How Well Can Large Language Models Understand Geometric Shapes? An exploration with synthesized polygon dataset

Elaine Li
University of Toronto Schools
liel@utschools.ca

Abstract
Mathematical and logical reasoning is an important component of human intelligence. Thus, a common metric for evaluating Large Language Models (LLMs) is their ability to solve mathematical problems. Recently, LLMs have shown remarkable performance in completing various tasks such as text generation, text understanding and image analysis. Their mathematical and reasoning ability has also advanced rapidly, allowing them to solve complex algebra problems. However, LLMs still exhibit limitations in describing and reasoning about geometric and spatial concepts, failing to accurately identify and understand the logic within geometric figures. In order to address this gap in understanding, numerous diverse datasets of geometric figures and metadata are needed to continue training their geometric reasoning capabilities. In this research paper, we introduce an innovative algorithm to create synthetic polygon geometric shape datasets, and define methods to integrate synthetic geometric images and metadata into major LLMs for training, validation, and evaluation of their geometric reasoning abilities.

1. Introduction
Recently, generative Artificial Intelligence (AI) models, particularly Large Language Models (LLMs), have undergone revolutionary developments. With the release of models such as ChatGPT and Gemini AI, high quality LLM models have become publicly available, and their use have been implemented in education and medical fields.

Large Language Models (LLMs) are generalized models trained on extensive datasets to solve a variety of common language problems, such as question answering and text generation, and can be fine tuned with a smaller dataset for specific purposes. In recent years, LLMs have demonstrated an extensive understanding of language, and are able to process text prompts from the user and generate accurate and informative responses. Moreover, they possess extensive computational power, allowing them to solve complex problems using reasoning. In mathematics, LLMs has also made significant progress, with various math datasets developed to to train and evaluate their mathematical problem-solving abilities. For instance, OpenAI's gsm8k dataset (Cobbe et al., 2021) includes elementary-level word problems, The MATH dataset (Hendrycks et al., 2021) provides step-by-step solutions for each problem, and MathEval (n.d.) evaluates the performance of LLMs on a diverse set of mathematical tasks across 20 languages. These datasets show that LLMs are able to perform calculations, solve mathematical equations, and even understand complex mathematical concepts, demonstrating their computational power and reasoning abilities.

However, unlike numerical and algebraic problems, geometric problems involve a combination of natural language and visual representations. They pose challenges such as symbolic and graphical abstraction, knowledge of geometric theorems and concepts, combined with quantitative reasoning. Training LLMs for geometry requires datasets with higher standards and greater diversity. If such datasets are scarce, the accuracy of LLM outputs may be very low, and may even exhibit hallucinations, delivering fabricated and fictitious information. For example, researchers from Auburn University and the University of Alberta observed that even advanced vision-language models (VLMs) struggled to understand spatial relationships involving basic geometric shapes, such as determining whether two circles overlap. (Rahmanzadehgervi, 2024)

To address this issue, several geometry datasets have been developed to enhance LLMs' adaptability and understanding, such as Geometry3K, GeoQA, GeoQA+, and UniGeo. However, the number of reliable datasets remains limited.

In this research paper, we use an innovative algorithm to randomly generate synthetic geometric images along with corresponding metadata, creating a synthetic polygonal geometric dataset. Our method does not require manual data collection and manual data labeling. We also define a system of methods in order to integrate these synthetic images and metadata into leading LLMs, offering an approach for training, evaluating, and validating their geometric reasoning capabilities.

2. Project Overview

The project includes the following components:

1. Defining an initial algorithm for image composition
2. Creating a synthetic polygonal geometry dataset.
3. Designing a set of prompts for integration into LLMs.
4. Validating and evaluating LLM models.

We identified key factors that constitute geometric figures and employed the Graham Scan algorithm to form connections between points and lines within each figure. This approach generated random pairs of images and metadata, forming a synthetic polygonal geometry dataset. A set of prompts was then designed for application in the Google Gemini model, allowing for data validation, accuracy analysis, and geometric capability assessment.

