EUCLID: SUPERCHARGING MULTIMODAL LLMS WITH SYNTHETIC HIGH-FIDELITY VISUAL DESCRIPTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal large language models (MLLMs) have made rapid progress in recent years, yet continue to struggle with *low-level visual perception* (LLVP)—particularly the ability to accurately describe the geometric details of an image. In this paper, we first demonstrate this limitation by introducing *Geoperception*, a benchmark designed to evaluate an MLLM's ability to accurately transcribe 2D geometric information from an image. We then conduct a comprehensive empirical study to explore strategies for improving LLVP performance through the use of synthetic high-fidelity visual description data. Our findings highlight the benefits of certain model architectures and training techniques, including the use of CNN-based visual encoders and multi-stage training with a data curriculum. Notably, we find that a data curriculum enables models to learn challenging geometry understanding tasks which they fail to learn from scratch. Lastly, we develop *Euclid*, a family of models specifically optimized for strong low-level geometric perception. Although trained on synthetic multimodal data, Euclid shows strong generalization ability on novel real-world geometry shapes. For instance, Euclid outperforms the best closed-source model in our benchmark by up to 58.56% on certain Geoperception benchmark tasks and 10.65% on average across all tasks.

1 INTRODUCTION

Multimodal large language models (MLLMs) have rapidly progressed in recent years, demonstrating remarkable potential in understanding and reasoning about the visual world through the powerful capabilities of large language models (LLMs) (Liu et al., 2024c;a; Achiam et al., 2023; Team et al., 2023; Hu et al., 2023; Tong et al., 2024a; Wang et al., 2024a). These models have showcased strong performance in tasks such as visual question answering (VQA) (Goyal et al., 2017), image captioning (Lin et al., 2014), and multimodal reasoning (Liu et al., 2023) – for example, LLaVA-NeXT-34B (Liu et al., 2024b) achieves an impressive 83.7% accuracy on the VQAv2 benchmark (Goyal et al., 2017), a comprehensive natural image VQA benchmark.

While MLLMs achieve impressive results on tasks like VQA, their performance relies on high-level semantic extraction (Tong et al., 2024b); in contrast, they often fall short on *low-level visual perception* (LLVP) — i.e., the ability to accurately describe the geometric details of an image, such as the points, lines, angles, and spatial relationships among its constituent objects. This limitation becomes especially apparent in tasks requiring precise visual descriptions, such as mathematical visual problem solving (Zhang et al., 2024a; Lu et al., 2023), scientific visual understanding (Yue et al., 2024; Fu et al., 2024a), abstract visual reasoning (Jiang et al., 2024; Ahrabian et al., 2024), and even simple visual comprehension (Rahmanzadehgervi et al., 2024; Wang et al., 2024b). Furthermore, LLVP is also vital in real-world applications, including spatial understanding for robotics, medical image analysis for accurate diagnosis, GUI agents, quality control in manufacturing to detect subtle defects, autonomous driving systems that rely on exact object localization, and augmented reality applications that demand precise overlay of virtual objects onto the real world. In this paper, we study the challenges of LLVP abilities in MLLMs, take steps to understand the root cause, and improve their performance. Our study focuses specifically on 2D geometry, a domain where components and relationships are rigorously defined, and both images and textual descriptions can be generated at scale synthetically.

We start by developing *Geoperception*, a 2D geometry multimodal benchmark that focuses exclusively on basic LLVP questions without incorporating higher-order reasoning. Our findings reveal

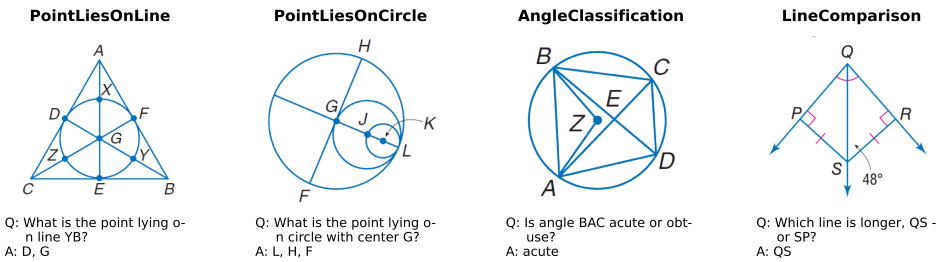


Figure 1: Four examples from our *Geoperception* dataset. The questions are sourced from the Geometry-3K corpus (Lu et al., 2021), which compiles problems from two widely-used high school textbooks. We perform filtering, validation, and generate question-and-answer text for each image.

that current MLLMs consistently struggle on this benchmark. This limitation raises an important research question: what is causing the difficulties exhibited by contemporary MLLMs in LLVP tasks? We hypothesize the main factor to be the insufficient availability of high-fidelity visual description datasets. Furthermore, in the absence of sufficient data, it is challenging to identify and adopt architectural choices and training strategies that could improve the effectiveness and efficiency of LLVP-specific training. To this end, we develop a synthetic dataset engine for large-scale generation of 2D geometry images paired with high-fidelity textual descriptions. This synthetic dataset enables us to conduct a comprehensive empirical study with controlled experiments to explore strategies for improving MLLMs' performance on LLVP tasks within the 2D geometry domain. Our key insights include: **1. Scaling LLM size does not benefit LLVP learning. 2. CNN visual encoders are more suitable in LLVP learning than ViT architectures. 3. Tuning vision encoders does not offer a strong advantage. 4. Curriculum learning significantly improves a model's performance, especially in understanding complex geometric shapes.** With these lessons learned, we then train a family of models—using a carefully designed curriculum of synthetic data—that are specifically optimized for strong LLVP, which we call Euclid. Upon evaluation, we show that our models excel on real-world low-level geometric perception tasks.

2 Geoperception: A Benchmark for Geometric LLVP

Although the shortcomings of MLLMs in LLVP are commonly recognized, there is no comprehensive benchmark that focuses purely on this task. Our goal is to construct a benchmark focusing solely on the LLVP ability of MLLMs, which is also representative enough of real-world applications. As a fundamental and broadly representative LLVP ability in many applications, we select geometry understanding as our domain of dataset construction. Following basic geometry definitions and axioms, we define seven basic geometric LLVP tasks in our benchmark. We include the full details of benchmark tasks and construction in Appendix B.

Current MLLMs struggle to perceive low-level geometry annotations and relationships. We evaluate seven leading MLLMs, both open source and closed source. Their performances are shown in Table 1. Despite the simplicity of Geoperception for humans, it remains a considerable challenge for even the most advanced MLLMs. Notably, all models fall short of achieving 30% accuracy on the *PointLiesOnLine* task and do not outperform the text-only GPT-4o mini model in *AngleClassification* task. Closed source models generally outperform open source ones, with Gemini-1.5-pro attaining the highest average score of 56.98%, followed by gemini-1.5-flash at 54.76%. Among open source models, Pixtral-12B achieves the best performance with an overall score of 41.95%. Noted Cambrian-1 (Tong et al., 2024a), which is reported to be trained on Geo-170K (Gao et al., 2023), a geometry multimodal instruction tuning dataset built on the logical annotation of Geometry-3K, the same source with Geoperception, still faces challenges in our Geoperception task, despite being trained on the dataset having the same images and augmented text annotations.

3 Empirical Study on MLLM Design Space

We hypothesize the insufficient availability of high-fidelity visual description datasets to be a main factor involved in this shortcoming. This may be because LLVP is intuitive for humans, and the

Table 1: Performance (average evaluation score) of different models on Geoperception. POL: PointLiesOnLine, POC: PointLiesOnCircle, ALC: AngleClassification, LHC: LineComparison, PEP: Perpendicular, PRA: Parallel, EQL: Equals. As the Random Baseline method, we use GPT-4o-mini, given the same textual instruction but without an image. The best model for each task is **bolded**.

	Logical		Numerical		Annotations			
Model	POL	POC	ALC	LHC	PEP	PRA	EQL	Overall
Random Baseline	1.35	2.63	**59.92**	51.36	0.23	0.00	0.02	16.50
Open Source								
Molmo-7B-D (Deitke et al., 2024)	11.96	35.73	56.77	16.79	1.06	0.00	0.81	17.59
Llama-3.2-11B (Dubey et al., 2024)	16.22	37.12	59.46	52.08	8.38	22.41	49.86	35.08
Qwen2-VL-7B (Wang et al., 2024a)	21.89	41.60	46.60	63.27	26.41	30.19	54.37	40.62
Cambrian-1-8B (Tong et al., 2024a)	15.14	28.68	58.05	61.48	22.96	30.74	31.04	35.44
Pixtral-12B (AI, 2023)	24.63	53.21	47.33	51.43	21.96	36.64	58.41	41.95
Closed Source								
GPT-4o-mini (Achiam et al., 2023)	9.80	61.19	48.84	69.51	9.80	4.25	44.74	35.45
GPT-4o (Achiam et al., 2023)	16.43	**71.49**	55.63	74.39	24.80	60.30	44.69	49.68
Claude 3.5 Sonnet (Anthropic, 2024)	25.44	68.34	42.95	70.73	21.41	63.92	**66.34**	51.30
Gemini-1.5-Flash (Team et al., 2023)	**29.30**	67.75	49.89	76.69	29.98	63.44	66.28	54.76
Gemini-1.5-Pro (Team et al., 2023)	24.42	69.80	57.96	**79.05**	**38.81**	**76.65**	52.15	**56.98**

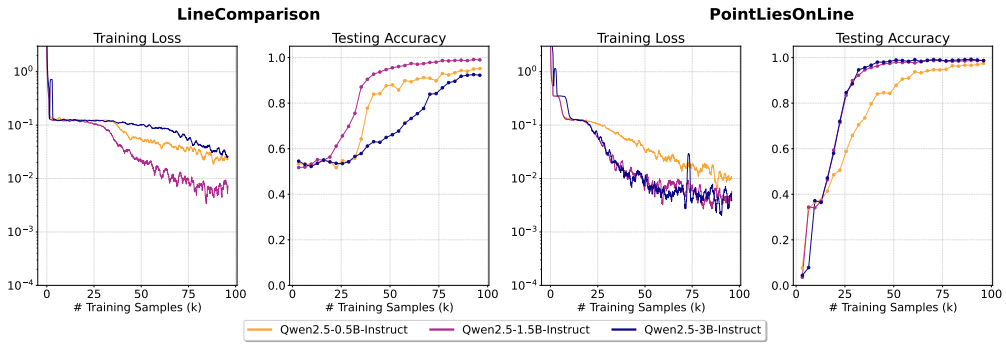


Figure 2: LLM size experiments. Training loss and testing accuracy curve comparing three choices of LLM size with a fixed visual encoder and multimodal connector. Training losses are window-smoothed using a window size of 10 for better visibility.

datasets used to train MLLMs are primarily annotated by humans[1]. When describing visual information, humans often overlook LLVP details, assuming such details to be straightforward and self-evident. Furthermore, in the absence of sufficient datasets, it is challenging to identify and adopt architectural choices and training strategies that could improve the effectiveness and efficiency of LLVP-specific training. To this end, we develop a synthetic dataset generation engine to programmatically produce geometry shapes for training our MLLM. We include the full detail of our dataset generation engine in Appendix F. With the sufficient training dataset, we are able to conduct **fully controlled experiments** on different aspects of MLLMs focusing specifically on their LLVP abilities on geometry domain.

