Yi: Open Foundation Models by 01.AI

01.AI

Code: https://github.com/01-ai/Yi
Model: https://huggingface.co/01-ai

Abstract

We introduce the Yi model family, a series of language and multimodal models that demonstrate strong multi-dimensional capabilities. The Yi model family is based on 6B and 34B pretrained language models, then we extend them to chat models, 200K long context models, depth-upscaled models, and vision-language models. Our base models achieve strong performance on a wide range of benchmarks like MMLU, and our finetuned chat models deliver strong human preference rate on major evaluation platforms like AlpacaEval and Chatbot Arena. Building upon our scalable super-computing infrastructure and the classical transformer architecture, we attribute the performance of Yi models primarily to its data quality resulting from our data-engineering efforts. For pretraining, we construct 3.1 trillion tokens of English and Chinese corpora using a cascaded data deduplication and quality filtering pipeline. For finetuning, we polish a small scale (less than 10K) instruction dataset over multiple iterations such that every single instance has been verified directly by our machine learning engineers. For vision-language, we combine the chat language model with a vision transformer encoder and train the model to align visual representations to the semantic space of the language model. We further extend the context length to 200K through lightweight continual pretraining and demonstrate strong needle-in-a-haystack retrieval performance. We show that extending the depth of the pretrained checkpoint through continual pretraining further improves performance. We believe that given our current results, continuing to scale up model parameters using thoroughly optimized data will lead to even stronger frontier models.

Contents

1 Introduction

Recent breakthroughs in large language models have revolutionized the whole field of artificial intelligence and potentially radiate across the entire human society. Our vision for large language models is to make them the next generation computational platform and empower the whole community with significantly amplified intelligence. As a step towards this mission, we present the Yi model series, 6B and 34B language models pretrained from scratch on 3.1T highly-engineered large amount of data, and finetuned on a small but meticulously polished alignment data. Due to the data quality resulting from our substantial engineering efforts, which we will detail in the upcoming sections, Yi achieves near GPT-3.5 benchmark scores and human preferences.

In designing the Yi model series, we are mostly concerned on the following dimensions regarding *model scale*, *data scale*, *and data quality*: (1). when choosing model scale, the desiderata is to have small enough model that is feasible for inference on consumer-grade hardware like the RTX 4090 where the bounding factor is its limited 24G memory, yet still large enough with complex reasoning and emergent abilities. This is why we found 34B gives a nice performance-cost balance; (2). since 34B is smaller than the conventional 70B used by Chinchilla [30] and LLaMA [77], we increase the pretrain data scale to 3.1T tokens to compensate for the decreased compute flops. This makes the model-data scale combination fall into the post Chinchilla optimal regime [64], i.e., we overtrain the model on more tokens (3T) than the compute optimal (around 1T). The benefit is from the inference side, as we achieve stronger performance with reduced serving cost: after int4 [81] quantization, one can serve the 34B chat model on 24G GPU memory with almost no performance drop; (3). our data engineering principle is to promote quality over quantity for both pretraining and finetuning. The pretraining data quality is guaranteed by a sophisticated data cleaning pipeline with cascaded filtering methods and intentionally increased deduplication strength; (4). for finetuning data we heavily emphasize quality by handcrafting less than 10K instructions over multiple iterations based on user feedback. This approach significantly deviates from the quantity-scaling styled instruction tuning works like FLAN [9] and UltraChat [19], but aligns more with the handcrafting styled works like LIMA [94].

