Recipe and Analysis for Large Multimodal Model Pre-training

Hengtao Li Cong Chen Chenchen Jing Mingyang Zhang
Weian Mao Zhouhang Zhu Bince Qu Hao Chen Bo Zhang Chunhua Shen

Zhejiang University, China

Abstract

We present HawkLlama, an open-source multimodal large language model based on Llama-3, trained using the NeMo framework with the aim for real-world vision and language comprehension. To improve the quality of visual tokens, we feed high-resolution images into the model. An efficient vision-language connector, designed to capture high-resolution details without increasing the number of visual tokens, helps reduce the training overhead associated with high-resolution images. For model training, we utilize a mixed dataset specifically curated for instruction tuning, which contains both multimodal and language-only data. We leverage the NeMo framework for our model development to facilitate 3D parallelism and future scalability to larger models. Throughout the training phase, we observe that the data sampling strategy influences loss computation, and Transformer's attention predominantly concentrates on initial tokens. Upon evaluation on popular benchmarks, HawkLlama achieves notable performance, surpassing state-of-the-art models of comparable size.

Models and code are available at: `https://github.com/aim-uofa/VLModel`

1 Introduction

Large language models (LLMs) [OpenAI, 2022, 2023a, Anthropic, 2023, Touvron et al., 2023a,b, Bai et al., 2023a, Du et al., 2022, DeepSeek-AI, 2024] built upon the foundation of Transformer [Vaswani et al., 2017] have had a profound impact on the AI community and human society, owing to their remarkable capabilities in text understanding and generation. Multimodal large Language models (MLLMs), a.k.a. Vision and Language models (VLMs) [OpenAI, 2023b, Team et al., 2023, Liu et al., 2024b, 2023b, 2024a, Zhang et al., 2023, Dong et al., 2024, Bai et al., 2023b, Sun et al., 2023b,a, Xu et al., 2024, Wang et al., 2023, Team et al., 2024, Team, 2024], extending beyond the base of LLM, have further integrated visual perception abilities, effectively endowing LLMs with the capability to see. Compared to purely textual inputs, multimodal inputs offer a more accurate representation of the real world's complex information landscape. Therefore, MLLMs are anticipated to find universal application in downstream visual tasks and represent a pathway towards realizing embodied AI and Artificial General Intelligence (AGI).

Currently, MLLMs can be categorized into non-native multimodal and native multimodal types. Non-native MLLMs [Liu et al., 2024b, Bai et al., 2023b, Dong et al., 2024] are typically bulit on the image and language components trained separately before performing multimodal fusion, whereas native MLLMs [Team et al., 2023, 2024, Team, 2024] are trained on multimodal data from scratch. Due to the substantial resources required for training native MLLMs and the current absence of open-source native MLLMs within the community, most research is conducted on non-native MLLMs, which similarly constitutes the focal point of our work.

With the increasing availability of both open-source and proprietary MLLMs, it is crucial to identify which MLLMs excel in performance and understand what factors contributing to their outstanding performance. Evaluating the performance of an MLLM typically involves considering both the speed and quality of generation. The speed of generation is primarily dictated by the model's computational complexity. The quality of generation is typically reflected in benchmark metrics and the actual user experience. It's determined by various factors, including the ability of the language components for image comprehension and text generation, the ability of the vision components to encode image information, the quality and amount of training data, among others. When developing non-native MLLMs, people commonly employ LLMs pretrained on large-scale text datasets for the language component, resulting that the capabilities of the pretrained LLM directly influence the text processing abilities of the MLLM. For the visual components, there is a preference for utilizing more advanced image encoders [Liu et al., 2023b, Lu et al., 2024] and high-resolution inputs [Liu et al., 2024a, Xu et al., 2024] to enhance the quality of visual tokens. However, high-resolution inputs can lead to an increase in visual tokens, significantly slow down the training process. Certain techniques [Zhu et al., 2023, Bai et al., 2023b] employ resamplers to decrease the visual token count, but these resamplers risk undermining the visual information's quality and integrity. Regarding training data, researchers prioritize assembling datasets that are broader in diversity, larger in scale, and superior in quality for model training, but not all of them can be accessed publicly.

In this work, we aim to provide a comprehensive and transparent training pipeline, methodically exploring how to train a leading-edge MLLM. This encompasses model architecture, data construction, training strategies, infrastructure, and more. We choose Llama3 [Meta, 2024] as the language component since its tokenizer has a larger vocabulary size and it's trained on a large amount of data, and SigLIP [Zhai et al., 2023] as the visual component due to its impressive performance and higher native resolution support compared to CLIP [Radford et al., 2021]. To make use of high-resolution information and reduce training overhead, we design a efficient vision-language connector that compresses the number of tokens while preserving high-resolution visual information and spatial structure. To efficiently train models on clusters and support the training of larger models in the future, we develop on the NeMo framework which supports 3D parallelism. To our knowledge, there have been no reported benchmark metrics for MLLMs trained on NeMo [Harper et al.] previously. Initially, we train our model on a relatively small dataset to identify the most effective training strategies. Subsequently, we proceed to train our model on an instruction tuning dataset that we have constructed, comprising publicly available image-text datasets. Throughout the training process, we encounter several intriguing findings, such as the impact of the sampling strategy on loss computation and the observation that a significant portion of attention is concentrated on the initial few tokens.

Building upon the aforementioned work, we introduce HawkLlama, an advanced MLLM. Hawk-Llama is trained on an instruction tuning dataset comprising 3.15 million samples, and is capable of processing high-resolution inputs without introducing an excessive number of visual tokens. We evaluated our model across multiple public benchmarks, including 38.7% on MMMU, 71.2% on SEEDBench, 70.5% on MMBench-EN, etc. The results demonstrate our model outperforms the state-of-the-art model LLaVA-Next. Our results are fully reproducible and we open-source our model.