Exploration of MLLM design space. We follow the current most popular design of MLLMs (Liu et al., 2024c): a visual encoder, a LLM and an MLP connector in between. Although there exists many ready-to-use MLLMs off the shelf, they differ significantly in many aspects, such as architectural choices (e.g., vision encoder, LLM, and multimodal connector), training datasets, and hyperparameters. Consequently, due to the high cost in re-pretraining the visual encoders or LLMs, we opt to train our own MLLM with existing vision encoders and LLMs. This is similar with some recent empirical exploration in MLLM design space (Tong et al., 2024a; McKinzie et al., 2024), but with our large-scale controllable 2D geometry dataset. Specifically, we start with a typical setting of MLLMs: CLIP-ViT-L/14 (Radford et al., 2021) as the visual encode, a two layer MLP as multimodal connector and the latest Qwen-2.5 series (Team, 2024b) as LLM. During training, we actively tune the MLP and LLM, while keeping visual encoder frozen.

[1]This includes both explicit annotation in multimodal instruction-tuning datasets and implicit annotation in large-scale internet collections of text-image pairs

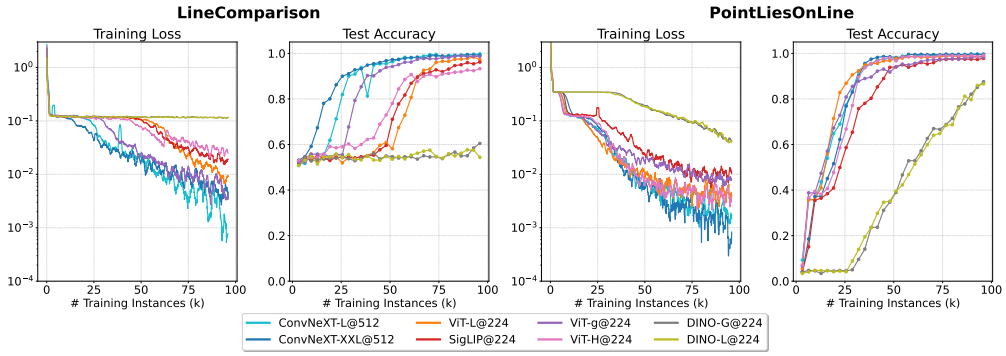


Figure 3: Vision encoder experiments. Training loss and testing accuracy (on a 1500 instances holdout test set) curve comparing eight visual encoders, with a fixed multimodal encoder and LLM. For a fair comparison, all visual encoder transcribe an image into 256 visual tokens. Training losses are window-smoothed using a window size of 10 for better visibility.

Lesson 1: Scaling LLM size does not benefit LLVP learning. Reflected in most MLLM releases (Liu et al., 2024a; Tong et al., 2024a; Wang et al., 2024a), scaling up the LLM often results in improved MLLM performance when trained on the same dataset. However, since their training datasets are complex mixtures from multiple sources, it is unclear whether this improvement stems from enhanced language-space reasoning or better visual perception. In contrast, our training dataset is designed with simplicity in the language space (following specific templates) and focuses on LLVP abilities. This setup allows us to better isolate and analyze the source of performance gains, providing clearer insights into the ability to learn better LLVP.

We use three variants of Qwen-2.5 (Team, 2024b): 0.5B, 1.5B, and 3B, while keeping other components in the MLLMs consistent and training them on the same dataset. The results are shown in Fig. 2. First, we observe a sharp decrease in loss at the start of training, which corresponds to the LLM adapting to answer templates, indicating no significant difficulty across different model sizes. The subsequent loss decrease after the plateau marks the beginning of learning LLVP, as also evidenced by the testing accuracy. For *LineComparison*, Qwen-2.5-1.5B performs the best, while Qwen-2.5-3B learns most slowly. For *PointLiesOnLine*, Qwen-2.5-1.5B and Qwen-2.5-3B perform nearly identically. while Qwen-2.5-0.5B learns relatively slower but eventually reaches a similar final performance as the other models. In summary, we do not observe a clear trend that larger LLMs learn LLVP tasks faster or better[2]. Based on these findings, we will use Qwen-2.5-1.5B for further exploration.

Lesson 2: CNN visual encoders are more suitable in LLVP learning than ViT architectures. We then study the choice of visual encoder architectures, including two families of architectures: Vision Transformer (ViT) (Dosovitskiy, 2020) and ConvNeXT (Liu et al., 2022); as well as two visual representation learning objectives: language-supervised learning (Radford et al., 2021) and self-supervised learning (Oquab et al., 2023). We summarize the visual encoders in our experiment in Table 4, for all vision encoders in our table, control the number of visual tokens to 256. The result is shown in Fig. 3. We find that ConvNeXt-XXLarge and ConvNeXt-Large consistently learns the fastest among all of the visual encoders. Notably, ConvNeXT-Large shows superior learning performance with the vision transformers which are 3-5 times larger. We hypothesize that CNN architecture extract visual features globally, effectively preserving low-level visual features. In contrast, ViT architectures split images into discrete patches, making it more challenging to retain the original low-level visual information. Self-supervised learning (SSL) visual encoders, DINO-v2, struggles to learn the geometry concept; we hypothesis this is due to the weak vision-language representation in these models. Surprisingly, although the SigLIP-family is widely-recognized as a better visual encoder (Tong et al., 2024a; Li et al., 2024a), we find that their performance in learning basic visual geometry attributes is limited.

[2]The BLINK (Fu et al., 2024c) benchmark shares similar observations. For example, among 14 tasks, LLaVA-1.5-13B outperforms its 7B variant in only 4 tasks. Additionally, LLaVA-one-vision (Li et al., 2024a)'s 0.5B variant outperforms its 7B variant by 3.9% and underperforms its 72B variant by only 3.3%.

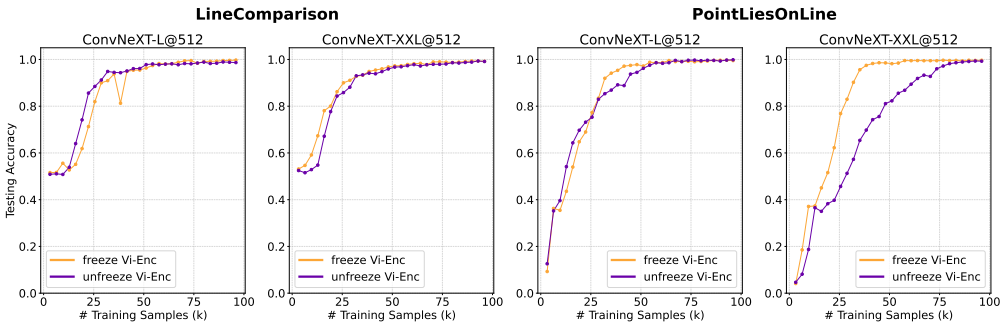


Figure 4: Tuning/freezing vision encoder experiments. Testing accuracy (on a 1500 instances hold-out test set) curve comparing freezing versus tuning the visual encoder during training.

Lesson 3: Tuning vision encoder does not offer strong advantage. By intuition, actively tuning the visual encoder can help it learn better visual representations which is helpful for LLVP. We aim to empirically justify the effect of tuning versus freezing the visual encoder. In Fig. 4, we show the testing accuracy curves of tuning and freezing visual encoders. Surprisingly, we find that compared with using a frozen encoder, tuning the visual encoder does not help the model learn LLVP faster or better. This suggests that current visual encoders seem to be able to preserve adequate information for LLVP, and it's sufficient to train LLMs to make better use of the visual features. In what follows, we will freeze the encoder for simplicity.

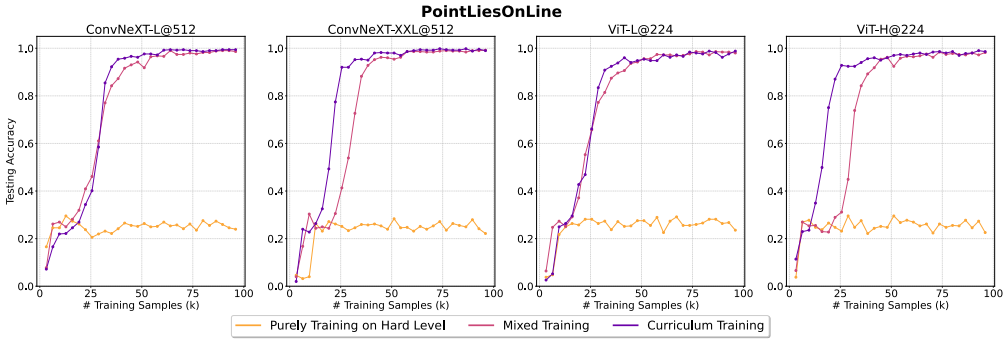


Figure 6: Curriculum learning experiments. Test accuracy on difficulty level hard of three training strategies: purely training on difficulty level hard, mixed training of difficulty levels easy/medium/hard, and curriculum training.

Lesson 4: Curriculum learning unleashes full potential. Finally, we study training data composition. In our preliminary experiment Fig. 19, we observe that the model fails to converge on difficulty level 3 of *PointLiesOnLine* and difficulty level 2 and 3 of *LineComparison*. However, when using mixed training set of all three difficulty levels, the model achieves convergence, despite using the same amount of data for each difficulty levels. We hypothesize that including easier levels aids the model in learning more complex levels. To test this hypothesis, we report the test accuracy for three difficulty levels separately during the mixed training of ConvNeXt-XXLarge, in Fig. 5, on both tasks. We notice that the testing accuracy for easier tasks increase earlier and more quickly than difficulty tasks. In *PointLiesOnLine* tasks, we notice an apparent plateau for hard level tasks until the model has trained on approximately 20K samples. During this period, the testing accuracy for easy and medium continue to increase. This suggests that learning easier shapes can significantly help the model tackle more challenging shapes, comparing with directly learning the challenging ones, aligning with the principles of curriculum learning.