Our pretraining data cleaning system features a sophisticated filtering pipeline based on language, heuristic textual features, perplexity, semantics, topic, and safety, as well as a cascaded deduplication process based on paragraph, MinHash, and exact matching. This thorough pipeline leads to a much higher removal ratio than existing pipelines like CCNet [80], RefinedWeb [56] and RedPajama [13], which we believe is key to the success of data engineering. The underlying principle is although pretraining requires data scaling, one would like to make sure the data used are of high quality, rather than training the model on large raw data, i.e., we prefer 3T tokens over sophasticated engineering over 10T tokens without extensive filtering. Regarding the model architecture, we use standard implementation of the Transformer architecture with Grouped-Query Attention (GQA) [1], SwiGLU [68] activation, and RoPE with an adjusted base frequency (RoPE ABF) [82]. This design choice is the standard approach rooted from the Transformer original paper [78], later modified by GPT-3 and Chinchilla [30], then followed by LLaMA [77], Baichuan [84], Qwen [3] and many related works.

To approach GPT-3.5-matching human preferences, our finetuning dataset is curated from carefully selected multi-turn instruction-response pairs, annotated directly by our team of machine learning engineers then polished over multiple iterations of user feedback. As mentioned above, the size of our finetuning dataset is less than 10K, but improved over and over again across the model development timeline. Benefiting from the dataset's manageable size, we employed an extensive grid search to identify the optimal data composition, promote diversity, and discover effective hyperparameters. After 8-bit and 4-bit quantization, the final chat model can be deployed on consumer-grade GPUs nearly without performance degradation compared to the bf16 format.

We further extend the Yi model capability from three dimensions: context scaling, vision-language adaptation, and depth-upscaling. To achive 200K context length, we continue pretrain the model on about 5B length-upsampled data, similar to the concurrent work in Fu et al. [22]. To adapt the model to vision-language tasks, we integrate a vision encoder and develop a multi-stage training method, following and improving the practice of Liu et al. [47]. We also study the effectiveness of depth-upscailng [38], i.e., making the model deeper by continual pretraining, and confirming its effectiveness to further improve model performance.

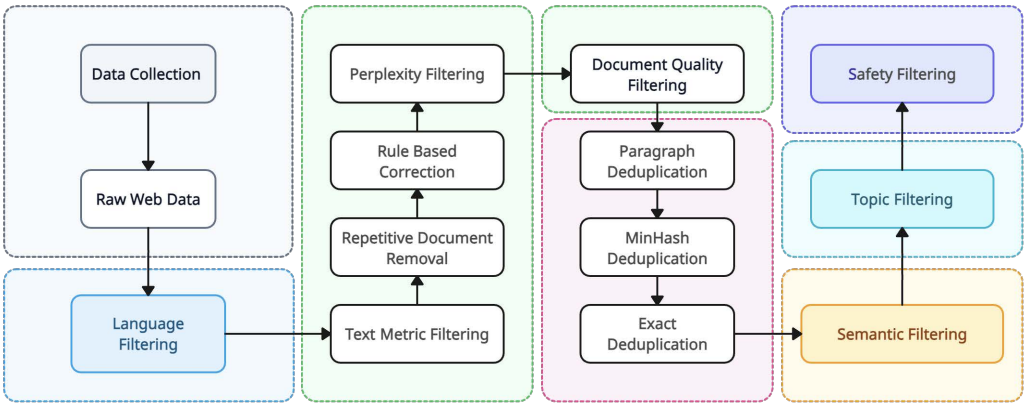

Figure 1: Yi's pretraining data cleaning pipeline.

Our infrastructure provides strong support for the full-stack development of the Yi model series, from pretraining to finetuning to serving. To support pretraining, we develop cross-cloud elastic task scheduling, automatic failure recovery, and topology-aware resource allocation which collectively enable us to run tasks according to the real-time available GPU nodes cross clusters with limited switching overhead. To support finetuning, we build a hierarchical scheduling framework supporting different distributed backends for different models (e.g., Megatron [70] for the policy model and DeepSpeed [60] for the reward model). For efficient inference, we use 4-bit model and 8-bit KV cache quantization, combining with PagedAttention [41] and Dynamic Batching.