2 Approach

2.1 Model Architecture

HawkLlama consists of three key components: a large language model, a vision encoder and a vision-language connector. The vision encoder converts RGB images into visual embeddings, which are then mapped to the space of linguistic modality through the vision-language connector. Finally, the visual embeddings are concatenated with text embeddings and fed into the LLM for text generation.

Large Language Model. Previous work [Laurençon et al., 2024] has demonstrated that the abilities of the LLM will have a significant impact on the MLLM's performance. HawkLlama adopts the lastest Llama3-8B as its large language model. Llama3 leverages a tokenizer featuring a 128K-token vocabulary and is trained on more than 15T tokens, approximately seven times larger than the tokens Llama2 used for training, thereby yielding a substantial performance boost.

Vision Encoder. HawkLlama utilizes the vision encoder of SigLIP [Zhai et al., 2023] with SoViT-400m [Alabdulmohsin et al., 2024] architecture instead of CLIP [Radford et al., 2021] for visual processing. Compared to CLIP, SigLIP natively supports higher resolutions (384 vs 336) and possesses superior image encoding capabilities. Specifically, SigLIP employs a sigmoid loss on

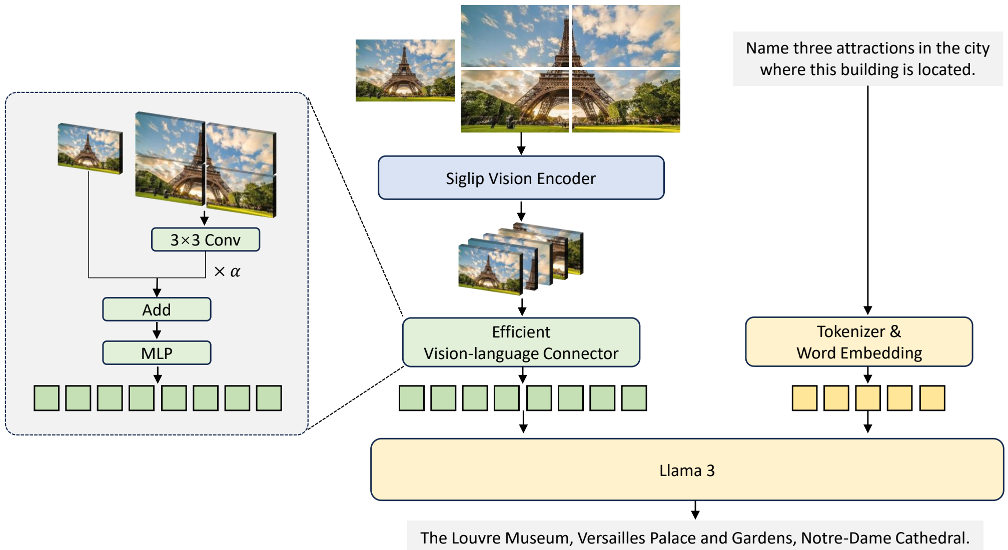


Figure 1: Overall architecture of HawkLlama. Low and high-resolution images are encoded into visual tokens by the vision encoder and an efficient vision-language connector, after which they are combined with text tokens and input into the LLM for next token prediction. The left side of the image shows the details of the efficient vision-language connector.

solely image-text pairs, enabling the utilization of a larger training batch size. This approach leads to enhanced its zero-shot capability on image recognition. SigLIP is capable of processing input images with dimensions of 384×384, encoding them into 729 visual tokens.

Vision-language Connector. The vision-language connector serves as a crucial intermediary, facilitating the integration of visual and textual embeddings. Prior studies [Zhu et al., 2023, Dai et al., 2023, Bai et al., 2023b, Laurençon et al., 2024] have implemented a resampler to distill embeddings-of-interest from the entirety of visual embeddings. While those methods effectively reduce the number of visual tokens, they potentially compromise spatial information and may result in the loss of certain semantic details. Conversely, some other approaches [Liu et al., 2024b, 2023b, 2024a] employ a straightforward linear layer or MLP to map visual embeddings into the textual domain. This strategy preserves the entirety of visual information from the vision encoder, delegating to the LLM the task of determining which features are pertinent. However, this approach may incur significant computational costs, especially for inputs of high resolution. For example, LLaVA-Next [Liu et al., 2024a] inputs both low-resolution and high-resolution images to the model simultaneously, resulting in significant computational overhead during training. The number of visual tokens in LLaVA-Next reaches up to 2880, which is five times that of LLaVA 1.5 [Liu et al., 2023b]. In most conversational scenarios, the visual tokens of LLaVA-Next occupy the majority of the LLM input, noticeably slowing down the training speed.

To provide the model with both the capability to perceive details and reduce training and inference cost, we designed a multimodal connector to compress visual features. We take high-resolution images of 768×768 and low-resolution images of 384×384 as model inputs. The low-resolution images are directly encoded by the visual encoder. The high-resolution images are encoded after being segmented. The high-resolution features are reassembled according to their original positions in the image, followed by a 3×3 convolutional layer to downsample the features. These features are multiplied by a ratio α, added to the low-resolution features and then input into an MLP to obtain the final visual embeddings. The parameter α, initially set to zero, is a learnable coefficient that undergoes optimization throughout the training process. It is a kind of residual connection controlling the proportion of high-resolution features within the visual tokens. With the convolutional layer and α, the model is expected to learn how to merge high-resolution features into the visual tokens without compromising the high-resolution processing ability. The process above can be formulated as

$$F_{\text{vision}} = \text{MLP}(F_{\text{low}} + \alpha \cdot \text{Conv}(F_{\text{high}})), \quad (1)$$

Strategy	MMB-EN dev	MMMU val	SEEDBench
LLaVA 1.5	66.4	33.3	65.1
+ Replace vision backbone from CLIP to SigLIP	68.6	34.3	67.0
+ Replace LLM backbone from Vicuna 1.5 7B to Llama3 8B	69.3	34.6	67.6
+ Finetune vision encoder in stage 2	71.2	36.4	70.6

Table 1: Impacts of different training strategy on the evaluation results on MMBench-EN, MMMU and SEEDBench.

where F_{low}, F_{high} and F_{vision} represents low-resolution and high-resolution features and final visual tokens.