3. Literature Review

Solving text-only mathematical problems with LLMs

Large language models (LLMs) have shown remarkable progress in their ability to solve mathematical problems in recent years. They can perform calculations, solve equations, and even understand complex mathematical concepts. Many benchmark datasets and evals have been created to evaluate LLM's abilities in solving word problems. There are many datasets in the literature evaluates LLMs math reasoning capabilities, including OpenAI's gsm8k dataset (Cobbe et al. 2021) which covers grade school word math problems, MATH dataset (Hendrycks et al. 2021) which contains step-by-step solutions for each of its problems.

Websites such (MathEval, n.d.) showcases how different large language models (LLMs) perform on various mathematical tasks over 20 different datasets in various languages.

Solving text-and-image geometric problems with VLMs

Vision language models (VLMs) have a distinct advantage over LLMs when it comes to solving math problems, particularly those involving geometry. VLMs can directly process visual information, such as diagrams and graphs, which is crucial for solving geometric problems which contain both text and images.

In the literature, datasets such as CLEVR-Math (Lindström and Abraham 2022), GeoS (Seo et al., n.d.), GeoQA (Chen et al. 2021) and GeomVerse (Kazemi et al. 2023) contain mathematical problems with both text and image and are curated to evaluate VLMs' mathematical reasoning capabilities. The systemic approaches to create such datasets typically involve a set of standard shapes which are used as atomic building blocks of the visual part of the questions. Such evaluations focus on not only the LVM's ability to understand the images, but also the ability to reason.

This work

In this work, we propose an algorithm to generate planar geometry images, accompanied with texts of a set of questions regarding the basics of the geometrics and the answers. The algorithm is easy to use to create text-and-image instances. The focus of this work is to evaluate the VLMs ability to understand basic geometric concepts such as edges, vertices, line segments etc.

4. Synthesized Geometric Dataset

We created an algorithm which generates a synthetic dataset to evaluate VLMs. The dataset contains images of convex polygons on a 2D plane with points on the sides of the polygons, as well as connected line segments and diagonals within the polygon. The dataset contains 10 sub datasets, each containing 100 images, paired with metadata of the images.

In order to create the polygons, we take the following steps.

1. generate points on a 2D plane
2. select and connect a subset of the points to form the polygon
3. create points on the edges of the polygon, connect diagonals and line segments
4. eliminate and regenerate polygons which are unclear
5. Plot the polygon and resize the image to be 256 pixels
6. Store metadata regarding the generated polygon

4.1 Generating Points

First, we generate random points within a 2D plane. We use two random variables x_i, y_i, and set them as the x-coordinate and y-coordinate of a point on the 2D plane. We set x_i, y_i to be integers and $0 \leq x_i, y_i \leq 20$ to avoid generating points which are too far away, and to avoid floating point precision issues. By generating around 20 values for each of x_i and y_i and pairing them, we form 20 points positioned randomly between (0, 0) and (20, 20) on the 2D plane.

If we connect these 20 points sequentially to form the edges of a polygon, it is very likely that such a polygon would be self intersecting or concave instead of convex polygons. To solve this issue, we use the Graham Scan Algorithm on the points.

4.2 Convex Polygons Generation with Graham Scan Algorithm

The Graham Scan algorithm generates a convex hull of the points from the set of random generated points. The convex hull is a sorted subset of points of a larger set that when connected in order, it forms a polygon which encloses all points in the set.

The Graham Scan is one of the most efficient algorithms that searches for the convex hull of a set of points. First, it finds the bottommost point and the second bottommost point, p_0 and p_1. Then, it sorts the rest of the points according to its angle with respect to the line p_0p_1. Let the sorted list of points be [p_2, p_3, …, p_n]. The sorted list of points have increasing angles with respect to p_0 p_1. Connecting these points in order and connecting p_2, p_n to p_1, p_0 respectively would form a simple closed path.