Extensive experiments demonstrate that Yi-34B can match GPT-3.5 in both performance and efficiency. On most standard benchmarks like MMLU [27] (for the base model) and LMSys ELO Rating [93] (for the chat model), Yi-34B generally achieves scores on par with GPT-3.5. After model parameter and KV cache quantization, the inference cost is also controlled such that a wide range of the community can deploy the model on cost effective devices. We further report a detailed performance comparison between Yi and major LLMs on commonsense reasoning, college exams, math, coding, reading comprehension, and human preference win-rate on multiple evaluation benchmarks.

Since its release, the Yi model series has benefited the community from the following perspectives: (1). it provides GPT-3.5-matching quality yet cost-effective models to researchers, and enables developers to build AI-native applications like language model based agents; (2). it empowers end users with locally runnable chatbots, which consequently helps protecting user data privacy; (3). it sheds light on the direction on further data and model scaling to achieve even stronger frontier models. for both research and commercial use.

2 Pretraining

Our approach to pretraining is to train a standard dense transformer architecture on a heavily engineered large pretraining corpora, where our underlying assumption is that when trained on extensive data of high-enough quality, a standard architecture can exhibit advanced capability. This is to say, we may not need much architectural modification, although we have indeed conducted extensive preliminary architectural experiments. In the following subsections, we first detail our data engineering pipeline, then briefly discuss the model architecture.

2.1 Data Processing

The Yi data mixture is shown in Fig. 2. To produce a high-quality bilingual pretraining data, we meticulously designed a cascaded data-processing pipeline, as illustrated in Fig 1. This pipeline features a series of data-cleaning strategies targeting quality and diversity. We start with web documents from Common Crawl, use the CCNet pipeline [79] for language identification and perplexity scoring. Then we use a combination of filtering and deduplication process, as detailed below.

Heuristic Rule Filters This part of filter aims for removing text of low quality. We filter out text based on: (1). URL, domain, word blocklists and garbled text filters; (2). document length, the ratio of special symbols, and the ratio of short, consecutive, or incomplete lines; (3). repeated words, n-grams, or paragraphs [58]; The filtering thresholds are based on a statistical analysis of large document samples, as described in Nguyen et al. [52]. Furthermore, we identify and anonymize Personal Identifiable Information (PII), such as email addresses and phone numbers.

Learned Filters We use learned filters to address nuanced cases that exceed the capabilities of standard heuristic rules. Notably, the Chinese content extracted from Common Crawl present unique challenges, particularly with a higher ratio of inappropriate content like pornography and gambling. Traditional heuristic-rule-based filters struggle to effectively identify and eliminate all harmful content. To enhance our filtering process, we have integrated a suite of learned scorers for filtering, namely the perplexity scorer, quality scorer, safety scorer, and document coherence scorer: (1). the *Perplexity Scorer*, utilizing the KenLM library as per CCNet [80], evaluates a vast array of web documents, discarding those with perplexity scores largely above average; (2). the *Quality Scorer* is a classifier trained to recognize and favor pages similar to Wikipedia in quality and assign scores accordingly. Documents that fail to meet the quality standard are subsequently removed; (3). the *Document Coherence Scorer* identifies low-quality web documents that consist of disparate sentences or paragraphs, thus being incoherence. Such documents are either segmented for further analysis or removed entirely. (4). the *Safety Scorer* identifies and removes web documents containing toxic content, such as violence, pornography, and political propaganda.

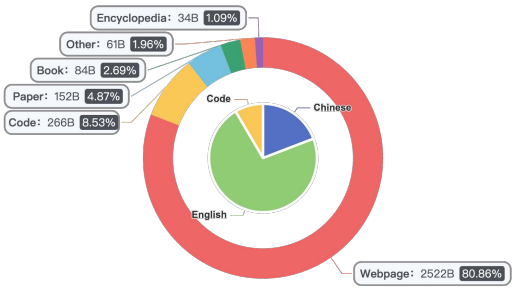

Figure 2: Yi's pre-training data mixture. Overall our data consist of 3.1T high-quality tokens in Both English and Chinese, and come from various sources. Our major differences from existing known mixtures like LLaMA [76] and Falcon [56] are that we are bilingual, and of higher quality due to our more rigorous cleaning pipeline.