Through the aforementioned decisions and designs, our model is equipped with robust high-resolution processing capabilities, enabling it to comprehend visual information and generate outputs following instructions.

2.2 Training Pipeline

We train the model following a two-stage pipeline: the vision-language alignment stage and the supervised fine-tuning stage.

Vision-language Alignment. During this stage, our goal is to align the embeddings of the vision encoder and the LLM, enabling the LLM to comprehend the information contained within images. We utilize the LLaVA-Pretrain dataset [Liu et al., 2023b] for training. The LLaVA-Pretrain dataset consists of 558k image-text pairs, with the text serving as descriptions of the images. In this phase, we freeze both the vision encoder and the LLM, focusing solely on training the vision-language connector. Similar to most LLMs, we train HawkLlamato predict the next token, employing the cross-entropy loss function as the objective function to optimize the network. Only tokens of the answer part is utilized in the loss calculation.

Supervised Fine-tuning. In this stage, we fine-tune the model using instruction-based conversational data to enhance the model's capability for instruction following and engaging in multi-turn continuous dialogue. During this phase, we train the model with the mixed data mentioned in Section 3, which is a combination of vision-language data and pure-language data, and apply conversational templates. Following [Liu et al., 2024a, Lu et al., 2024] we perform fully fine-tuning on the entire model. Optimization object is the same with that in the first stage.

In formulating the aforementioned training strategy, we are faced with several decisions that need to be made, such as selecting which pretrained weights to initialize the model with and deciding which parameters to optimize during model training. To identify the most effective training strategy, we have progressively iterated our model starting from LLaVA 1.5. All models in this section are trained on the pretraining and fine-tuning datasets of LLaVA 1.5. As shown in Table 1, our LLaVA 1.5 baseline model achieved an accuracy of 66.4 on MMBench-EN. After replacing the vision encoder from CLIP to SigLIP, the model saw an accuracy improvement of +2.2, indicating that the quality of visual features significantly impacts the evaluation results. Subsequently, replacing the LLM from Vicuna 1.5 7B to Llama3 8B resulted in only a +0.7 increase in accuracy. Finally, incorporating the optimization of visual encoder parameters during the supervised fine-tuning phase led to a +1.9 improvement in accuracy, due to better aligning the visual and textual embeddings. Based on these results, we adopted such training strategy.

2.3 Training Techniques

2.3.1 Modality Grouping

In our training process, we implemented a modality grouping strategy to effectively organize the training data. To achieve this, we utilized a specialized data sampler designed to ensure that all training data within a single gradient backpropagation step are either exclusively multimodal data or purely textual data. This approach guarantees that the model processes homogeneous data types during each gradient update. By organizing the data in this manner, we aim to preserve the language priors inherent in the language model. The consistent exposure to homogeneous data types during backpropagation helps the model maintain its learned linguistic structures and nuances, leading to more stable and accurate performance.

2.3.2 Group Similar Length

In our experiments, we observed that enabling gradient accumulation can lead to imbalances in the loss per token. Specifically, considering two steps in one gradient accumulation, with token lengths L_1 and L_2 respectively, the average loss per token for each step is calculated as follows

$$\mathcal{L}_{\mathrm{k}} = \frac{\sum_{i=1}^{L_k} \ell_i}{L_k}, \quad k = 1, 2. \tag{2}$$

Here, $\mathcal{L}_{\mathrm{k}}$ represent the average loss per token for different steps. The terms ℓ_i denote the individual loss values for each token in the respective steps. The overall average loss during gradient accumulation is then computed as

$$\mathcal{L}_{\text{overall}} = \frac{\mathcal{L}_1 + \mathcal{L}_2}{2} = \frac{\left(\frac{\sum_{i=1}^{L_1} \ell_i}{L_1}\right) + \left(\frac{\sum_{j=1}^{L_2} \ell_j}{L_2}\right)}{2}. \tag{3}$$

However, the target loss should be the token-length-weighted loss, calculated as follows

$$\mathcal{L}_{\text{weighted}} = \frac{\sum_{i=1}^{L_1} \ell_i + \sum_{j=1}^{L_2} \ell_j}{L_1 + L_2} \tag{4}$$

The discrepancy arises because the average per-step loss does not account for the different token lengths in each step, leading to an inaccurate representation of the true average loss. To mitigate this issue, we adopted a strategy of grouping tokens of similar lengths. By ensuring that the number of tokens in each step during gradient accumulation is comparable, we can adjust the loss calculation to better approximate the length-weighted loss. This strategy can be represented as

$$\mathcal{L}_{\text{grouped}} \approx \frac{\sum_{k=1}^{n} \ell_{\text{grouped},k}}{\sum_{k=1}^{n} L_{\text{grouped},k}}. \tag{5}$$

In this equation, $\ell_{\text{grouped},k}$ represents the loss for each token in the grouped set, and $L_{\text{grouped},k}$ denotes the token length for each group k. Additionally, the grouping similar length strategy reduces the number of token paddings required in different batches within a step, thus improving training efficiency and reducing overall training time. This optimization enhances the accuracy of the gradient accumulation process and leads to more efficient resource utilization during training.