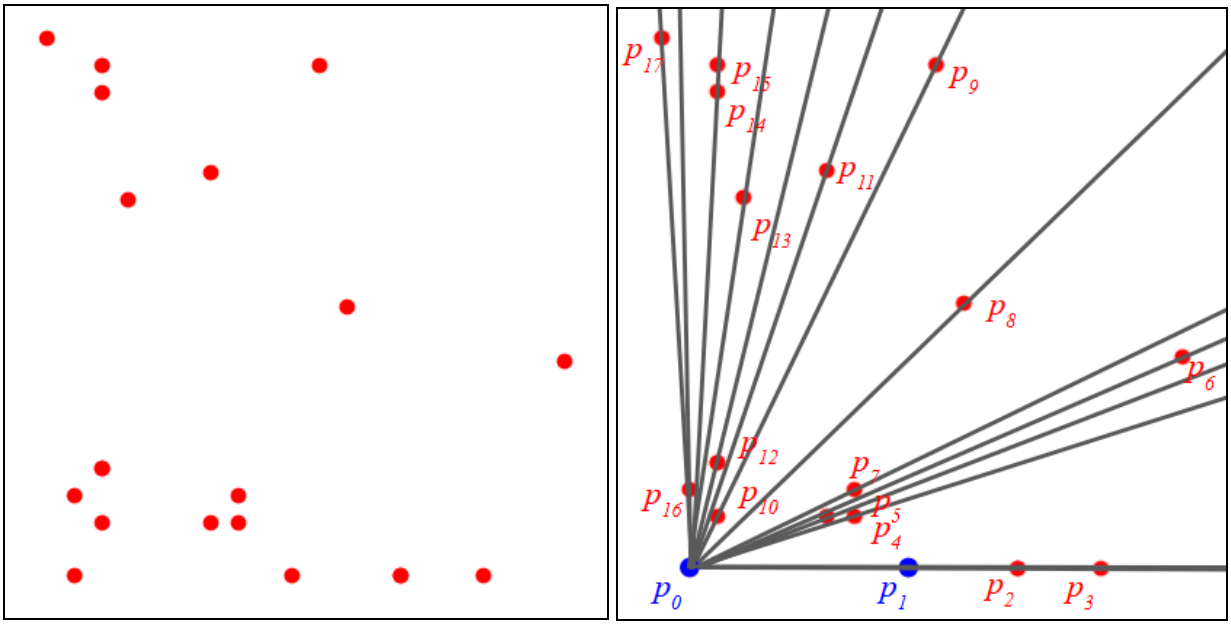

Figure 1: The sorting algorithm is run on a set of randomly generated points shown on the left. p_0 and p_1 are identified, and the rest of the points are sorted based on increasing angle between the lines $p_0 p_i$ and $p_0 p_1$ for each point p_i.

Compared to other Convex Hull algorithms, which use trigonometry or other methods to sort the points, the Graham Scan uses a simpler and faster way of doing so, where it considers the orientation of the points. For example, if we were comparing points p_i and p_j, we see that if p_i, p_0, p_j are in counterclockwise order, p_i should be placed before p_j in the list.

Then, we remove all points that form concave or linear sides from the list of points. To do this, we search for all points p_i, p_{i+1}, p_{i+2} such that p_i, p_{i+1}, p_{i+2} forms a counterclockwise orientation, and remove p_{i+1} from the list. To do this, we simply connect the line $p_i p_{i+1}$, and determine p_{i+2} is above or below this line. If it is above, then p_i, p_{i+1}, p_{i+2} forms a counterclockwise orientation. Otherwise, it forms a clockwise orientation, or p_i, p_{i+1}, p_{i+2} are on the same line. In both cases, we keep the point p_{i+1}. Then, connecting this list of points results in a convex closed path.