Cluster-based Filters We further use unsupervised semantic clustering to group web documents. This clustering process enables efficient identification and analysis of documents sharing similar semantic features. The clustered data are subsequently annotated with quality labels, providing essential references for the optimization of Yi's data mixture strategy. Documents identified as low-quality through automatic and manual verification are excluded from the dataset.

Deduplication After filtering, we implement a comprehensive deduplication pipeline following the procedure in Penedo et al. (2023) [56]. This pipeline integrates document-level MinHash deduplication and sub-document exact-match deduplication, effectively identifying and removing duplicate content within and across documents. We further categorize web documents into specific themes using a topic model predicting labels like as news, ads, and knowledge-based content. In the final pretraining dataset, we down-sample less helpful content, mostly advertisements, to ensure information density. The final composition of Yi's pretraining data is shown in Fig. 2.

2.2 Tokenization

We use byte-pair encoding (BPE) [69] implemented in the SentencePiece framework [40], to tokenize the pretraining data. The vocabulary size of Yi is set to 64,000 to balance computational efficiency and word comprehension. Specifically, we split numbers into individual digits to facilitate a better understanding of numeric data. We allow rare characters to fall back to the unicode-byte encoding to ensure fault tolerance. We employ the identity tokenizer to avoid transferring all punctuations to the half-width format. LLMs prioritizing English usually utilize dummy prefix (whitespace at the beginning of text) in their tokenizers to generalize the same words at different positions of sentences. We do not use this approach because the assumption does not always hold even in the English context, especially for sentences that begin with quotation marks, also it does not show positive effect in Chinese context.

Models	Hidden Size	Q-heads	KV-heads	Layers	Pretrain Seq. Len	Max LR
6B	4096	32	4	32	4096	3×10^{-4}
34B	7168	56	8	60	4096	1.5×10^{-4}

Table 1: Model configs of Yi-6B and Yi-34B. LR stands for learning rate.

2.3 Model Architecture

Yi uses a modified version of the classical decoder-only Transformer architecture [78] where the code is based on LLaMA's [77] implementation. The main parameter setting is summarized in Table 1. The modifications from LLaMA to Yi are further summarized below:

Attention Mechanism LLaMA 2 uses Grouped-Query Attention(GQA) [1] only on its largest 70B model, and its 7B and 13B uses full attention. We incorporate GQA in both Yi-6B and Yi-34B. GQA splits query-heads into G groups, sharing a single key and value head within each group of query [1]. This approach offers substantial reductions of training and inference costs, compared to the original Multi-Head Attention (MHA) [16, 57, 67]. We do not observe performance degradation after applying GQA to our 6B smaller model.

Activation Function We use SwiGLU [68] as Yi's post-attention layer, reducing its activation size from $4h$ to $8/3h$ (h denotes hidden size) to be consistent with the normal post-attention layer. This adjustment also compensates for the reduction in parameter resulted from GQA, making the overall parameter count comparible of existing 7B and 34B models.

Positional Embedding and Long Context We use Rotary Position Embedding (RoPE) [73] following the standard implementation. We adjust the base frequency (RoPE ABF), introduced in Xiong et al. [82], to support long context windows up to 200K where the base model itself is trained on 4K context length. To adapt the base model to longer context, we continue pretrain the model on 10B tokens from our pretraining data mixture with slightly upsampled long sequences, mostly from book. We observe that only 1-2B tokens is enough for the model to converge to low loss on 4K-200K length, and a lightweight finetuning further induces near-perfect long-context retrieval performance. Based on this observation, we tend to view that the capability of modeling longer dependency than the pretrained length (4K) is a intrinsic capability (rather than an being injected by post-train). This is to say, the base model already has the capability to model longer than 4K dependency even the model is trained shorter, and the post-train / finetuning procedure simply release this capability.