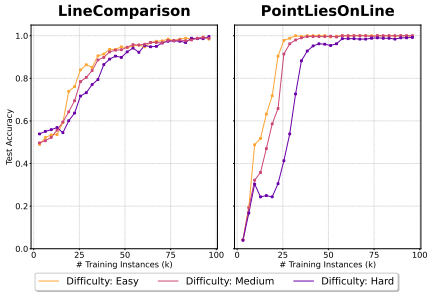


Figure 5: Separate testing accuracy curves on difficulty levels easy, medium, and hard, shown over the course of training on a mixture of all difficulty levels.

While mixed training enables effective spontaneous curriculum learning, we investigate whether a structured curriculum can further enhance model efficiency on challenging shapes. To this end, we monitor the model's performance and dynamically increase the difficulty level of training data (i.e., the curriculum stage) based on this performance. Specifically, the model starts by training on the easy level data. and is evaluated when it finishes a training round, using testing accuracy from the current level of data. Upon evaluation, if the model achieves an accuracy exceeding a predefined threshold θ, the framework advances the level to the next. Formally, the update rule for advancing stages is given by: if $\text{accuracy}_s > \theta \quad \Rightarrow \quad c \leftarrow c + 1$. The model is trained on a total of M rounds and K steps within each round. To avoid forgetting, we apply data smoothing at each stage. Specifically, we smooth our dataset distribution over all stages using an exponential attenuation function: $\text{ratio}_s = \exp\left(-\alpha \cdot |\text{stage}_s - c|\right)$, where α denotes the attenuation rate. This ensures that stages proximal to the current stage receive higher sampling probabilities.

We refer to this as our curriculum training strategy. Specifically, the accuracy threshold for advancing training stage θ is set to 0.99. We train all the models for $M = 30$ rounds, each round with $K = 50$ steps. The results are shown in Fig. 6. Firstly, we find that all of the models fail to converge when trained purely on hard level for *PointLiesOnLine* task. In contrast, the mixed training strategy shown by the red curve, consistently reaches faster convergence on hard level. Curriculum training strategy, shown by the purple curve, proves more efficient than mixed training.

4 EUCLID: A FAMILY OF MLLMS FOR GEOMETRIC VISUAL PERCEPTION

We take all of the lessons we learned in the previous sections and train Euclid, a family of MLLMs specifically designed for strong geometric LLVP. We use the same strategy as the curriculum training in Section 3, but scale our training to all tasks in Geoperception. For each task, we create N stages of training dataset shapes with progressively increasing geometric complexity.

Table 2: Performance comparison between Euclid and the best leading open source and closed source MLLMs on the seven tasks. Note that Euclid is *not* trained on any of the in-distribution data from the benchmark tasks below. The best model for each task is **bolded**.

	Logical		Numerical		Annotations			
Model	POL	POC	ALC	LHC	PEP	PRA	EQL	Average
Random Baseline	0.43	2.63	59.92	51.36	0.25	0.00	0.02	16.37
Pixtral-12B (AI, 2023)	24.63	53.21	47.33	51.43	21.96	36.64	**58.41**	41.95
Gemini-1.5-Pro (Team et al., 2023)	24.42	**69.80**	57.96	79.05	38.81	**76.65**	52.15	56.98
Euclid-ConvNeXt-Large	80.54	57.76	86.37	88.24	42.23	64.94	34.45	64.93
Euclid-ConvNeXt-XXLarge	**82.98**	61.45	**90.56**	**90.82**	**46.96**	70.52	31.94	**67.89**

Evaluation results. The results are shown in Table 2. Overall, although only trained on very simple synthetic geometry shapes, and using only a 1.5B language model, Euclid significantly outperforms current leading MLLMs in most of the tasks, showing strong generalization abilities on real-world geometry LLVP. Notably, in the *PointLiesOnLine* task, which is particularly challenging for existing MLLMs, Euclid achieves up to 82.98% accuracy, more than three times the performance of Gemini-1.5-Pro. On all both numerical tasks, *LineComparison* and *AngleClassification*, Euclid keeps superior performance. However, on three annotation tasks, Euclid's performance is limited. We hypothesis this is due to the limited setting of our annotation types and styles, making the model hard to generalize to diverse human geometry annotations.

5 CONCLUSION

In this work, we highlight the importance of accurate *low-level visual perception* (LLVP) in MLLMs. To this end, we first introduce Geoperception, a large-scale multimodal benchmark focused exclusively on geometry-domain LLVP. We find that even top models such as Gemini-1.5-Pro struggle significantly, although it is straightforward for humans. We then conduct an empirical study to explore the design space of MLLM training and architectures using the dataset generated by a geometric high-fidelity synthetic-data engine that we develop. Our key insides include that CNN-based visual encoders outperform ViT in our tasks; and employing a curriculum-based training approach yields much more model potential than direct task training. Based on insights from this study, we develop Euclid, a model trained purely on synthetic data generalizes effectively to real-world geometric LLVP tasks, surpassing the leading MLLMs by a substantial margin.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Kian Ahrabian, Zhivar Sourati, Kexuan Sun, Jiarui Zhang, Yifan Jiang, Fred Morstatter, and Jay Pujara. The curious case of nonverbal abstract reasoning with multi-modal large language models. *arXiv preprint arXiv:2401.12117*, 2024.

Mistral AI. Pixtral 12b. https://mistral.ai/news/pixtral-12b/, 2023. Accessed: 2024-09-27.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

Anthropic. The claude 3 model family: Opus, Sonnet, Haiku, March 2024. URL https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

Roman Bachmann, Oğuzhan Fatih Kar, David Mizrahi, Ali Garjani, Mingfei Gao, David Griffiths, Jiaming Hu, Afshin Dehghan, and Amir Zamir. 4m-21: An any-to-any vision model for tens of tasks and modalities. *arXiv preprint arXiv:2406.09406*, 2024.

Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

Markus J Buehler. Cephalo: Multi-modal vision-language models for bio-inspired materials analysis and design. *Advanced Functional Materials*, pp. 2409531, 2024.

Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14455–14465, 2024.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin Bransom, Kiana Ehsani, Huong Ngo, YenSung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, Andrew Head, Rose Hendrix, Favyen Bastani, Eli VanderBilt, Nathan Lambert, Yvonne Chou, Arnavi Chheda, Jenna Sparks, Sam Skjonsberg, Michael Schmitz, Aaron Sarnat, Byron Bischoff, Pete Walsh, Chris Newell, Piper Wolters, Tanmay Gupta, Kuo-Hao Zeng, Jon Borchardt, Dirk Groeneveld, Jen Dumas, Crystal Nam, Sophie Lebrecht, Caitlin Wittlif, Carissa Schoenick, Oscar Michel, Ranjay Krishna, Luca Weihs, Noah A. Smith, Hannaneh Hajishirzi, Ross Girshick, Ali Farhadi, and Aniruddha Kembhavi. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models, 2024. URL https://arxiv.org/abs/2409.17146.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Deqing Fu, Ruohao Guo, Ghazal Khalighinejad, Ollie Liu, Bhuwan Dhingra, Dani Yogatama, Robin Jia, and Willie Neiswanger. Isobench: Benchmarking multimodal foundation models on isomorphic representations. In *First Conference on Language Modeling*, 2024a. URL https://openreview.net/forum?id=KZd1EErRJ1.

Deqing Fu, Tong Xiao, Rui Wang, Wang Zhu, Pengchuan Zhang, Guan Pang, Robin Jia, and Lawrence Chen. Tldr: Token-level detective reward model for large vision language models. *arXiv preprint arXiv:2410.04734*, 2024b.

Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. *arXiv preprint arXiv:2404.12390*, 2024c.

Jiahui Gao, Renjie Pi, Jipeng Zhang, Jiacheng Ye, Wanjun Zhong, Yufei Wang, Lanqing Hong, Jianhua Han, Hang Xu, Zhenguo Li, et al. G-llava: Solving geometric problem with multi-modal large language model. *arXiv preprint arXiv:2312.11370*, 2023.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6904–6913, 2017.

Jinyi Hu, Yuan Yao, Chongyi Wang, Shan Wang, Yinxu Pan, Qianyu Chen, Tianyu Yu, Hanghao Wu, Yue Zhao, Haoye Zhang, et al. Large multilingual models pivot zero-shot multimodal learning across languages. *arXiv preprint arXiv:2308.12038*, 2023.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, and Ranjay Krishna. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. *arXiv preprint arXiv:2406.09403*, 2024.

Yifan Jiang, Jiarui Zhang, Kexuan Sun, Zhivar Sourati, Kian Ahrabian, Kaixin Ma, Filip Ilievski, and Jay Pujara. Marvel: Multidimensional abstraction and reasoning through visual evaluation and learning. *arXiv preprint arXiv:2404.13591*, 2024.

Mehran Kazemi, Hamidreza Alvari, Ankit Anand, Jialin Wu, Xi Chen, and Radu Soricut. Geomverse: A systematic evaluation of large models for geometric reasoning. *arXiv preprint arXiv:2312.12241*, 2023.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4015–4026, 2023.

Xianghao Kong, Ollie Liu, Han Li, Dani Yogatama, and Greg Ver Steeg. Interpretable diffusion via information decomposition. *arXiv preprint arXiv:2310.07972*, 2023.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024a.

Zhihao Li, Yao Du, Yang Liu, Yan Zhang, Yufang Liu, Mengdi Zhang, and Xunliang Cai. Eagle: Elevating geometric reasoning through llm-empowered visual instruction tuning. *arXiv preprint arXiv:2408.11397*, 2024b.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740–755. Springer, 2014.

Zhiqiu Lin, Xinyue Chen, Deepak Pathak, Pengchuan Zhang, and Deva Ramanan. Revisiting the role of language priors in vision-language models. *arXiv preprint arXiv:2306.01879*, 2023.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26296–26306, 2024a.

Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024b. URL https://llava-vl.github.io/blog/2024-01-30-llava-next/.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024c.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? *arXiv preprint arXiv:2307.06281*, 2023.

Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022.

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning. *arXiv preprint arXiv:2105.04165*, 2021.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*, 2023.

Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, Sam Dodge, Bowen Zhang, Philipp Dufter, Dhruti Shah, Xianzhi Du, Futang Peng, Floris Weers, et al. Mm1: Methods, analysis & insights from multimodal llm pre-training. *arXiv preprint arXiv:2403.09611*, 2024.