2.3.3 Sink Token

During the inference phase, we analyzed the similarity matrix from the self-attention mechanism in the final layer of our model. Our findings indicate that, despite the `[BOS]` (Beginning of Sequence) token lacking intrinsic semantic content, it contributes over 50% of the weight in the prediction of subsequent tokens, which is significantly higher than the contribution of any other individual token. This observation aligns with the phenomenon described in StreamingLLM [Xiao et al., 2023], where the initial token functions as an "Attention Sink".

Setting	MMB-EN dev
w `[BOS]` token	66.4
wo `[BOS]` token	61.1

Table 2: The impact of the presence of a `[BOS]` at the start of a conversation on model performance during training.

The `[BOS]` token's visibility to all subsequent tokens allows it to be more readily learned as an Attention Sink, stabilizing the distribution of attention scores post-SoftMax activation, thereby enhancing inference accuracy. To empirically validate the effectiveness of the `[BOS]` token as an Attention Sink, we conducted ablation studies using the LLaVA 1.5 [Liu et al., 2023b] model. Specifically, we performed comparative experiments by including and excluding the `[BOS]` token during the training phase. The experimental results, summarized in Table 2, indicate a +5.3 improvement on the MMBench [Liu et al., 2023c] benchmark when the `[BOS]` token is present, thus affirming that models incorporating a sink token exhibit superior convergence and stability.

Class	Task	Dataset	Ratio
General Multi-modality	Caption	ShareGPT4V [Chen et al., 2023]	25.0 %
	Conversation	LAION-GPTV [LAION, 2023]	
	Conversation	textOCR-GPT4V [Carter, 2024]	
	Grounding	RefCOCO [Kazemzadeh et al., 2014]	
	General QA	VQAv2 [Goyal et al., 2017]	
	General QA	GQA [Hudson and Manning, 2019]	
	General QA	IconQA [Lu et al., 2021]	
	General QA	Visual7w [Zhu et al., 2016]	
	General QA	VSR [Liu et al., 2023a]	
	General QA	In-house data	
	Knowledge QA	OK-VQA [Marino et al., 2019]	
Table and chart	Chart	ScreenQA [Hsiao et al., 2022]	30.8%
	Chart	ChartQA [Masry et al., 2022]	
	Chart	DVQA [Kafle et al., 2018]	
	Chart	PlotQA [Methani et al., 2020]	
	Math	Geo170K [Gao et al., 2023]	
	Science	ScienceQA [Lu et al., 2022]	
	Science	AI2D [Kembhavi et al., 2016b]	
	Science	TQA [Kembhavi et al., 2017]	
	Document	DocVQA [Clark and Gardner, 2018]	
	Table, Chart, Document	Ureader [Ye et al., 2023]	
	Table, Chart, Document	The Cauldron [Laurençon et al., 2024]	
OCR	QA	TextVQA [Singh et al., 2019]	3.6%
	QA	OCRVQA [Mishra et al., 2019]	
	QA	InfoVQA [Mathew et al., 2022]	
		Synthdog (en&zh) [Kim et al., 2022]	
Text-only SFT	Math	MetaMathQA [Yu et al., 2023]	40.1%
	Math	MathInstruct [Yue et al., 2023b]	
	Math	Orca-math [Mitra et al., 2024]	
	Math	Camel [Li et al., 2023b]	
	Code	Codeit [Raheja et al., 2023]	
	Logical reasoning	Logiqa [Liu et al., 2020]	
	Conversation	ShareGPT [Zheng et al., 2024]	
	Chinese QA	COIG [Bai et al., 2024]	

Table 3: The composition of our instruction tuning dataset.

These findings suggest that the inclusion of a special `[BOS]` token as an Attention Sink can significantly enhance the performance of multimodal models during both training and inference. Therefore, we recommend this approach for improving model stability and performance.

3 Data Construction

Quantity, quality, and diversity of training data are critical factors for multi-modal large language models. Nowadays, there are various types of image-text data available on the Internet, including image recognition, visual question answering, chart and table understanding, text transcription, visual reasoning, etc. We have collected multiple open-source datasets from image-text tasks available publicly, converting them to a shared format and ultimately compiling them into an instruction-tuning dataset with a very broad distribution. Our dataset includes the following data:

General multi-modality data. Multi-modal data in the general domain is fundamental to multi-modal large language models. We first use including widely-used open-source datasets, such as ShareGPT4V [Chen et al., 2023], LAION-GPT4V [LAION, 2023], textOCR-GPT4V [Carter, 2024],

	Stage 1	Stage 2
Learning rate	1.0×10^{-3}	2.0×10^{-5}
LR scheduler	CosineAnealing	CosineAnealing
Warmup ratio	0.03	0.05
Training epochs	1	1
Batch size	256	128
Sequence length	2048	2048
Optimizer	AdamW	AdamW

Table 4: Details of some hyperparameters during training.

RefCOCO [Kazemzadeh et al., 2014], VQAv2 [Goyal et al., 2017], GQA [Hudson and Manning, 2019], Visual7w [Zhu et al., 2016], VSR [Liu et al., 2023a], IconQA [Lu et al., 2021], and OK-VQA [Marino et al., 2019]. These public datasets focus on general image captioning, grounding, question-answering, and conversation. Specifically, ShareGPT4V, textOCR-GPT4V and LAION-GPTV are obtained based on GPT4V [OpenAI, 2023b]. Benefiting from the multi-modal understanding ability of GPT4V, these datasets contain large-scale high-quality image-text data. We also curated 100k high-quality in-house multi-modality SFT data as the deepseek-vl [Lu et al., 2024] according to its taxonomy, to further enhance the quality of multi-modality data.

