Dual-Use Risks and Mitigation Strategies

The use of LLMs in the field of design and manufacturing offers significant benefits in terms of efficiency, precision, and innovation. However, it also raises several ethical and dual-use risks.

Job Displacement: Development of software copilots for design and manufacturing can lead to job displacement, impacting the livelihoods of professionals in these fields. Tasks that once required human expertise, such as drafting, modeling, analysis, and preparing design documentation, will increasingly be automated using copilots. This change could potentially decrease the demand for skilled labor in design and manufacturing. We believe that a significant investment in education and training is necessary to adapt the existing workforce to using the new generative AI workflows.

Intellectual Property: Developing LLMs relies on existing data and designs. If training datasets include patented designs, their use could infringe existing intellectual property (IP) rights. This raises concerns about the originality of the generated output using these new tools. For example, generated designs might replicate elements of preexisting designs. Furthermore, the use of LLMs might raise questions about the ownership of the generated designs: is the owner the designer, the creator of the AI tools, or the creator of the designs that are used to develop LLMs? The use of design and manufacturing copilots could lead to challenges to the current IP law.

Dual-Use Nature and Potential Military Applications: Automated design and manufacturing tools greatly reduce the knowledge barrier for creating intricate systems, but they also pose potential risks when misused for creating prohibited items. These tools could expedite the development of advanced military assets, like precision weaponry and unmanned systems (e.g., drones and robots). The rapid production of such military equipment can offer substantial strategic advantages. Consequently, there is a pressing need for strong regulatory frameworks, international collaboration to deter misuse, and a strong emphasis on ethical considerations throughout the development and deployment of these technologies.

Design Data Privacy and Security. LLMs have the capability to replicate sensitive and proprietary design data used in model creation. Protecting the security and privacy of this data is a paramount concern, given its highly sensitive nature. Unauthorized access to such data can result in substantial competitive disadvantages and financial losses. Additionally, if the training data is compromised or tampered with, it could potentially result in flawed new designs and manufacturing processes. As LLM systems gain popularity, addressing privacy and security issues associated with them becomes increasingly vital. This necessitates the development of a comprehensive strategy to implement robust cybersecurity measures.

Design Reliability and Accountability: Utilizing LLM tools for design and manufacturing introduces the possibility of products with design or safety deficiencies. Consequently, it is crucial not to place undue trust in these tools. Establishing an accountability process for designs and manufactured products is imperative. For

instance, when the tool operates as a copilot, human experts should retain the ultimate responsibility for verifying and approving the design.

Limited Access to Tools: Due to significant development expenses and limited training datasets, it is conceivable that generative AI tools for design and manufacturing may only be accessible to large enterprises. This scenario is expected to exacerbate the gap in design and manufacturing capabilities between small businesses and large corporations.

5. Conclusions

We find that LLMs possess numerous capabilities that can be leveraged within the domain of design and manufacturing. This area, where creativity converges with practicality, presents exciting opportunities for advancements that can potentially bring about a significant shift in the way we ideate, prototype, and manufacture a broad range of products. However, it is essential to recognize that substantial work remains to be done to fully support the integration of these tools within this field. A fundamental issue is that design for manufacturing involves a delicate balance between creativity and formal verification. Engineering design presents a paradox; it requires precision and exactness, yet thrives on an iterative and exploratory spirit. While our experiments have managed to circumvent the LLMs' limitations for formal reasoning through user guidance, DSL crafting, and external APIs for complex computations, there is still much to understand about how best to implement these strategies. Similarly, in terms of workflow development, we have observed a myriad of possibilities. One approach would divide problems into parts that can be tackled by LLMs and others that are best solved by traditional methods; another approach boasts iterative solutions; and yet another suggests a complete reframing of the problem by asking the LLM to generate problem-solving code. Each of these approaches carries potential advantages and disadvantages. So, what should an optimal framework look like? We hope our evaluation will aid others in formulating an answer to this question.

Our study also addresses ethical concerns and the dual risks associated with this technology, encompassing job displacement, intellectual property issues, potential military uses, data privacy and security, accountability, and restricted access to tools. We offer potential mitigation strategies for these risks.

In summary, we believe our analysis offers valuable insights into how LLMs can be harnessed in the domain of design and manufacturing. Although our work has made initial strides, the path to fully leverage the potential of these tools in this domain remains open, rich with opportunities for further exploration and innovation.

Bibliography

Makatura, Liane, Michael Foshey, Bohan Wang, Felix HahnLein, Pingchuan Ma, Bolei Deng, Megan Tjandrasuwita et al. "How Can Large Language Models Help Humans in Design and Manufacturing?" Preprint submitted July 25, 2023. <https://arxiv.org/abs/2307.14377>.

JSCAD Organization. “JSCAD User Guide.” <https://openjscad.xyz/dokuwiki/doku.php>. Accessed July 14, 2023.

OpenAI. “GPT-4 Technical Report.” Preprint submitted March 15, 2023. <https://arxiv.org/abs/2303.08774>.

Slic3r. “Open Source 3D Printing Toolbox.” <https://slic3r.org/>. Accessed July 20, 2023.

Footnotes

1.

The study presented in this report is a condensed summary of the earlier report by Makatura et al., “How Can Large

Language Models Help Humans in Design and Manufacturing?” [↵](#)

2. We use the OpenAI ChatGPT interface to interact with the GPT-4 versions released between May 24, 2023 and July 19, 2023. OpenAI, “GPT-4 Technical Report.” [↵](#)

3. JSCAD Organization, “JSCAD User Guide.” [↵](#)

4. Slic3r, “Open Source 3D Printing Toolbox.” [↵](#)