3 Finetuning

Our finetuning method significantly emphasizes data quality over quantity. Our approach does *not* follow existing data-intensive approaches like FLAN [9] and UltraChat [19], which scales the SFT data to millions of entries but each of the entries may not been examined carefully because the scale is too large. In contrast, our method aligns with the LIMA [94] and DEITA [48] approach, which focus on data selection rather than scaling. With the scale being less than 10K, we are able to examine and optimize *every single data point*. Below we discuss our data construction and training details.

3.1 Data Preprocessing

Quality is All You Need Our finetuning dataset consists of less than 10K multi-turn instruction-response dialog pairs, with each and every one of the entry constructed and polished over multiple iterations and from user feedback. We take this approach because in our preliminary experiments, we observe that compared to the open-source data of several hundred thousand entries, the results from a smaller, manually annotated dataset are superior. These observations align with those reported in Gemini Team et al. [23], Touvron et al. [77], Zhou et al. [94].

We use the following techniques to improve prompt distribution selection, response formatting, and chain-of-thought formatting: (1). for prompt distribution selection, drawing inspiration from WizardLM[83], we develope compound instructions and progressively evolved them to increase their complexity. This approach has significantly reduced the size of SFT data in our experiments; (2). for response formatting, we generally use a default style extended from LIMA[94]. Overall, the responses are structured in an introduction-body-conclusion format where the body is usually a list of bullet point; (3). for CoT data formatting, we have use a "Step-Back" pattern, inspired by Zheng et al. [92], by performing abstraction to formulate higher-level solutions before delving into reasoning about the original, more concrete questions.

We spend extra efforts on reducing hallucination and repetition: (1). to reduce hallucinations, we examine and ensure that the knowledge in the responses is not contained within the model, and eliminate responses that might lead to memorization; (2). to reduce repetition, we rewrite the repetitive turns of the responses that usually exist but may be overlooked in the finetuning data.

Diversity and Mixture To ensure the coverage of different capabilities, we have included a wide spectrum of open-source prompt, encompassing areas such as question answering, creative writing, dialogue, reasoning, mathematics, coding, safety, bilingual capabilities, and others.

To obtain a fine-grained control of different directions of capabilities, inspired by InsTag[49], we develop a instruction tagging system. By designing a diversity-focused sampling algorithm, we carefully balanced the distribution of instructions across various tags. This approach ensures a diverse finetuning dataset, aiming to achieve enhanced cross-task robustness.

To achieve the optimal data ratio for balancing different directions of the capability, we use an approximate grid search to determine our data mixture. Motivated by Dong et al. [20], this process involved experimenting with {1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64} proportions for each ability. The search process was guided by validation results and our in-house human evaluation sets.

ChatML Format Beyond the focus on data quality and diversity, our observations revealed that the format of the data substantially influences the model's ultimate performance. To this end, we implemented the ChatML-style format [53]. This structured approach empowers the model to differentiate among various information types, such as system configurations, user inputs, and assistant responses.

3.2 Training Method

We use next-word prediction loss for finetuning, and only compute loss on the responses, but not system and user instructions. We use AdamW optimizer with β_1 set to 0.9, β_2 set to 0.999, and ϵ set to 10^{-8}. We use a sequence length of 4096, alongside a batch size of 64. We set training step to 300 with a constant 1×10^{-5} learning rate, a weight decay of 0.1, gradient clipping with a maximum threshold of 1.0, and NEFTune [34] with a noise scale of 45 for Yi-34B-Chat and 5 for Yi-6B-Chat.

4 Infrastructure

We build the infrastructure supporting the full-stack data processing, pretraining, finetuning, and serving. Our infrastructure feasures: (1). automated managing and monitoring the computing resource; (2). improved the training speed from optimized parallel strategies, kernel efficiency, and long-context support; (3). unified finetuning framework supporting heterogeneous distributed training backend, such as simultaneously using Megatron and DeepSpeed for multiple models in Direct Preference Optimization (DPO) [59]; (4). reducing the deployment cost by various LLM serving accelerations such as quantization, continuous batching, and paged attention. Below we explain these techniques one by one.