Table and chart data. The ability to understand tables and charts is critical for open-world multi-modal large language models in many real scenarios. We use ScreenQA [Hsiao et al., 2022], Geo170K [Gao et al., 2023], ScienceQA [Lu et al., 2022], AI2D [Kembhavi et al., 2016b], TQA [Kembhavi et al., 2017], ChartQA [Masry et al., 2022], DVQA [Kafle et al., 2018] and PlotQA [Methani et al., 2020]. We also extract table and chart data from pre-training datasets such as Ureader [Ye et al., 2023] and the Cauldron [Laurençon et al., 2024]. Both datasets are collections of multiple vision-language datasets including table, chart, and document understanding. Note that some duplicate datasets are removed. ScreenQA contains question-answer pairs over mobile app screenshots. Geo170K and ScienceQA are about math and science knowledge, respectively.

OCR data. Optical character recognition (OCR) is an indispensable ability for multi-modal large language models. Models with better OCR capability usually perform better on understanding tables and charts. Several ocr-based question-answering datasets such as TextVQA [Singh et al., 2019], OCRVQA [Mishra et al., 2019], and InfoVQA [Mathew et al., 2022] are used. We also use the Synthdog [Kim et al., 2022] dataset to boost the OCR capability. The dataset contains different languages such as English, Chinese, Japanese, and Korean. In our training process, we only use the English and Chinese data. The English and Chinese data contain over a million samples. We only randomly select 120k samples to avoid the OCR data dominating the training process.

Text-only data. To maintain the language ability in the supervised fine-tuning stage of multi-modal large language models, we include publicly available text-only data in our training data. Specifically, several datasets about math (*e.g.*, MetaMathQA [Yu et al., 2023], MathInstruct [Yue et al., 2023b]), code (Codeit [Raheja et al., 2023]), logical reasoning, conversation (ShareGPT [Zheng et al., 2024]), and Chinese QA (COIG [Bai et al., 2024]) to boost the language ability of specific domains.

Overall, the training data has 3.15 million samples including image-text pairs and text-only data. Table 3 shows the detailed statistics of our training data. Benefiting from the high quality and diversity of the SFT data, the obtained large models not only achieve good performance on benchmarks but also can chat with humans in a user-friendly manner.

4 Experiments

4.1 Implementation Details

We train our model on NeMo [Harper et al.], a scalable training framework based on Megatron-LM[NVIDIA], which supports 3D parallelism and distributed training. When adopting pipeline parallelism, we split the Transformer blocks of the LLM into several stages, and place the vision encoder and vision-language connector in the first stage. When adopting tensor parallelism, we split the Embedding layer and all linear layers within the LLM into different GPUs. Since the vision

Benchmark	Our HawkLlama	LLaVA-Llama3-v1.1 [Contributors, 2023]	LLaVA-Next [Liu et al., 2024a]
MMMU val	**37.8**	36.8	36.9
SEEDBench img	**71.0**	70.1	70.0
MMBench-EN dev	**70.6**	70.4	68.0
MMBench-CN dev	**64.4**	64.2	60.6
CCBench	**33.9**	31.6	24.7
AI2D test	65.6	**70.0**	67.1
ScienceQA test	**76.1**	72.9	70.4
HallusionBench	41.0	**47.7**	35.2
MMStar	43.0	**45.1**	38.1

Table 5: Evaluation results on public vision-language benchmarks.

encoder and vision-language connector are relatively small, we do not split them and keep a copy of them in each tensor parallelism rank. Data parallelism is implemented by PyTorch's [Ansel et al., 2024] distributed package. With the techniques above, we are able to train a larger model on a larger cluster in theory. In this work, we use a pipeline parallelism size of 1, a tensor parallelism size of 4 and a data parallism size of 4. We further adopt activation checkpointing to save memory. During the vision-language alignment stage, a learning rate of 1.0×10^{-3} and a batch size of 256 are employed. For the supervised fine-tuning phase, the learning rate is adjusted to 2.0×10^{-5} and a batch size of 128. We use AdamW [Loshchilov and Hutter, 2017] optimizer and train 1 epoch for both stages. We train our model on 2 nodes, each with 8 NVIDIA H800 GPUs. Some more hyperparameters we use during training are shown in detail in Table 4.

4.2 Evaluation

We conducted evaluations of our model across a variety of public benchmarks to verify its capabilities in vision-language tasks, including MMMU [Yue et al., 2023a], SEEDBench [Li et al., 2023a], MMBench [Liu et al., 2023c], CCBench [Liu et al., 2023c], AI2D [Kembhavi et al., 2016a], ScienceQA [Lu et al., 2022], HallusionBench [Guan et al., 2023], MMStar [Chen et al., 2024] and MME [Fu et al., 2023]. These benchmarks encompass a variety of image-text conversation types, serving as metrics for evaluating the MLLM's capabilities in image-text comprehension and generation.

We present the performance of our model alongside several leading MLLMs in Table 5. HawkLlama surpasses models of comparable size on benchmarks, such as 37.8 on MMMU, 71.0 on SEEDBench, 64.4 on MMBench-EN, 64.4 on MMBench-CN, 33.9 on CCBench, and 76.1 on ScienceQA, demonstrating its superior image comprehension and generation capabilities. However, HawkLlama lags behind LLaVA-Next and LLaVA-Llama3-v1.1 on benchmarks like MMStar, AI2D and HallusionBench. This implies that there is still room for improvement in our model regarding diagram comprehension and the reduction of hallucinations. It's worth noting that LLaVA-Llama3-v1.1 uses a larger pretraining dataset than ours. Our forthcoming objective is to develop a more substantial pretraining dataset to enhance our model's proficiency in OCR, table and chart understanding, and world knowledge assimilation.

