L-LLM: Large Language LEGO Models

Alex Wang,[1] Calvin Laughlin[1]

[1]Department of Computer Science, Stanford University
{alexjw3, calvin3}@stanford.edu

Stanford
Computer Science

Abstract

Our project introduces a multifaceted approach to generating novel LEGO instruction manuals in a text-based format. We leverage the vision capabilities of GPT-4o and fine-tune models such as GPT-3.5-turbo, Llama-2-7B-chat-hf, and Mistral-7B using a corpus of 90 existing text-based LEGO manuals. We detail our methodology, which includes fine-tuning these models on both existing and synthetically generated manuals from GPT-4o vision prompt engineering. Our contributions include a novel vision-to-text agent and the generation of new, small-scale LEGO instructions.

Using our custom dataset comprised of instructions, most human-created for Bricks for the Blind and some translated from PDFs, we finetune our models to generate instructions for simple LEGO builds such as cars, castles, houses, boats, and spaceships. Additionally, we parse through visual instruction sets native to the LEGO website and translate them into the text-based format, enhancing the Bricks for the Blind dataset with synthetic data.

For evaluation, we use two grading rubrics to score each generated build and instruction manual out of 100. GPT-4o evaluates the quality of instructions, while human scoring assesses the actual builds. We aim to highlight the creative potential of LLMs and their limitations in planning, creativity, and instruction. Results show that GPT-4o and fine-tuned LLaMa-2-7B have shown the most promise in novel instruction generation, but there is still much work do be done in planning and data gathering.

1. Data

We use human-created LEGO text instructions, originally created for the blind and visually impaired, giving us just 91 examples ranging from 87 to 16.4k characters per example.To begin, we manually separated the text-based instructions into different categories:

– `introduction`: A description of the visual and structural appearance of the LEGO set as well as its backstory and characters.

– `terms`: Contains common abbreviations for LEGO pieces (i.e. "Stud: the bump on a LEGO brick").

– `sorting`: Instructions for the assistants of the visually impaired on how to best situate the bricks for the builder's convenience.

– `instructions`: Detailed instructions walking the reader through the build.

– `misc`: Contains copyrights, advertisements, and general tips on how to assist the visually impaired with their build.

One example of our dataset in chatML format can be seen below, with 2,053 tokens and 7,419 characters

```
{"text": "<s>[INST] Generate LEGO instructions for building a model of the
iconic Tower Bridge, complete with its iconic towers, a working drawbridge,
and a red double-decker bus. [/INST] Book 1 - Bag 1: Large bag 1 contains
instructions for building the bridge.  \n- Step 1: Place a 32x16 blue flat
piece on the table to represent the Thames. \n- Step 2: Attach a 6x1 piece
                                        ...
Thank you for your patience and creativity! </s>"}
```

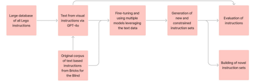
Large database of all Lego instructions
Text from visual instructions via GPT-4o
Fine-tuning and using multiple models leveraging the text data
Generation of new and constrained instruction sets
Evaluation of instructions
Original corpus of text based instructions from Bricks for the Blind
Building of novel instruction sets

Figure 1: Methods Flowchart

2. Methods

Mistral-7B

To fine-tune Mistral-7B, we used our custom training dataset and 4-bit quantization. We conducted the training on 2 H100 GPUs with gradient checkpointing and k-bit training. Our learning rate was 2.5 × 10−6, and we trained for 500 steps resulting in a training loss of **1.076** and evaluation loss of **0.883**. We additionally employed LoRA, which allowed for fine-tuning on a smaller subset of parameters

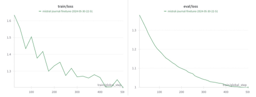

Figure 2: Mistral-7B Training & Eval. Loss

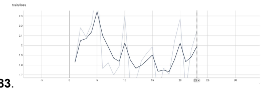

Figure 3: LLaMA-2-7B Training & Eval. Loss

LLaMA-2-7B-chat-hf

To finetune LLaMA-2-7B-chat-hf, we used Google Colab running an L4 GPU with 22.5 GB of RAM. We adapted code from a LLaMA fine tuning notebook created by Labonne (2024) to fit our needs and passed in our chatML formatted dataset. After 1 epoch with batch size 4 and an LR multiplier of 1, we report a loss of **1.937**.

GPT-3.5-turbo-1106

To finetune GPT-3.5-turbo-1106, we used the OpenAI platform and formatted our data into the required format that matches their Chat Completions API. We ran 2 separate finetune jobs with the `auto` hyperparameter selection, which in both finetune jobs defaulted to 3 epochs, batch size 1, and an LR multiplier of 2. Using our dataset containing 311,382 tokens, our loss was **1.359**.