David Mizrahi, Roman Bachmann, Oguzhan Kar, Teresa Yeo, Mingfei Gao, Afshin Dehghan, and Amir Zamir. 4m: Massively multimodal masked modeling. *Advances in Neural Information Processing Systems*, 36, 2024.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Aitor Ormazabal, Che Zheng, Cyprien de Masson d'Autume, Dani Yogatama, Deyu Fu, Donovan Ong, Eric Chen, Eugenie Lamprecht, Hai Pham, Isaac Ong, et al. Reka core, flash, and edge: A series of powerful multimodal language models. *arXiv preprint arXiv:2404.12387*, 2024.

Shuai Peng, Di Fu, Liangcai Gao, Xiuqin Zhong, Hongguang Fu, and Zhi Tang. Multimath: Bridging visual and mathematical reasoning for large language models. *arXiv preprint arXiv:2409.00147*, 2024.

Cyril Picard, Kristen M Edwards, Anna C Doris, Brandon Man, Giorgio Giannone, Md Ferdous Alam, and Faez Ahmed. From concept to manufacturing: Evaluating vision-language models for engineering design. *arXiv preprint arXiv:2311.12668*, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Pooyan Rahmanzadehgervi, Logan Bolton, Mohammad Reza Taesiri, and Anh Totti Nguyen. Vision language models are blind. *arXiv preprint arXiv:2407.06581*, 2024.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36, 2024.

Min Shi, Fuxiao Liu, Shihao Wang, Shijia Liao, Subhashree Radhakrishnan, De-An Huang, Hongxu Yin, Karan Sapra, Yaser Yacoob, Humphrey Shi, et al. Eagle: Exploring the design space for multimodal llms with mixture of encoders. *arXiv preprint arXiv:2408.15998*, 2024a.

Wenhao Shi, Zhiqiang Hu, Yi Bin, Junhua Liu, Yang Yang, See-Kiong Ng, Lidong Bing, and Roy Ka-Wei Lee. Math-llava: Bootstrapping mathematical reasoning for multimodal large language models. *arXiv preprint arXiv:2406.17294*, 2024b.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11888–11898, 2023.

Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024a.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Qwen Team. Qwen2.5: A party of foundation models, September 2024b. URL https://qwenlm.github.io/blog/qwen2.5/.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, et al. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. *arXiv preprint arXiv:2406.16860*, 2024a.

Shengbang Tong, Zhuang Liu, Yuexiang Zhai, Yi Ma, Yann LeCun, and Saining Xie. Eyes wide shut? exploring the visual shortcomings of multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9568–9578, 2024b.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024.

Kirill Vishniakov, Zhiqiang Shen, and Zhuang Liu. Convnet vs transformer, supervised vs clip: Beyond imagenet accuracy. *arXiv preprint arXiv:2311.09215*, 2023.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024a.

Zhenhailong Wang, Joy Hsu, Xingyao Wang, Kuan-Hao Huang, Manling Li, Jiajun Wu, and Heng Ji. Text-based reasoning about vector graphics. *arXiv preprint arXiv:2404.06479*, 2024b.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 2023.

Sifan Wu, Amir Khasahmadi, Mor Katz, Pradeep Kumar Jayaraman, Yewen Pu, Karl Willis, and Bang Liu. Cadvlm: Bridging language and vision in the generation of parametric cad sketches. *arXiv preprint arXiv:2409.17457*, 2024.

Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10371–10381, 2024.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9556–9567, 2024.

Mert Yuksekgonul, Federico Bianchi, Pratyusha Kalluri, Dan Jurafsky, and James Zou. When and why vision-language models behave like bags-of-words, and what to do about it? *arXiv preprint arXiv:2210.01936*, 2022.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11975–11986, 2023.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Peng Gao, et al. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? *arXiv preprint arXiv:2403.14624*, 2024a.

Renrui Zhang, Xinyu Wei, Dongzhi Jiang, Yichi Zhang, Ziyu Guo, Chengzhuo Tong, Jiaming Liu, Aojun Zhou, Bin Wei, Shanghang Zhang, et al. Mavis: Mathematical visual instruction tuning. *arXiv preprint arXiv:2407.08739*, 2024b.

Wenwen Zhuang, Xin Huang, Xiantao Zhang, and Jin Zeng. Math-puma: Progressive upward multimodal alignment to enhance mathematical reasoning. *arXiv preprint arXiv:2408.08640*, 2024.

APPENDIX

A EXTENDED BACKGROUND AND RELATED WORK

Vision-Language MLLMs. While recent iterations of LLMs feature a standardized model architecture and pretraining recipe, MLLMs still often differ in design choices for infusing visual inputs. One popular design is to align *continuous* visual features with the embedding space of a backbone LLM (Liu et al., 2024a;b; Dubey et al., 2024; McKinzie et al., 2024; Tong et al., 2024a; Beyer et al., 2024; AI, 2023; Wang et al., 2024a); another approach involves *tokenizing* visual inputs to be trained jointly with language tokens (Team et al., 2023; Team, 2024a). These modules are often infused with a decoder-only LLM, but others have explored encoder-decoder architectures to integrate a more varied collection of modalities (Alayrac et al., 2022; Mizrahi et al., 2024; Ormazabal et al., 2024; Bachmann et al., 2024). Our study focuses on *decoder* MLLMs with a *continuous* visual encoder, and we carry out an empirical study to explore the effect of synthetic dataset mixture, training recipe, and encoder design (Liu et al., 2022; Radford et al., 2021; Zhai et al., 2023; Oquab et al., 2023).

Geometry-Oriented MLLMs. At the core of these choices is the hardness in designing a module adept in general visual reasoning (McKinzie et al., 2024; Tong et al., 2024a). In this work, we explore the optimal design of MLLMs specialized in low-level visual perception, a crucial aspect for (among other applications) multimodal mathematical understanding (Lu et al., 2023; Zhang et al., 2024a). This paper supplements prior efforts in improving mathematical reasoning (Gao et al., 2023; Zhang et al., 2024b; Zhuang et al., 2024; Li et al., 2024b; Peng et al., 2024; Shi et al., 2024b) with a detailed study on the effect of dataset mixture, curriculum, and visual encoder, to reach a recipe that elicits strong performance on geometric tasks (Kazemi et al., 2023) that require low-level perception.

Evaluating LLVP. Many benchmarks (Rahmanzadehgervi et al., 2024) have reported that frontier-class MLLMs struggle with visual perception tasks, which are prerequisites for applications that emphasize low-level geometric perception (Chen et al., 2024; Fu et al., 2024c), including mathematical (Yue et al., 2024; Lu et al., 2023; Zhang et al., 2024a; Jiang et al., 2024) and spatial reasoning (Chen et al., 2024; Fu et al., 2024b). These findings collectively identify that MLLMs exhibit a language prior (Lin et al., 2023)—a preference of textual inputs over visual inputs—leading to a performance gap between modalities (Wang et al., 2024b; Zhang et al., 2024a; Fu et al., 2024a). Meanwhile, there lacks a high-quality benchmark that evaluates low-level geometric perception in MLLMs, and the Geoperception benchmark represents a first effort to narrow this gap. This type of efforts have led to significant improvements in certain capabilities of MLLMs, such as compositionality of objects (Yuksekgonul et al., 2022; Kong et al., 2023).

Improving LLVP. Many prior works study *data-driven* approaches to improve low-level perception skills. For example, Gao et al. (2023); Li et al. (2024b); Zhuang et al. (2024) employ a standardized supervised finetuning recipe, and optionally adjust the training data mixture. This type of training data is often synthesized from text-only math problems (Lu et al., 2021; Trinh et al., 2024) or via rule-based systems (Kazemi et al., 2023). In parallel, Vishniakov et al. (2023); Shi et al. (2024a); Tong et al. (2024b) have explored the design space of visual encoders for general-purpose vision-language reasoning. We identify best practices over the union of these design spaces, and then train small MLLMs with strong performance in low-level perception tasks.

Lastly, several works (Schick et al., 2024; Surís et al., 2023; Hu et al., 2024) have opted to augment an MLLM with external APIs that process low-level features with specialized vision modules, such as object detection (Redmon et al., 2016), segmentation (Kirillov et al., 2023), and depth estimation (Yang et al., 2024). While these agentic frameworks (Wu et al., 2023) present a promising alternative that directly addresses the shortcomings of visual encoders, they are limited by their scalability to novel use cases, and may be insufficient for precise tool routing that requires low-level perception as a primer (Picard et al., 2023; Wu et al., 2024; Buehler, 2024).

B GEOPERCEPTION BENCHMARK DETAILS

Benchmark Tasks. Although there are many complex geometry problems in textbooks and real-world, the basic constitutions of it is relatively simple: only five axioms can already underpin all further geometric shapes and reasoning steps, introduced by Euclid over two thousand years ago. These axioms involve establishing and extending lines using points (Axioms 1 and 2), constructing circles from a point and a radius (Axiom 3), and defining perpendicularity (Axiom 4) and parallelism (Axiom 5). Additionally, Euclid provided common notions regarding the properties of equality. Accordingly, we define seven tasks in our Geoperception dataset: *PointLiesOnLine* (`POL`), *PointLiesOnCircle* (`POC`), *Parallel* (`PRA`), *Perpendicular* (`PEP`), *Equal* (`EQL`), *AngleClassification* (`ALC`) and *LineComparison* (`LHC`). In geometric diagrams, perpendicularity, parallelism, and equality are often indicated by annotation symbols. Thus, we classify *Parallel*, *Perpendicular*, and *Equal* as annotated geometry understanding. Meanwhile, *PointLiesOnLine*, *PointLiesOnCircle*, *AngleClassification*, and *LineComparison* fall under primitive geometry shape understanding, which includes both logical (*PointLiesOnLine*, *PointLiesOnCircle*) and numerical (*AngleClassification*, *LineComparison*) tasks.

Benchmark Construction. Thanks to the precise annotated logical forms for geometric diagrams from Geometry-3K (Lu et al., 2021), we are able to build a large scale benchmark based on it, focusing just on geometric LLVP without requiring any further reasoning. Four examples from Geoperception are illustrated in Fig. 1.

Data Filtering. Geoperception is sourced from the Geometry-3K (Lu et al., 2021) corpus, which offers precise logical forms for geometric diagrams, compiled from popular high-school textbooks. However, certain points in these logical forms are absent in the corresponding diagrams. To resolve this, we use GPT-4o-mini MLLM to confirm the presence of all points listed in the logical forms. This process filters the 3,002 diagrams to retain 1,584, where at least one logical form fully represents its points in the diagram. A random inspection of 100 annotations reveals only two errors, indicating high annotation accuracy.