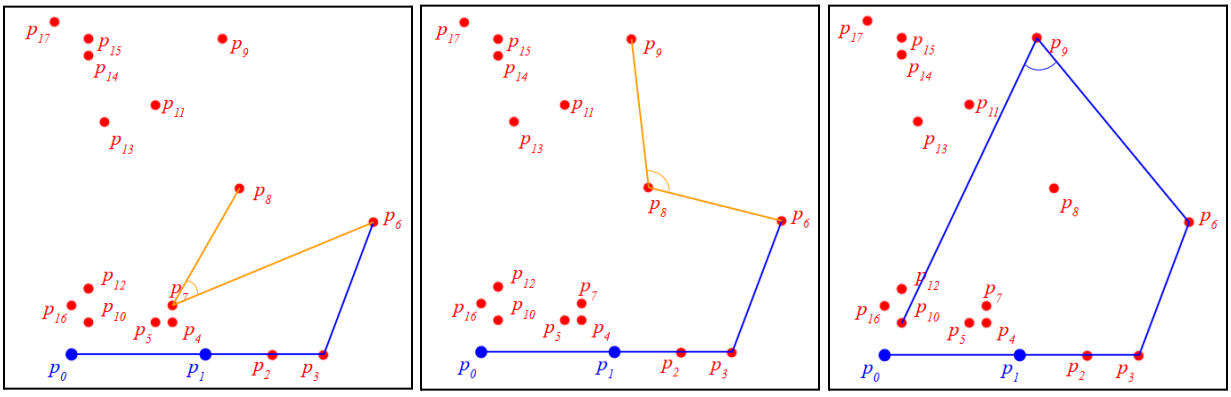

Figure 2: As shown in the leftmost figure, p_6, p_7, p_8 has a counterclockwise orientation, so we remove p_7 and consider p_6, p_8, p_9. As shown in the middle figure, p_6, p_8, p_9 also has a counterclockwise orientation, so we remove p_8 and consider p_6, p_9, p_{10}. As shown in the figure on the left, p_6, p_9, p_{10} has a clockwise orientation, so we keep the point p_9 in the list, and proceed to check p_9, p_{10}, p_{11}.

After searching through the points using the Graham Scan algorithm, we designate the set of points in the convex hull as the vertices of a polygon. Connecting the vertices in the order of the sorted convex hull, we are able to form a convex polygon.

4.3 Line segments generation

After generating a convex polygon, we select a random set of edges of the polygon and select points on the edge. On each edge, we generate a random number n of points that evenly partition the edge into segments. For each of these points, we search for a list of other points it could possibly connect to. In particular, we exclude points on the same edge in the list of points. Then, we select a random set of points among this list and connect them, creating inner edges within the polygon, and thus increasing the complexity of the image. To avoid the polygon being too complex, we select no more than 5 edges, and no more than 2 points on each edge, and we connect each point to at most 5 other points.

4.4 Eliminating Polygons

Some polygons generated through this method may have angles close to 180° or side lengths which are close to having length 0. A side length of close to 0 makes the polygon appear similar to a degenerate polygon, while having a vertex with an angle close to 180° would make the vertex appear to be just a point on an edge. Both of these cases would interfere with the LLM's ability to analyze the image. Thus, we check the length of each edge as well as the angle of each internal angle of the polygon to ensure that the polygon possesses neither of the two undesirable traits mentioned above. If the polygon fails this check, it is deleted, and the algorithm would attempt to generate another polygon.

4.5 Plot the polygon and resize the image

Finally, we plot the polygon and the edges on a 2D plane with matplotlib. Some randomly generated data points are considerably far from each other, and varying image sizes may have an impact on the LLM's ability to analyze the image. Thus, we resize the height of the image to a uniform height of 256 pixels.

4.6 Image Metadata Generation

We store all the metadata within a dictionary, including the number of sides in the polygon and the corresponding shape; the vertices and edges of the polygon; the number of sides with points, the number of points on each side, as well as the inner edges within the polygon. We store the dictionary of metadata for each image within a json file paired with each image. When evaluating the LLM's responses for each image, we would have easy access to the metadata of the image, providing the ground truth for the queries.

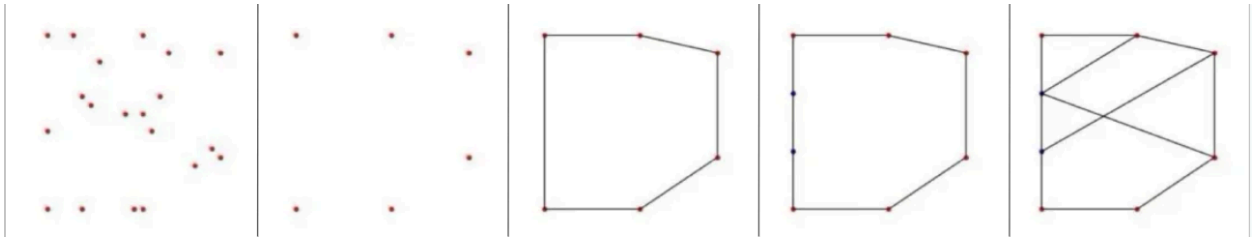

Figure 3: A side-by-side comparison of the image through each step of the process, from point generation, Graham Scan algorithm, connecting edges, generating points on edges, and connecting line segments.