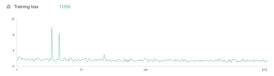

Figure 4: GPT-3.5-turbo Training Loss

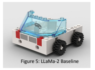

Figure 5: LLaMa-2 Baseline

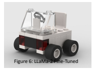

Figure 6: LLaMa-2 Fine-Tuned

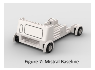

Figure 7: Mistral Baseline

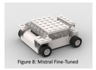

Figure 8: Mistral Fine-Tuned

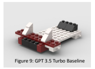

Figure 9: GPT 3.5 Turbo Baseline

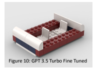

Figure 10: GPT 3.5 Turbo Fine Tuned

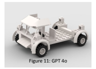

Figure 11: GPT 4o

3. Evaluation

Model	Structural Integrity (25 points)	Design Aesthetics (25 points)	Functionality (20 points)	Creativity (15 points)	Completeness (15 points)	Total (100 points)
Mistral-7B finetuned	23.1	23.1	17.6	10.8	12.9	87.5
Mistral-7B not finetuned	16.2	12.7	12.6	7.3	9.3	58.1
GPT-3.5 finetuned	17.3	7.1	5.6	9.1	4.4	43.4
GPT-3.5 not finetune	17.9	14.4	4.1	12.1	3.5	52.0
Llama-2 finetuned	20.9	20.5	18.0	13.8	13.9	87.1
Llama-2 not finetuned	21.3	18.3	18.6	13.8	11.4	83.4
GPT-4o baseline	21.1	18.5	17.3	13.2	7.7	77.8

Figure 12: LEGO Car Build Evaluation Scores

	Mistral-7B finetuned	Mistral-7B not finetuned	GPT-3.5 finetuned	GPT-3.5 not finetune	Llama-2 finetuned	Llama-2 not finetuned	GPT-4o baseline
Car average	65.33	59.00	36.75	80.33	55.67	86.67	80.33
Space ship average	63.67	73.00	37.00	82.16	36.0	85.33	83.33
House average	54.00	49.67	64.5	80.46	27.33	93.66	85.67
Boat average	62.33	71.00	54.0	79.33	37.0	87.33	87.67
Castle average	73.67	54.67	66.67	77.5	42.67	91.33	80.33
Total average	63.8	61.47	51.78	79.96	39.73	88.86	84.13

Figure 13: LEGO Car Instruction Evaluation Scores

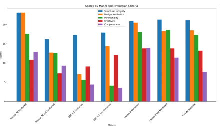

Figure 14: LEGO Car Build Evaluation Scores

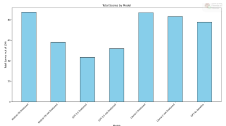

Figure 15: Total Scores of LEGO Car Builds

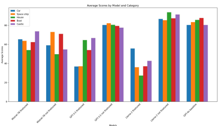

Figure 16: LEGO Instruction Evaluation Scores

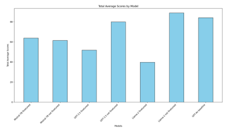

Figure 17: Total Scores of LEGO Instructions

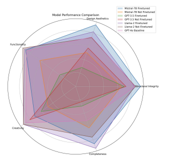

Figure 18: Build Radar Chart

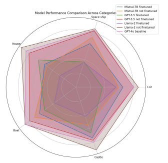

Figure 19: Instruction Radar Chart

4. Analysis

Creativity and Imagination
Models like Llama-2-7B and Mistral-7B excelled in producing creative instructions that introduced new and unique LEGO builds. While some of our fine-tuned models were outperformed in their instruction evaluation, their performance in the builds they created demonstrated creative leaps. However, we were unable to produce outputs that produced comprehensive builds that we had not seen before, i.e. with detail and complexity that was higher than elementary school-level creativity.

Handling Complexity
While our best models performed well on simpler builds, they struggled with generating more complex sets. They oftentimes lost sight of where they were in the process and where they needed to get to, repeating themselves or outputting nonsense. Overall, their ability to create a larger plan and then take the smaller steps to execute said plan was inadequate.

Efficiency of LoRA Fine-Tuning
The use of LoRA fine-tuning proved effective in enhancing model performance without requiring extensive computational resources. This approach allowed for targeted improvements in instruction generation and reduced the cost of our training.

Piece Identification Errors
All of our models misidentified LEGO pieces to some extent, providing incorrect LEGO piece codes while detailing piece requirements as well as within the instructions themselves.

Physical Placement
Our models performed surprisingly well in the realm of understanding physical placement and location. Their detailing of where pieces needed to be placed in relation to cardinal directions and within the context of what had already been built was impressive in the smaller LEGO generations.

Evaluation
Our evaluation methods were unique to our project and developed for our specific purposes. However, it was difficult for us to stray away from the gold standard of human evaluation, and felt that human evaluation was the most trustworthy causing us to validate all other outputs by our GPT-4o evaluations.

5. Conclusion

Main Findings

We achieved superior performance in creating LEGO builds with most of our fine-tuned models compared to their baselines. Llama-2-7B and Mistral-7B stood out for generating creative and unique builds, embodying the playful spirit of LEGO.

However, our fine-tuned models often overfitted, leading to poorer text-based instruction evaluations compared to baselines, except for Mistral-7B. A limited and noisy dataset affected overall performance and generalizability.

From this project, we learned the importance of high-quality, diverse, and sufficient training data for improving model performance and generalization. Balancing model complexity with task-specific requirements is crucial to avoid overfitting. Robust and objective evaluation methods for creative tasks are essential for advancing AI applications in this domain.