Converting Logical Forms Into Questions. We convert logical forms into question-and-answer pairs for each of the seven tasks in Geoperception. In the `Equals` task, for example, we directly convert the logical form (e.g., `Equals(LengthOf(Line(Q, T)), 86)`) into a question-answer pair (e.g., `Q: What is the length of line QT as annotated? A: 86`). For `PointLiesOnLine`, two points on the line are chosen to form the question, with the remaining points on the line as the answer. Similarly, for `PointLiesOnCircle`, we ask which points lie on the circle, using its center as the basis for the question. For `Parallel` and `Perpendicular`, we represent each line by two points and query which other lines are parallel or perpendicular to it. In `AngleClassification`, we ensure the queried angle is in the range of $[10, 80] \cup [100, 170]$ degrees to avoid ambiguity. For `LineComparison`, we ensure that the shorter line is less than 70% of the length of the longer line. Since multiple equivalent questions can be generated for a single logical form (e.g., a line containing five points generates ${}^5P_2$ equivalent questions), we randomly select one to avoid redundancy. Table 3 summarizes the question statistics for each task, as well as the number of images involved. Extended examples from Geoperception are illustrated in Fig. 7.

Statistics. In Table 3, we provide more details on the Geoperception benchmark, such as the number of logic forms present before and after filtering, the number of questions, and the number of images. `AngleClassification` and `LineComparison` are directly derived from points coordinates without filtering.

Evaluation Details. During evaluation, we carefully craft the evaluation prompts for each question type to ensure the clarity, the full prompts can be found in Appendix E. We use greedy sampling during evaluation to get deterministic results. Additionally, GPT-4o-mini without image input is used for generating the random baseline, employing the same textual instructions.

The open source models include Molmo-7B-D (Deitke et al., 2024), Cambrian-1-8B (Tong et al., 2024a), Qwen2-VL-7B (Wang et al., 2024a), Llama-3.2-11B (Dubey et al., 2024), and Pixtral-12B (AI, 2023). The closed-source models include GPT-4o-mini (Achiam et al., 2023), GPT-4o (Achiam et al., 2023), Claude-3.5-Sonnet (Anthropic, 2024), Gemini-1.5-flash (Team et al., 2023),

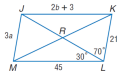
Predicate	# LF Before Filter	# LF After Filter	# Q	# I
PointLiesOnLine	6988	2567	1901	924
PointLiesOnCircle	1966	1240	359	322
Parallel	222	123	106	101
Perpendicular	1111	680	1266	456
Equals	6434	4123	4436	1202
AngleClassification	-	-	2193	1389
LineComparison	-	-	1394	1394

Table 3: Statistics of the five predicates in our Geoperception dataset. Including number of logic forms before filter, after filter and the number of questions and images.

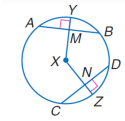
and Gemini-1.5-pro (Team et al., 2023). Additionally, GPT-4o-mini without image input is used for generating the random baseline, employing the same textual instructions. To prevent stretching, all images are padded to square dimensions before being fed into the models. During evaluation of a given question by an MLLM, let G denote the ground truth set of answers, and let P denote the predicted set of answers; then the evaluation score is defined as

$$\text{Evaluation score} = \begin{cases} \dfrac{|P|}{|G|} & \text{if } P \subseteq G, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

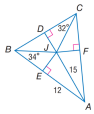
PointLiesOnLine



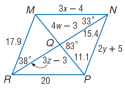
Q: What is the point lying on line JL?
A: R



Q: What is the point lying on line ZX?
A: N

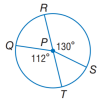


Q: What is the point lying on line AB?
A: E

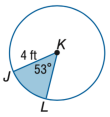


Q: What is the point lying on line RN?
A: Q

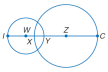
PointLiesOnCircle



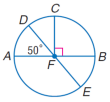
Q: What is the point lying on circle with center P?
A: T, S, R, Q



Q: What is the point lying on circle with center K?
A: L, J

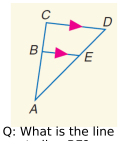


Q: What is the point lying on circle with center Z?
A: X, C

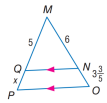


Q: What is the point lying on circle with center F?
A: A, C, B, D, E

Parallel

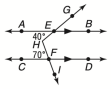


Q: What is the line parallel-to line BE?
A: CD



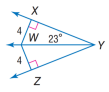
Q: What is the line parallel-to line NQ?
A: OP

Q: What is the line parallel-to line EB?
A: CD

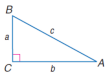


Q: What is the line parallel-to line CD?
A: BE, AB, AE

Perpendicular

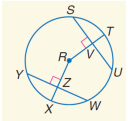


Q: What is the line perpendicular to line ZW?
A: YZ



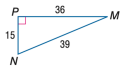
Q: What is the line perpendicular to line CB?
A: AC

Q: What is the line perpendicular to line LF?
A: LM, KM, GH, HJ, KL, GJ

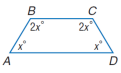


Q: What is the line perpendicular to line VS?
A: RT, TV, RV

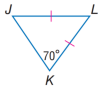
Equals



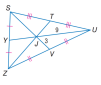
Q: What is the length of line NM as annotated?
A: 39



Q: What is the measure of angle ABC as annotated?
A: 2x

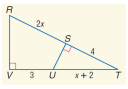


Q: What is the measure of angle JKL as annotated?
A: 70

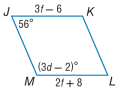


Q: What is the line in the diagram that is equal to line VU?
A: ZV, VZ

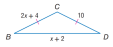
AngleClassification



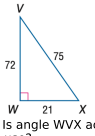
Q: Is angle SUV acute or obtuse?
A: obtuse



Q: Is angle JKL acute or obtuse?
A: obtuse

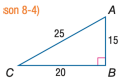


Q: Is angle CBD acute or obtuse?
A: acute

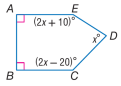


Q: Is angle WVX acute or obtuse?
A: acute

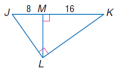
LineComparison



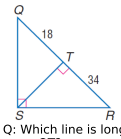
Q: Which line is longer, AB - or AC?
A: AC



Q: Which line is longer, AE - or ED?
A: AE



Q: Which line is longer, JM - or JL?
A: JL



Q: Which line is longer, RQ - or QT?
A: RQ

Figure 7: Examples of our *Geoperception* dataset.

C FUTURE DIRECTIONS.

Our work examines the potential of using synthetic multimodal data to improve MLLM performance in low-level geometric perception tasks. However, there are still directions that remain underexplored:

1. Automatic curriculum learning. Incorporating a more diverse dataset, including varied geometric shapes and different domain dataset, introduces challenges in defining the learning order. Rule based definition and manual curation may become impractical, necessitating automated strategies like hard negative sampling to organize the curriculum based on training loss or testing accuracy. This approach could streamline the process, reduce human effort, provide more suitable and efficient curriculum learning orders.
2. Using a more-diverse training dataset. Currently, the text portion of our synthetic multimodal training data uses a restricted set of templates, and the model trained on such templates could fail to generalize to other question types; it could therefore be beneficial to increase the diversity of our training images as well as the instruction-following formats.
3. Generalizing to other task domains. In this work, our study is focused on data from 2D geometry, as it provides a focused test bed of fundamental tasks. We believe the lessons we learn from this domain can be effectively generalized to a broader set of downstream domains that benefit from high-quality LLVP.

D EXPERIMENT DETAILS IN EMPIRICAL STUDY AND EUCLID TRAINING.

Experimental setting for empirical study. We use *PointLiesOnLine* and *LineComparison* as the test bed tasks for the exploration. For each task, we carefully create three levels with incremental difficulties. We name them as difficulty level easy, medium and hard. Based on the insight from our preliminary experiments, to increase the difficulty levels, for *PointLiesOnLine*, we increase the complexity of geometry shapes as is shown in Fig. 16, for *LineComparison*, we increase the total number of letters in letter pool while mixing geometry shapes. To report stable results, we run the training for three times and report the best run among them (i.e., having the lowest overall training loss or testing accuracy).

Table 4 summarize the visual encoder we use in our empirical study.

Table 4: Summary of Visual Encoders

Model	Params	Objective
ConvNeXt Large@512	200M	CLIP
ConvNeXt XXLarge@512	847M	CLIP
ViT-g/14@224	1.01B	CLIP
ViT-H/14@224	632M	CLIP
ViT-L/14@224	303M	CLIP
SigLIP@224 (ViT)	428M	CLIP-like
DINOv2 Giant@224 (ViT)	1.14B	Self-Sup
DINOv2 Large@224 (ViT)	304M	Self-Sup

Experimental setting for Euclid training. For models, we select the best visual encoder architecture we found in our investigation, ConvNeXt, including ConvNeXt-Large@512 and ConvNeXt-XXLarge@512, and keep the same multimodal connector (2 layers MLP) and LLM (Qwen2.5-1.5B-instruct). The accuracy threshold for advancing training stage θ is set to 0.99. All models are trained on $N = 3$ stages with manually curated geometry shapes and $M = 50$ rounds with $K = 500$ steps in each round, and the batch size is 64 for each training step. The total training dataset volume for both of the models is 1.6M.

Euclid error analysis. We take a deep look into Euclid's prediction on *Geoperception* and find that its performance is hindered when diagrams are heavily annotated. An example is shown in Fig. 8, where a line is annotated by "x", preventing the model from choosing the correct point. We hypothesize that incorporating training data with more diverse annotation types and geometry shapes could help the model with such scenarios.

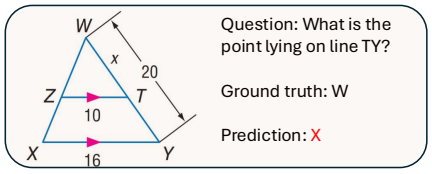


Figure 8: An error case where Euclid fails to predict the correct point on a line, potentially distracted by the annotation "x".