4.3 Qualitative Results

We provide some qualitative results in Figure 2, Figure 3 and Figure 4. These examples demonstrate our model's capabilities in image comprehension, code recognition, understanding and generation, meme interpretation, mathematical problem solving, and multilingual dialogue.

5 Discussion

In this work we introduce HawkLlama, an open-source multi-modal large language model based on Llama-3 for real-world vision and language understanding. Our model still faces challenges in chart comprehension and exhibits a certain degree of hallucination issues. Our future work will aim to mitigate these problems by increasing the input image resolution, enhancing the visual perception module, and collecting a larger pretraining dataset.

Acknowledgements: This work was supported by National Key R&D Program of China (No. 2022ZD0118700).

Figure 2: More Visualization Results.

What coin is that?

The image shows a coin with an embossed design on its surface. The central feature of the coin is a depiction of a temple complex with multiple spires and domes, which suggests it might be related to a religious or cultural significance.
The text around the edge of the coin appears to be in Thai script, which indicates that the coin is likely from Thailand. The numbers "10" are prominently displayed at the bottom of the coin, suggesting that this coin has a value of 10 units of currency, possibly baht (THB), which is the official currency of Thailand.
The coloration of the coin is silver-toned, and there's a visible sheen on the surface, which implies that the material used for the coin could be metal, such as silver or a silver alloy. The overall condition of the coin seems to be well-preserved, with no visible signs of wear or damage.
Based on these observations, the coin can be identified as a Thai baht coin featuring a temple design. The specific year or series is not discernible from the image provided.

Turn the python function in the image into c++ code

```python
python                                                    复制代码
def insertion_sort(arr):
    # 遍历数组中的每一个元素，从第二个元素开始
    for i in range(1, arr.length):
        key = arr[i]
        j = i - 1
        # 将当前元素与已排序部分进行比较并插入到正确的位置
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key
    return arr

# 示例用法
arr = [12, 11, 13, 5, 6]
sorted_arr = insertion_sort(arr)
print("排序后的数组:", sorted_arr)
```

The image shows a section of C++ code with comments and variable names. The code is written in a programming style that suggests it's part of a larger program or algorithm. However, without the full context or the rest of the code, it's challenging to provide an accurate conversion to C++.
From what we can see, there are several lines of code with comments explaining their purpose:
def insertion_sort(arr): - This line defines a function named "insertion_sort" which takes an array as input.
for i n range(0, len(arr), -1): - This line appears to be a loop that iterates over the elements of the array in reverse order.
key = arr[i] - This line assigns the value at index i from the array to a variable called key.
j = 0 - This line initializes a counter j to zero.
while j < len(arr) and key >= arr[j]: - This line enters a loop that continues until key is greater than or equal to the current element arr[j].
arr[j + 1] = key - If the condition in the while loop is not met, this line moves the element at index j + 1 to the position where key should be inserted.
return arr - This line returns the sorted array.
Given these observations, here is how you might translate this into C++ code:

```cpp
#include <iostream>
#include <vector>
// Function to sort an array using Insertion Sort
std::vector<int> insertion_sort(std::vector<int>& arr) {
   for (int i = 0; i < arr.size(); --i) {
      int key = arr[i];
      int j = 0;
      while (j < arr.size() && key >= arr[j]) {
         arr[j + 1] = arr[j];
         j++;
      }
      arr[j] = key;
   }
   return arr;
}
int main() {
   std::vector<int> arr = {64, 34, 25, 12, 22, 11, 90};
   auto sorted_arr = insertion_sort(arr);
   for (int i : sorted_arr) {
      std::cout << i << '';
   }
   return 0;
}
```

Please note that the above code assumes that the input array is passed by reference (&arr) to the insertion_sort function, as indicated by the & symbol before arr. Also, the main function is included to demonstrate how to use the insertion_sort function. The output of this code will be the sorted version of the input array.
Remember, the actual implementation may vary based on the specific requirements of your project, including the data type of the elements in the array, whether the array is required to be modified in-place or returned, and any additional constraints or conditions that apply to the sorting process.

If there are factual errors in the questions, point it out; if not, proceed answering the question. What's happening in the desert?

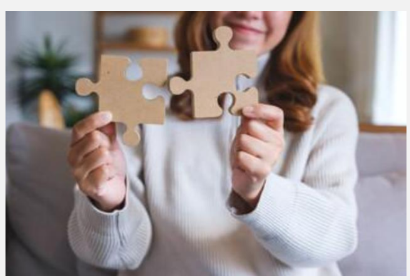
The image provided does not depict a desert scene but rather shows a beachfront at dusk or dawn with a view of a city skyline across the water. The sky is cloudy, and the ocean is visible with reflections from the city lights. There are no sandy dunes, cacti, or any typical desert elements present in this image. Therefore, I cannot answer what's happening in the desert as it is not depicted here.

Figure 3: More Visualization Results.

What animal is running?

There is no animal running in this image. The elephants are standing still in their enclosure.

Can the two puzzles be combined together?

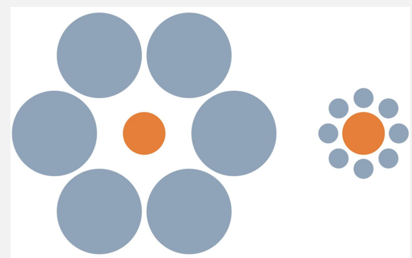


The image shows two puzzle pieces being held by a person's hands. The puzzle piece on the left is oriented vertically, while the one on the right is oriented horizontally. Both pieces have a similar shape and size, suggesting they could potentially fit together.