4.7 Limitations

Currently, our algorithm relies on 2-dimension (2D) points and lines connecting 2D points. This representation has not yet addressed geometric spaces and dimensions beyond 2D. For our next steps, we will incorporate 3-dimension points and models, extending the algorithm to three-dimensional geometry to generate irregular, convex prisms.

5. Data Validation and Model Evaluation

Data validation ensures that parameters such as the type, position, and size of the synthetic polygonal geometry dataset meet expectations and can be accurately utilized by LLMs. This is a critical step in integrating the dataset with LLMs. We used the Google Gemini model as a sample, loaded the synthetic polygonal geometry dataset, and conducted extensive queries and validations to evaluate the model's understanding of synthetic geometric concepts (e.g., lines and polygons).

5.1 LLM Model Selection

Due to the high costs of Large Language Models, we chose to use Google's Gemini models in this project. In particular, we chose to use Google's "gemini-1.5-flash". There are also three parameters we set which impacts the performance of the LLM - temperature, top_p and top_k. Given the questions involved in this project all have deterministic answers, we choose to use 0.1 for temperature, 0.9 for top_p and 2 for top_k in order to generate more deterministic answers from LLM.

5.2 Questions and Prompts

We loaded the synthetic datasets and used the Google Gemini model to submit large amounts of queries to the LLM. We tested the LLM with 10 datasets, each consisting of 100 images, and we ran 5 trials for each dataset. In each trial, we started by giving the LLM the prompt "Imagine you are a mathematician, and you are given an image of a 2D polygon with lines connected within it." in order to give it context. Then, we gave the following questions to the LLM:

1) "What is the shape of the polygon in the image? Choose one of the answers in 'triangle, quadrilateral, pentagon, hexagon, none of above'."
2) "How many sides does the polygon in the image have? Just give one number."
3) "How many line segments are there in total in the image where each endpoint of the line segment is either a vertex or a point on the sides of the polygon? Just give one number."
4) "How many points (including vertices) in total are there on the sides of the polygon? Just give one number."
5) "What is the maximum number of endpoints (not including vertices) on any single edge of the polygon? Just give one number"
6) "How many sides of the polygon contain endpoints which are not vertices? Just give one number"

In particular, for each question, we prompted the LLM each of the 50 images, and verified its response to the question and image. For question 1, we checked if the response contained the correct word answer (triangle, quadrilateral, pentagon, hexagon, none of the above). We counted the response as correct if it contained the phrase corresponding to the right answer anywhere in the response. For all other questions, we verified the response by checking the number from the response with the answer. We recorded down the number of correct responses, calculating the % accuracy by dividing the number of correct responses by the number of total responses, then moved on to the next question.

This data validation strategy focuses on logical coherence and depth of reasoning during problem-solving. In multi-step reasoning tasks (e.g., arithmetic or logic), LLMs often make errors. Our multi-step reasoning queries addresses this issue and improves the model's performance.

5.3 Answer Correctness Check

For each image, we extracted the ground truth of the queries using the json file containing metadata of the image. Specifically, the file stores the number of sides in the polygon and the corresponding shape; the vertices and edges of the polygon; the number of sides with points, the number of points on each side, as well as the inner edges within the polygon.

For question 1, the correct answer is obtained by the "shape" key in the metadata. For question 2, the correct answer is obtained by the "num_sides" key in the metadata. For question 3, the correct answer is

obtained by the "num_lines" key in the metadata. For question 4, the correct answer is obtained by the "num_points" key in the metadata. For question 5, the correct answer is obtained by finding the maximum value within the list obtained by the "num_points_on_sides" key in the metadata. For question 6, the correct answer is obtained by finding the length of the list obtained by the "num_points_on_sides" key in the metadata.