E PROMPTS FOR THE GEOPERCEPTION DATASET EVALUATION

PROMPT TEMPLATE FOR THE POINTLIESONLINE TASK

```
Answer me directly just with the all points lie on the line
mentioned in the question (do not include the point mentioned in
the question).
Answer template:
  (If only one point) The other point is:  "your point".
Or
  (if multiple points) The other points are:  "your points".
For example:
  The other point is:  A
Or
  The other points are:  A, B, C
```

Figure 9: TEMPLATE FOR THE POINTLIESONLINE TASKS

PROMPT TEMPLATE FOR THE POINTLIESONCIRCLE TASK

```
Answer me directly just with the all points lie on the circle
mentioned in the question.
Answer template:
  (If only one point) The point is:  "your point".
Or
  (If multiple points) The points are:  "your points".
For example:
  The point is:  A
Or:
  The points are:  A, B, C
```

Figure 10: TEMPLATE FOR THE POINTLIESONCIRCLE TASKS

PROMPT TEMPLATE FOR THE PARALLEL TASK

```
Answer me directly just with the all lines which are parallel
to the line mentioned in the question (do not include the line
mentioned in the question).
Answer template:
  (If only one line) The line is:  "your line".
Or
  (If multiple lines) The lines are:  "your lines".
For example:
  The line is:  BC
Or:
  The lines are:  BC, DE
```

Figure 11: TEMPLATE FOR THE PARALLEL TASKS

PROMPT TEMPLATE FOR THE PERPENDICULAR TASK

```
Answer me directly just with the all lines which are perpendicular
to the line mentioned in the question (do not include the line
mentioned in the question).
Answer template:
  (If only one line) The line is:  "your line".
Or
  (If multiple lines) The lines are:  "your lines".
For example:
  The line is:  BC
Or:
  The lines are:  BC, DE
```

Figure 12: TEMPLATE FOR THE PERPENDICULAR TASKS

PROMPT TEMPLATE FOR THE EQUALS TASK

```
Answer me directly just with the annotations presented on the
image.
Answer template:
  The annotation is:  "your annotation".
For example:
  The annotation is:  2x+4
Or:
  The annotations is:  90
```

Figure 13: TEMPLATE FOR THE EQUALS TASKS

PROMPT TEMPLATE FOR THE ANGLE CLASSIFICATION TASK

```
Answer me directly just with the classification of the angle
mentioned in the question.
Answer template:
  The angle is:  "your angle".
For example:
  The angle is:  acute
Or:
  The angle is:  obtuse
```

Figure 14: TEMPLATE FOR THE ANGLE CLASSIFICATION TASKS

PROMPT TEMPLATE FOR THE LINECOMPARISON TASK

```
Answer me directly just with the longer line mentioned in the
question.
Answer template:
  The longer line is:  "your line".
For example:
  The longer line is:  BC
Or:
  The longer line is:  DE
```

Figure 15: TEMPLATE FOR THE LINECOMPARISON TASKS

F DETAILS FOR TRAINING DATA ENGINE

In this section, we provide all geometry shapes we use for Euclid training, including the pseudocode for generating text describing the geometry shapes and diagram examples.

Our geometry dataset generation engine is built on AlphaGeometry (Trinh et al., 2024). Given an input formal language describing a geometry shape, the geometry engine is able to render infinite actual geometry images (exampled in Fig. 16 with full metadata (e.g. point coordinates) then create different types of accurate textual annotations to train MLLMs with corresponding images.

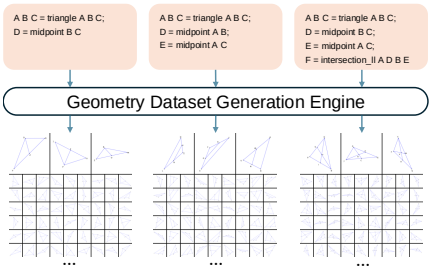


Figure 16: Three geometry logical shapes, of increasing complexity, used in our empirical study. Our geometry image generation engine is able to produce infinite visual instances for each of these logical shapes. All letters are randomly sampled from the alphabet and reassigned to each of the points before drawing.

F.1 PSEUDOCODE FOR TRAINING TEXTUAL DATASET SYNTHESIS

Algorithm 1 Data Synthesis for the POINTLIESONLINE Task

```
Input: data_info, points_set
Output: data
for points_set ∈ data_info do
    for (A, B) ∈ permutations(points_set, 2) do
        all_rest_points ← [p for p in points_set if p not in [A, B]]
        for rest_points ∈ permutations(all_rest_points) do
            verb_agreement ← 'is' if len(rest_points) == 1 else 'are'
            rest_points ← [f"{p}" for p in rest_points]
            rest_points ← sorted(rest_points)
            question ← 'What is the point lying on line ' + A + B + '?'
            answer ← 'The point lying on line ' + A + B + ' ' + verb_agreement + ' ' + ', '.join(rest_points)
            gt ← ''.join(rest_points)
            data ← {'question': question, 'answer': answer, 'gt': gt}
        end for
    end for
end for
```

Algorithm 2 Data Synthesis for the POINTLIESONCIRCLE Task

```
Input: data_info
Output: data
point_set ← random.choice(list(data_info.items()))
center_point ← point_set[0]
target_points ← point_set[1]
target_points ← sorted(target_points)
question ← 'What are the point lying on circle ' + center_point + '?'
answer ← 'The point lying on circle ' + center_point + ' are ' + ', '.join(target_points)
gt ← ''.join(target_points)
data ← {'question': question, 'answer': answer, 'gt': gt}
```

Algorithm 3 Data Synthesis for the ANGLECLASSIFICATION Task

```
Input: data_info
Output: data
angle ← data_info
angle_options ← [f'{angle[1][0]}{angle[1][1]}{angle[1][2]}',
  f'{angle[1][2]}{angle[1][1]}{angle[1][0]}']
angle_letter ← random.choice(angle_options)
angle_class ← 'acute' if angle[0] < 90 else 'obtuse'
question ← 'Is angle ' + angle_letter + ' acute or obtuse?'
answer ← 'Angle ' + angle_letter + ' is ' + angle_class
gt ← angle_class
data ← {'question': question, 'answer': answer, 'gt': gt}
```

Algorithm 4 Data Synthesis for the LINECOMPARISON Task

```
Input: data_info
Output: data
names ← [data_info[0][1], data_info[1][1]]
lengths ← [data_info[0][0], data_info[1][0]]
if lengths[0] > lengths[1] then
   longer_name, shorter_name ← names[0], names[1]
else
   longer_name, shorter_name ← names[1], names[0]
end if
data ← [
  { 'question': 'Which line is longer, ' + longer_name + ' or '
  + shorter_name + '?',
    'answer': 'The longer line is ' + longer_name,
    'gt': longer_name },
   { 'question': 'Which line is longer, ' + shorter_name + ' or
  ' + longer_name + '?',
    'answer': 'The longer line is ' + longer_name,
    'gt': longer_name }
]
```

Algorithm 5 Data Synthesis for the PARALLEL Task

```
Input: data_info
Output: data
points_set ← data_info
for line_points ∈ points_set do
    for (A, B) ∈ permutations(line_points, 2) do
        all_rest_lines ← [p for p in points_set if p != line_points]
        gts ← [''.join(
            f'{p}' for line in all_rest_lines for p in line)
        ]
        rest_point_pairs ← []
        for rest_line ∈ all_rest_lines do
            C, D ← random.sample(rest_line, 2)
            rest_point_pairs.append([C, D])
        end for
        all_possible_answer ← ', '.join(
            [f'{C}{D}' for C, D in rest_point_pairs]
        )
        verb_agreement ← 'is' if len(rest_point_pairs) == 1 else 'are'
        question ← 'What is the line parallel to line ' + A + B + '?'
        answer ← (
            'According to the diagram, the line parallel to ' +
            A + B + verb_agreement + all_possible_answer
        )
        gt ← ', '.join(gts)
        data ← {
            'question': question, 'answer': answer, 'task': task, 'gt': gt
        }
    end for
end for
```

Algorithm 6 Data Synthesis for the PERPENDICULAR Task

```
Input: data_info
Output: data
source_lines, target_lines ← data_info
all_possible_answer ← []
gts ← target_lines                    ▷ Randomly choose two points from each target line
for target_line ∈ target_lines do
    C, D ← random.sample(target_line, 2)
    all_possible_answer.append(f'{C}{D}')
end for
verb_agreement ← 'is' if len(all_possible_answer) == 1 else 'are'
for (A, B) ∈ permutations(source_line, 2) do
    question ← 'What is the line perpendicular to line ' + A + B + '?'
    answer ← (
        'According to the diagram, the line perpendicular to ' +
        A + B + verb_agreement + ', '.join(all_possible_answer
    )
    gt ← ', '.join(gts)
    data ← {
        'question': question, ', 'answer': answer, ''ask': task, 'gt': gt
    }
end for
```

Algorithm 7 Data Synthesis for the EQUAL Task

```
Input: data_info
Output: data
statement, content ← data_info.split(';')
if statement == 'angles_value' then
    angle_letter, angle_measure ← content.split('=')
    angle_letter ← random.choice([angle_letter, angle_letter[::-1]])
    question ← 'What is the measure of angle ' + angle_letter + ' as annotated?'
    answer ← 'Angle ' + angle_letter + ' is annotated as ' + angle_measure
    gt ← angle_measure
else if statement == 'segments_value' then
    segment_letter, segment_length ← content.split('=')
    segment_letter ← random.choice([segment_letter, segment_letter[::-1]])
    question ← 'What is the length of line ' + segment_letter + ' as annotated?'
    answer ← 'Line ' + segment_letter + ' is annotated as ' + segment_length
    gt ← segment_length
else if statement == 'angles' then
    angle1, angle2 ← content.split('=')
    angle1 ← random.choice([angle1, angle1[::-1]])
    angle2 ← random.choice([angle2, angle2[::-1]])
    query_angle ← random.choice([angle1, angle2])
    answer_angle ← angle2 if query_angle == angle1 else angle1
    question ← 'What is the angle in the diagram that is equal to angle ' + query_angle
    answer ← 'Angle ' + query_angle + ' is equal to angle ' + answer_angle
    gt ← answer_angle
else if statement == 'segments' then
    segment1, segment2 ← content.split('=')
    segment1 ← random.choice([segment1, segment1[::-1]])
    segment2 ← random.choice([segment2, segment2[::-1]])
    query_segment ← random.choice([segment1, segment2])
    answer_segment ← segment2 if query_segment == segment1 else segment1
    question ← 'What is the segment in the diagram that is equal to segment ' + query_segment
    answer ← 'Segment ' + query_segment + ' is equal to segment ' + answer_segment
    gt ← answer_segment
end if
data ← {
        'question': question, 'answer': answer, 'task': task, 'gt': gt
}
```