To determine if these two puzzle pieces can be combined together, we need to look at their shapes and sizes relative to each other. The vertical piece has a rectangular shape with a square base, and the horizontal piece also has a square base but is oriented differently. If we were to place them together, the top edge of the vertical piece would align with the bottom edge of the horizontal piece, and vice versa. This suggests that they could indeed fit together as part of a larger puzzle.

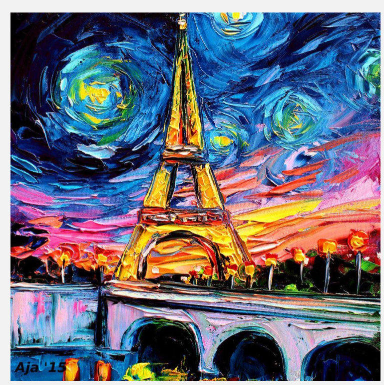
However, without seeing how the puzzle pieces fit together or having additional context about the design of the puzzle, it is not possible to definitively confirm that they will form a complete puzzle when combined. The visual clues in the image, such as the similar shape and size of the pieces and their orientation, support the idea that they could be part of a puzzle designed to be assembled together.

Is the right orange circle appears to be the same size as the left orange circle?



Yes, the right orange circle does appear to be the same size as the left orange circle. Both circles are of equal size in terms of their diameter, which is evident from the uniform thickness of the lines forming the circles. The diameter of both circles is consistent, indicating that they have the same radius.
The visual clue supporting this statement is the proportionate relationship between the two circles. They are placed on either side of a central point, and their sizes relative to each other are identical. There are no visual indications of one being larger or smaller than the other; they are both equally large within the context of the image provided.

What is in the picture?



The image depicts an artistic representation of the Eiffel Tower set against a vibrant sunset sky. The tower is rendered in a blend of warm colors, giving it a golden hue. The painting style appears to be abstract, with brushstrokes creating a textured effect. There's also a bridge visible in the foreground, leading towards the tower.

Figure 4: More Visualization Results.

References

Ibrahim M Alabdulmohsin, Xiaohua Zhai, Alexander Kolesnikov, and Lucas Beyer. Getting vit in shape: Scaling laws for compute-optimal model design. *Advances in Neural Information Processing Systems*, 36, 2024.

Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang, William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024. doi: 10.1145/3620665.3640366. URL https://pytorch.org/assets/pytorch2-2.pdf.

Anthropic. Introducing Claude, 2023. URL https://www.anthropic.com/index/introducing-claude.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023a.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023b.

Yuelin Bai, Xinrun Du, Yiming Liang, Yonggang Jin, Ziqiang Liu, Junting Zhou, Tianyu Zheng, Xincheng Zhang, Nuo Ma, Zekun Wang, et al. Coig-cqia: Quality is all you need for chinese instruction fine-tuning. *arXiv preprint arXiv:2403.18058*, 2024.

Jimmy Carter. Textocr-gpt4v. https://huggingface.co/datasets/jimmycarter/textocr-gpt4v, 2024.

Lin Chen, Jisong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. Sharegpt4v: Improving large multi-modal models with better captions. *arXiv preprint arXiv:2311.12793*, 2023.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Jiaqi Wang, Yu Qiao, Dahua Lin, et al. Are we on the right way for evaluating large vision-language models? *arXiv preprint arXiv:2403.20330*, 2024.

Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension. pages 845–855, 2018.

XTuner Contributors. Xtuner: A toolkit for efficiently fine-tuning llm. https://github.com/InternLM/xtuner, 2023.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning, 2023.

DeepSeek-AI. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024. URL https://github.com/deepseek-ai/DeepSeek-LLM.

Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Bin Wang, Linke Ouyang, Xilin Wei, Songyang Zhang, Haodong Duan, Maosong Cao, et al. Internlm-xcomposer2: Mastering free-form text-image composition and comprehension in vision-language large model. *arXiv preprint arXiv:2401.16420*, 2024.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335, 2022.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, and Rongrong Ji. Mme: A comprehensive evaluation benchmark for multimodal large language models. *arXiv preprint arXiv:2306.13394*, 2023.

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 *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 6904–6913, 2017.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, Dinesh Manocha, and Tianyi Zhou. Hallusionbench: An advanced diagnostic suite for entangled language hallucination & visual illusion in large vision-language models, 2023.

Eric Harper, Somshubra Majumdar, Oleksii Kuchaiev, Li Jason, Yang Zhang, Evelina Bakhturina, Vahid Noroozi, Sandeep Subramanian, Koluguri Nithin, Huang Jocelyn, Fei Jia, Jagadeesh Balam, Xuesong Yang, Micha Livne, Yi Dong, Sean Naren, and Boris Ginsburg. NeMo: a toolkit for Conversational AI and Large Language Models. URL https://github.com/NVIDIA/NeMo.

Yu-Chung Hsiao, Fedir Zubach, Maria Wang, et al. Screenqa: Large-scale question-answer pairs over mobile app screenshots. *arXiv preprint arXiv:2209.08199*, 2022.

Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 6700–6709, 2019.

Kushal Kafle, Brian Price, Scott Cohen, and Christopher Kanan. Dvqa: Understanding data visualizations via question answering. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 5648–5656, 2018.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798, 2014.

Aniruddha Kembhavi, Michael Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. *arXiv: Comp. Res. Repository*, abs/1603.07396, 2016a. URL https://api.semanticscholar.org/CorpusID:2682274.

Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *Proc. Eur. Conf. Comp. Vis.*, pages 235–251, 2016b.

Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 4999–5007, 2017.

Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. Ocr-free document understanding transformer. In *European Conference on Computer Vision (ECCV)*, 2022.

LAION. Gpt-4v dataset. https://huggingface.co/datasets/laion/gpt4v-dataset, 2023.

Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. What matters when building vision-language models?, 2024.

Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. Seed-bench: Benchmarking multimodal llms with generative comprehension. *arXiv preprint arXiv:2307.16125*, 2023a.

Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for" mind" exploration of large language model society. *Advances in Neural Information Processing Systems*, 36:51991–52008, 2023b.

Fangyu Liu, Guy Emerson, and Nigel Collier. Visual spatial reasoning. *Transactions of the Association for Computational Linguistics*, 11:635–651, 2023a.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023b.

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 2024a. 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, 2024b.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint arXiv:2007.08124*, 2020.

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*, 2023c.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Haoyu Lu, Wen Liu, Bo Zhang, Bingxuan Wang, Kai Dong, Bo Liu, Jingxiang Sun, Tongzheng Ren, Zhuoshu Li, Hao Yang, Yaofeng Sun, Chengqi Deng, Hanwei Xu, Zhenda Xie, and Chong Ruan. Deepseek-vl: Towards real-world vision-language understanding, 2024.

Pan Lu, Liang Qiu, Jiaqi Chen, Tony Xia, Yizhou Zhao, Wei Zhang, Zhou Yu, Xiaodan Liang, and Song-Chun Zhu. Iconqa: A new benchmark for abstract diagram understanding and visual language reasoning. *arXiv preprint arXiv:2110.13214*, 2021.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521, 2022.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 3195–3204, 2019.

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. pages 2263–2279, 2022.

Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and CV Jawahar. Infographicvqa. In *Proc. Workshop of Applications of Comp. Vis.*, pages 1697–1706, 2022.

Meta. Introducing meta llama 3: The most capable openly available llm to date, 2024. URL https://ai.meta.com/blog/meta-llama-3/.

Nitesh Methani, Pritha Ganguly, Mitesh M Khapra, and Pratyush Kumar. Plotqa: Reasoning over scientific plots. In *Proc. Workshop of Applications of Comp. Vis.*, pages 1527–1536, 2020.

Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. Ocr-vqa: Visual question answering by reading text in images. In *Proc. International Conference on Document Analysis and Recognition*, pages 947–952, 2019.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-math: Unlocking the potential of slms in grade school math. *arXiv preprint arXiv:2402.14830*, 2024.

NVIDIA. Megatron-lm. URL https://github.com/NVIDIA/Megatron-LM.

OpenAI. Chatgpt: Optimizing language models for dialogue. 2022. URL https://openai.com/blog/chatgpt.

OpenAI. GPT-4 technical report. *arXiv*, 2023a.

OpenAI. Gpt-4v(ision) system card. 2023b.

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*, pages 8748–8763. PMLR, 2021.

Vipul Raheja, Dhruv Kumar, Ryan Koo, and Dongyeop Kang. Coedit: Text editing by task-specific instruction tuning. *arXiv preprint arXiv:2305.09857*, 2023.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 8317–8326, 2019.

Quan Sun, Yufeng Cui, Xiaosong Zhang, Fan Zhang, Qiying Yu, Zhengxiong Luo, Yueze Wang, Yongming Rao, Jingjing Liu, Tiejun Huang, and Xinlong Wang. Generative multimodal models are in-context learners. 2023a.

Quan Sun, Qiying Yu, Yufeng Cui, Fan Zhang, Xiaosong Zhang, Yueze Wang, Hongcheng Gao, Jingjing Liu, Tiejun Huang, and Xinlong Wang. Generative pretraining in multimodality. *arXiv preprint arXiv:2307.05222*, 2023b.

Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models, 2024.

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.

Reka Team, Aitor Ormazabal, Che Zheng, Cyprien de Masson d'Autume, Dani Yogatama, Deyu Fu, Donovan Ong, Eric Chen, Eugenie Lamprecht, Hai Pham, Isaac Ong, Kaloyan Aleksiev, Lei Li, Matthew Henderson, Max Bain, Mikel Artetxe, Nishant Relan, Piotr Padlewski, Qi Liu, Ren Chen, Samuel Phua, Yazheng Yang, Yi Tay, Yuqi Wang, Zhongkai Zhu, and Zhihui Xie. Reka core, flash, and edge: A series of powerful multimodal language models, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023b. doi: 10.48550/arXiv.2307.09288. URL https://doi.org/10.48550/arXiv.2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, et al. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*, 2023.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Ruyi Xu, Yuan Yao, Zonghao Guo, Junbo Cui, Zanlin Ni, Chunjiang Ge, Tat-Seng Chua, Zhiyuan Liu, and Gao Huang. LLaVA-UHD: an lmm perceiving any aspect ratio and high-resolution images. *arXiv preprint arXiv:2403.11703*, 2024.

Jiabo Ye, Anwen Hu, Haiyang Xu, Qinghao Ye, Ming Yan, Guohai Xu, Chenliang Li, Junfeng Tian, Qi Qian, Ji Zhang, et al. Ureader: Universal ocr-free visually-situated language understanding with multimodal large language model. *arXiv preprint arXiv:2310.05126*, 2023.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.

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. *arXiv preprint arXiv:2311.16502*, 2023a.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023b.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. *arXiv preprint arXiv:2303.15343*, 2023.

Pan Zhang, Xiaoyi Dong Bin Wang, Yuhang Cao, Chao Xu, Linke Ouyang, Zhiyuan Zhao, Shuangrui Ding, Songyang Zhang, Haodong Duan, Hang Yan, et al. Internlm-xcomposer: A vision-language large model for advanced text-image comprehension and composition. *arXiv preprint arXiv:2309.15112*, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Proc. Advances in Neural Inf. Process. Syst.*, 36, 2024.

Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. Visual7w: Grounded question answering in images. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 4995–5004, 2016.