For each question, we take the LLM's response, and search for the ground truth string within the response string. For example, for question 1, if the correct answer is "triangle" and the LLM's response contains the word "triangle," it would be marked as a correct response. For questions 2 to 6, if the number that is the correct answer appears in the LLM's response, the response is marked as a correct response. This way, we are able to evaluate the correctness of the LLM's responses using the ground truth.

6. Evaluation Results

We tested the Google Gemini model using 10 datasets, each containing 100 images. For each dataset, we conducted five trials and recorded the accuracy for each question across all trials. The average accuracy was calculated based on these five trials.

After calculating and averaging the % accuracy per problem across the 5 trials, the results are as follows:

Model parameters: 'gemini-1.5-flash', "temperature": 0.1, "top_p": 0.9, "top_k": 2

Question 1: "What is the shape of the polygon in the image? Choose one of the answers in 'triangle, quadrilateral, pentagon, hexagon, none of above'."

Dataset #	1	2	3	4	5	6	7	8	9	10
# Correct	66	58	61	68	69	69	56	55	54	55
# Wrong	34	42	39	32	31	31	44	45	46	45
Accuracy	66%	58%	61%	68%	69%	69%	56%	55%	54%	55%
Average accuracy: 61.1%										

Question 2: "How many sides does the polygon in the image have? Just give one number."

Dataset #	1	2	3	4	5	6	7	8	9	10
# Correct	32	30	30	18	28	27	27	23	29	21
# Wrong	68	70	70	82	72	73	73	77	71	79
Accuracy	32%	30%	30%	18%	28%	27%	27%	23%	29%	21%
Average accuracy: 26.5%										

Question 3: "How many line segments are there in total in the image where each endpoint of the line segment is either a vertex or a point on the sides of the polygon? Just give one number."

Dataset #	1	2	3	4	5	6	7	8	9	10
# Correct	6	8	5	8	6	6	7	6	8	5
# Wrong	94	92	95	92	94	94	93	94	92	95
Accuracy	6%	8%	5%	8%	6%	6%	7%	6%	8%	5%
Average accuracy: 6.5%										

Question 4: "How many points (including vertices) in total are there on the sides of the polygon? Just give one number."

Dataset #	1	2	3	4	5	6	7	8	9	10
# Correct	3	2	1	2	1	1	3	1	2	0
# Wrong	97	98	99	98	99	99	97	99	98	100
Accuracy	3%	2%	1%	2%	1%	1%	3%	1%	2%	0%
Average accuracy: 1.6%										

Question 5: “What is the maximum number of points (not including vertices) on any single edge of the polygon? Just give one number”

Dataset #	1	2	3	4	5	6	7	8	9	10
# Correct	33	29	39	32	35	36	32	34	28	22
# Wrong	67	71	61	68	65	64	68	66	72	78
Accuracy	33%	29%	39%	32%	35%	36%	32%	34%	28%	22%
Average accuracy: 32%										

Question 6: “How many sides of the polygon contain endpoints which are not vertices? Just give one number”

Dataset #	1	2	3	4	5	6	7	8	9	10
# Correct	37	24	24	28	19	23	23	32	22	24
# Wrong	63	76	76	72	81	77	77	68	78	76
Accuracy	37%	24%	24%	28%	19%	23%	23%	32%	22%	24%
Average accuracy: 25.6%										

6.1 Analysis

The average accuracy for Question 1 was 61.1%, indicating that the Google Gemini model has a solid understanding of polygons and can detect the shape of the external polygon from the image.

The model performed poorly on Question 2, with an average accuracy of 26.5%, demonstrating that it failed to establish a connection between the polygon's shape and the number of its sides. This suggests that the model deducts the shape of the polygon based on the image's configuration rather than calculating the number of sides.