F.2 GEOMETRY SHAPES USED FOR EUCLID TRAINING

GEOMETRY SHAPE GENERATION CODE

```
PointLiesOnLine
  (stage 1) A B C = triangle A B C; D = midpoint B C
  (stage 1) A B C = triangle A B C; D = midpoint B C; O = circle O A B C
  (stage 2) A B C = triangle A B C; D = midpoint A B; E = midpoint A C
  (stage 2) A B C = triangle A B C; D = midpoint A B; E = midpoint A C; O = circle O A B C
  (stage 3) A B C = triangle A B C; D = midpoint B C; E = midpoint A C; F = intersection_ll A D B E
  (stage 3) A B C = triangle A B C; D = midpoint B C; E = midpoint A C; F = intersection_ll A D B E; O
= circle O A B C
PointLiesOnCircle
  (stage 1) A B = segment A B; C = on_circle C A B
  (stage 1) A B = segment A B; C = on_circle C A B; D = on_circle D A B
  (stage 1) A B = segment A B; C = on_circle C A B; D = on_circle D A B; E = on_circle E A B
  (stage 1) A B = segment A B; C = on_circle C A B; D = on_circle D A B; E = on_circle E A B; F =
on_circle F A B
  (stage 1) A B = segment A B; C = on_circle C A B; D = on_circle D A B; E = on_circle E A B; F =
on_circle F A B; G = on_circle G A B
  (stage 2) A B = segment A B; C = on_circle C A B; D = midpoint A B
  (stage 2) A B = segment A B; C = on_circle C A B; D = midpoint A B
  (stage 2) A B = segment A B; C = on_circle C A B; D = midpoint A B; E = on_circle E A B
  (stage 2) A B = segment A B; C = on_circle C A B; D = midpoint A B; E = on_circle E A B; F =
on_circle F A B
  (stage 2) A B = segment A B; C = on_circle C A B; D = midpoint A B; E = on_circle E A B; F =
on_circle F A B; G = on_circle G A B
  (stage 2) A B = segment A B; C = on_circle C A B; D = midpoint A B; E = on_circle E A B; F =
on_circle F A B; G = on_circle G A B; H = on_circle H A B
  (stage 3) A B = segment A B; C = on_circle C A B; D = midpoint A B; E = midpoint A C
  (stage 3) A B = segment A B; C = on_circle C A B; D = midpoint A B; E = midpoint A C; F = on_circle
F A B
  (stage 3) A B = segment A B; C = on_circle C A B; D = midpoint A B; E = midpoint A C; F = on_circle
F A B; G = on_circle G A B
  (stage 3) A B = segment A B; C = on_circle C A B; D = midpoint A B; E = midpoint A C; F = on_circle
F A B; G = on_circle G A B; H = on_circle H A B
  (stage 3) A B = segment A B; C = on_circle C A B; D = midpoint A B; E = midpoint A C; F = on_circle
F A B; G = on_circle G A B; H = on_circle H A B; I = on_circle I A B
  (stage 3) A B = segment A B; C = on_circle C A B; D = midpoint A B; E = on_circle E A B; F =
on_circle F A B; G = on_circle G A B; H = on_circle H A B; I = midpoint B C
  (stage 3) A B = segment A B; C = on_circle C A B; D = midpoint A B; E = midpoint B C
  (stage 3) A B = segment A B; C = on_circle C A B; D = midpoint A B; E = lc_tangent E C A
  (stage 3) A B = segment A B; C = on_circle C A B; D = midpoint A B; E = on_circle E A B; F =
on_circle F A B; G = on_circle G A B; H = lc_tangent H C A
AngleClassification
  (stage 1) A B C = triangle A B C
  (stage 3) A B C = triangle A B C; D = midpoint B C
  (stage 3) A B C = triangle A B C; D = midpoint B C; E = midpoint A C; F = intersection_ll F A D B E
LengthComparison
  (stage 1) A B C = triangle A B C
  (stage 2) A B C = triangle A B C; D = midpoint B C
  (stage 3) A B C = triangle A B C; D = midpoint A B; E = midpoint A C
Parallel
  (stage 1) A B C = triangle A B C; D = midpoint A B; E = midpoint A C
  (stage 1) A B C = triangle A B C; D = midpoint A B; E = midpoint A C
  (stage 1) A B C = triangle A B C; D = midpoint A B; E = midpoint A C
  (stage 2) A B C = triangle A B C; D = parallelogram A B C D
  (stage 3) A B C = triangle A B C; D = midpoint A B; E = midpoint A C; F = midpoint B C
Perpendicular
  (stage 1) A B C = triangle A B C; D = foot A B C
  (stage 1) A B C = r_triangle A B C
  (stage 1) A B = segment A B; C = eq_triangle C A B; D = eq_triangle D A B; E = on_circle E A B
  (stage 2) A B C = triangle A B C; D = foot A B C; E = foot C A B
  (stage 2) A B C = r_triangle A B C; D = foot A B C
  (stage 2) A B C = triangle A B C; O = circle A B C; D = foot O A B; E = foot O C A
  (stage 3) A B C D = rectangle A B C D; E = intersection_ll A C B D
  (stage 3) A B C = triangle A B C; O = incenter A B C; D = foot O A C; E = foot O B C; F = foot O A
B
  (stage 3) A B C = r_triangle A B C; D = foot A B C; E = foot D A B
  (stage 3) A B C = triangle A B C; D = foot A B C; E = foot C A B; F = foot B A C
Equal
  (stage 1) A B C = triangle A B C; D = midpoint C B
  (stage 1) A B C = triangle A B C; D = midpoint C B; O = circle O A B C
  (stage 1) A B C = triangle A B C; D = angle_bisector B A C, on_line D C B
  (stage 2) A B C = triangle A B C; D = midpoint A B; E = midpoint A C
  (stage 2) A B C = triangle A B C; D = midpoint A B; E = midpoint A C; O = circle O A B C
  (stage 2) A B C = triangle A B C; D = midpoint A B; E = midpoint A C

  (stage 3) A B C = triangle A B C; O = circle A B C; D = on_circle D O C, angle_bisector C A B
```

Figure 17: GEOMETRY SHAPE GENERATION CODE FOR EUCLID TRAINING

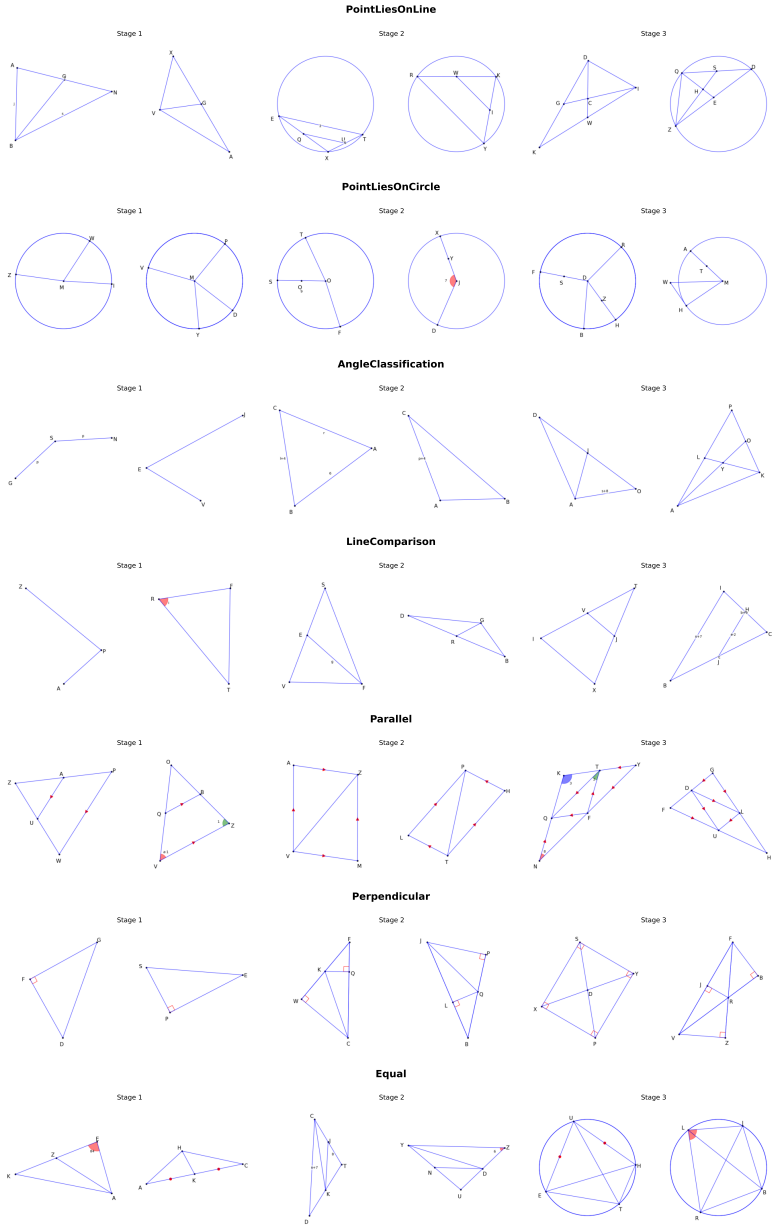


Figure 18: Examples of the geometry diagrams used to train Euclid, the diagrams are generated by our dataset engine.

G ADDITIONAL EXPERIMENTAL RESULTS

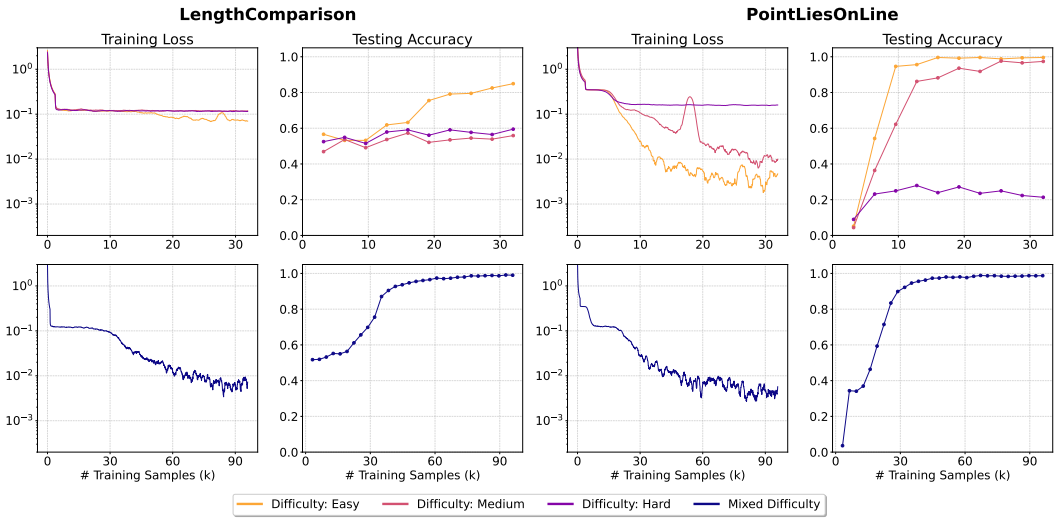


Figure 19: Result of our preliminary experiments, we use a standard setting of MLLMs: an OpenAI-CLIP@224 as visual encoders (Radford et al., 2021), two-layer MLP as multimodal connector and Qwen-2.5-1.5B as language model. We find that the model can reach convergence in some of the easy tasks, while struggle to learn hard tasks. We also find mixed training is better than separate training, given the same amount of training data in each difficulty level.

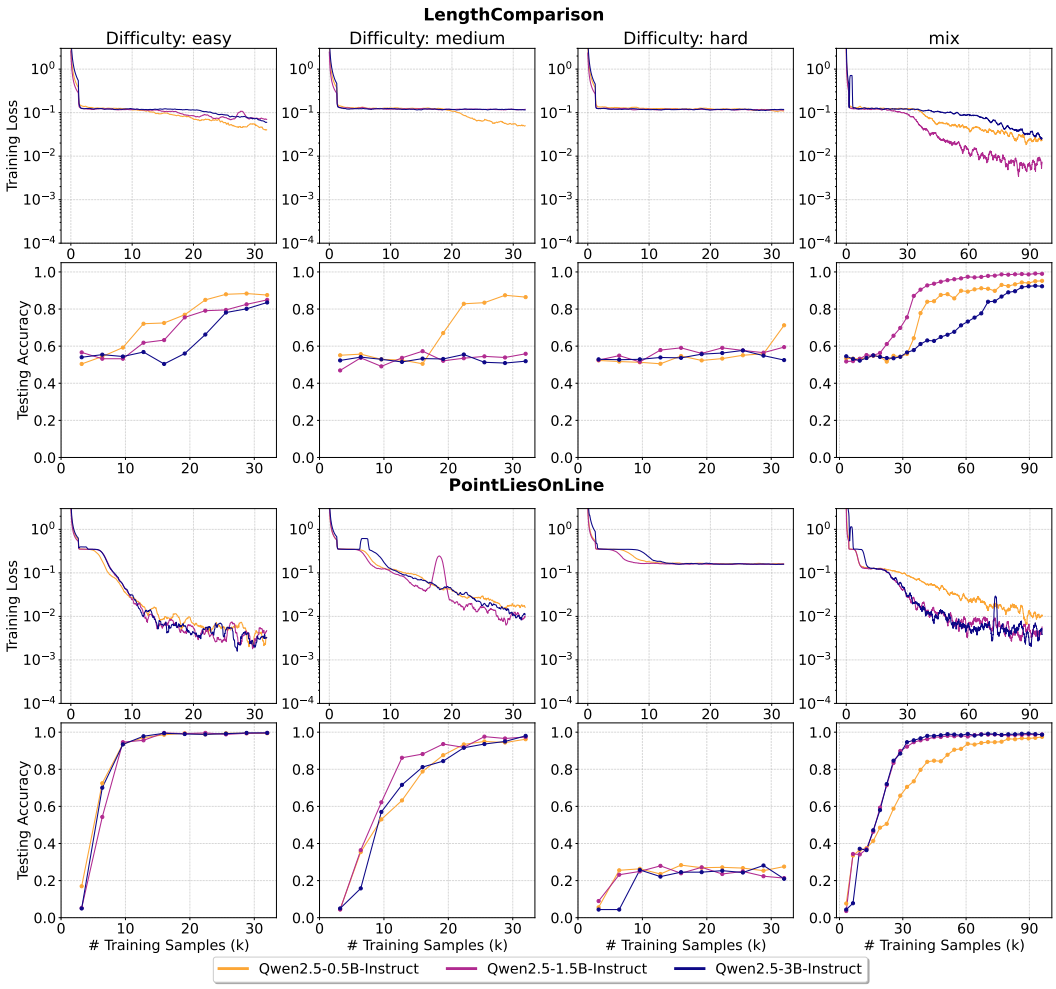


Figure 20: The complete result of the effect of LLM size. The finding is similar with Fig. 2.

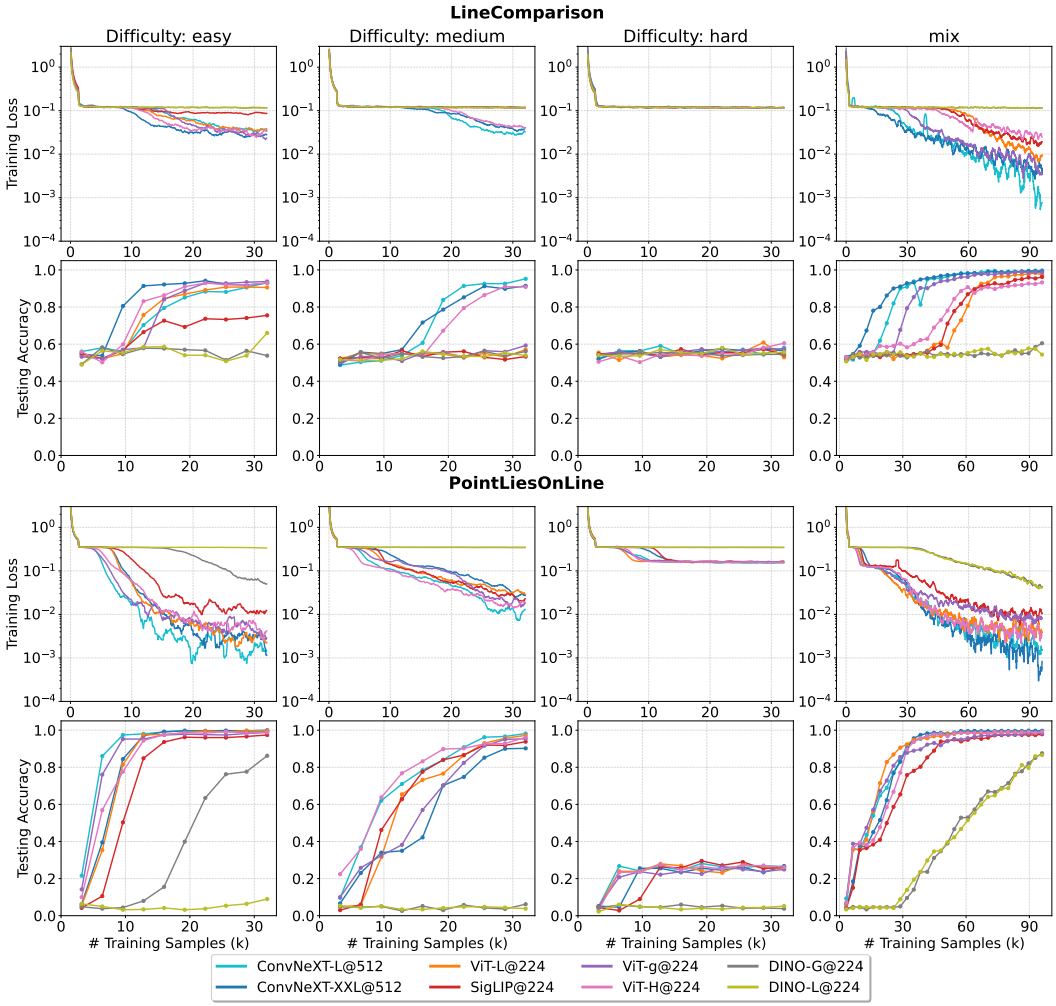


Figure 21: The complete result of the effect of different visual encoders. The finding is similar with Fig. 3.

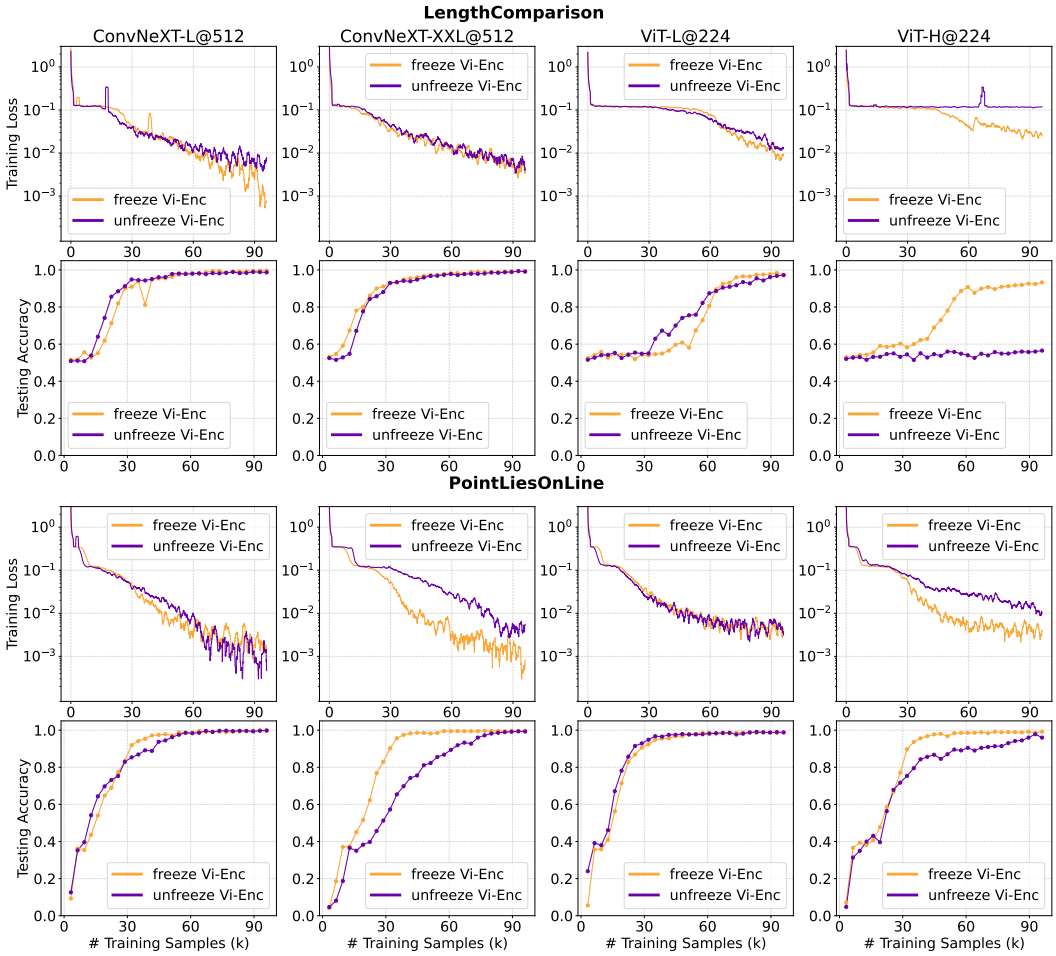


Figure 22: The complete result of the effect of tuning visual encoders. The finding is similar with Fig. 4.