The model performed very poorly on Questions 3, 4, with the highest accuracy being 6.5% for Question 3, below 3% for Question 4. These questions involve asking to identify points on lines which are the sides of the polygon. Meanwhile, the model performed poorly but still attains a 32% and 25.6% average accuracy on question 5 and 6, respectively. These two questions only involve identifying and counting points on lines. This suggests that the model has limited ability to identify points on lines, and is not able to identify the boundary edges of a polygon given such a geometric image.

7. Conclusion

This paper evaluates the Google Gemini model’s understanding of geometric concepts, namely lines and polygons. We generated synthesized image datasets and prompted the model with the image and various questions.

7.1 Integration of Dataset with LLM models

The integration of the synthetic polygonal geometry dataset with LLMs has proven effective, providing a novel and viable method for training, evaluating, and validating the geometric reasoning abilities of LLMs.

7.2 Advantage of Synthetic Datasets

The synthetic polygonal geometry dataset is not reliant on collecting existing datasets or creating data and labelling their descriptions and metadata manually, but is generated randomly through synthetic data. This eliminates the need for manual data collection and labeling, making it particularly suitable for training large language models.

7.3 Efficiency of the Synthetic Algorithm

The synthetic polygon algorithm efficiently leverages parallel computing technology to rapidly generate images. In this experiment, a standard household laptop was able to generate more than 100 images in under 10 seconds. After filtering out similar samples, the massive data pool can ultimately form a synthetic polygonal geometry dataset containing infinite unique and diverse samples of varying difficulty levels. This provides large-scale training data to "feed" LLMs.

7.4 Evaluation Insights

The effective evaluation of the Google Gemini model revealed and simulated the current limitations of LLMs in geometric reasoning, offering research directions for future improvements in this area, in particular, the identification of geometric shapes and its boundary edges.

7.5 Limitations and Future Steps

Due to time and budget constraints, this study only verified the integration of the synthetic polygonal geometry dataset with the Google Gemini model. In the next phase, we will validate and evaluate other major LLM models individually.

8. Acknowledgments

This paper was written by the author Yilin Li with the guidance from her mentor, Ms. Zhang. Ms. Zhang's tutoring had helped me establish an understanding of modern generative large language models, their capabilities and limitations. Zhang has also guided me through designing the algorithm and coding for the end results.

9. References

Chen, Jiaqi, Jianheng Tang, Jinghui Qin, Xiaodan Liang, Lingbo Liu, Eric Xing, and Liang Lin. 2021. "[2105.14517] GeoQA: A Geometric Question Answering Benchmark Towards Multimodal Numerical Reasoning." arXiv. https://arxiv.org/abs/2105.14517.

Cobbe, Karl, Vineet Kosaraju, Mohammad Bavarian, Reiichiro Nakano, Christopher Hesse, Mark Chen, Heewoo Jun, et al. 2021. "Training Verifiers to Solve Math Word Problems." arXiv. https://arxiv.org/abs/2110.14168.

Hendrycks, Dan, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. "[2103.03874] Measuring Mathematical Problem Solving With the MATH Dataset." arXiv. https://arxiv.org/abs/2103.03874.

Kazemi, Mehran, Hamidreza Alvari, Ankit Anand, Jialin Wu, Xi Chen, and Radu Soricut. 2023. "GeomVerse: A Systematic Evaluation of Large Models for Geometric Reasoning." arXiv. https://arxiv.org/abs/2312.12241.

Lindström, Adam D., and Savitha S. Abraham. 2022. "CLEVR-Math: A Dataset for Compositional Language, Visual and Mathematical Reasoning." arXiv. https://arxiv.org/abs/2208.05358.

MathEval. n.d. "MathEval." Evaluation Leaderboard - MathEval. Accessed September 6, 2024. https://matheval.ai/en/leaderboard/.

Seo, Minjoon, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. n.d. "Solving Geometry Problems: Combining Text and Diagram Interpretation." ACL Anthology. Accessed September 6, 2024. https://aclanthology.org/D15-1171/.

Rahmanzadehgervi, Pooyan, Logan Bolton, Mohammad Reza Taesiri, and Anh Totti Nguyen. 2024. "Vision language models are blind: Failing to translate detailed visual features into words." arXiv. https://arxiv.org/abs/2407.06581