Computing Resources Management To efficient schedule large-scale language model development, particularly pretraining, which may take months on thousands of GPUs , we build a highly efficient multi-cloud task scheduling algorithm to manage pre-training, SFT, and RLHF tasks of different priorities. We also build a high-performance in-house training framework that allows us to automatically elastic scale the pre-train jobs to different node sizes based on the GPU availability. More importantly, all the training-related hyper-parameters will be scaled at the same time seamlessly.

	Size	MMLU	BBH	C-Eval	CMMLU	Gaokao	CR	RC	Code	Math
GPT-4	-	**83.0**	**86.7**	69.9	71.0	72.3	**89.3**	-	**65.3**	**66.1**
GPT-3.5	-	69.1	70.1	52.5	55.5	51.1	83.1	-	54.8	35.6
Qwen	14B	66.7	53.4	72.1	71.0	62.5	74.2	72.5	40.6	43.1
Llama2	34B	62.6	44.1	-	-	-	71.1	68.9	27.8	24.2
	70B	69.7	64.9	50.1	53.3	23.3	72.7	72.3	38.4	35.2
Baichuan-2	13B	55.0	49.0	59.0	61.97	45.6	66.3	62.4	23.4	16.1
InternLM	20B	62.1	52.5	58.8	59.0	45.5	78.3	-	34.8	30.26
Skywork	13B	62.1	41.7	60.6	61.8	68.1	72.4	61.4	64.9	18.1
Falcon	180B	70.4	54.0	57.8	58.0	59.0	74.4	-	-	-
Yi	6B	63.2	42.8	72.0	75.5	72.2	72.2	68.7	21.1	18.6
	34B	76.3	54.3	**81.4**	**83.7**	**82.8**	80.7	**76.5**	32.1	40.8

Table 2: Overall performance on grouped academic benchmarks compared to open-source base models. **CR** stands for Commonsense Reasoning. **RC** stands for Reading Comprehension.

During the large language model training stage, a wide range of failures regularly occur, ranging from GPU crashes to communication fabric errors to loss spikes. We use the following strategies to address these reliability challenges: (1) we apply automated inspection, prediction, and labeling of nodes for different kind of software/hardware error categories. Nodes marked as tainted will be temporarily removed from the resource pool until the errors got cleared. (2) we implement a task queuing system with pre-checks and the capability for fast, automatic recovery in the event of failures during training tasks. (3) we develop of a user-friendly multi-task submission and management console, enabling developers to seamlessly manage and track their training tasks and hyper-parameters.

Performance and Cost Efficiency *Memory* and *communication* restrictions are the two major technical challenges of large scale model training requiring integrated solutions beyond adding more GPUs. We use and improve upon the following techniques to tackle the memory and communication restrictions: (1) ZeRO-1 [60] to remove the memory consumption by partitioning optimizer states cross data-parallel processes; (2) tensor parallel combined with pipeline parallel [70] within each compute node to avoid inter-node communication bottleneck, and the 3D parallel strategy is well designed and optimized to avoid using activation checkpointing and minimize the pipeline bubbles; (3) kernel fusion techniques like flash attention[15][14] and JIT kernels to reduce redundant global memory access and consumption; (4) topology-aware resource allocation (ranking strategy) to minimize the communication across different layers of switches, which is the limitation of a typical fat-tree-topology.

Finetuning Framework Different from pretraining, finetuning LLMs may require the orchestration of multiple models, as is the practice of DPO [59] and PPO [54]. In such training jobs, a typical process is to use reference/reward model to predict a batch of data (which also requires nontrivial time), then let the target model use this data to calculate loss and update parameters. To this end, we build a multi-model scheduling framework to support multiple backends for different LLMs in a single job. For example, when finetuning a language model with DPO, the intermediate results from the reference model can be cached and reused, improving the training speed and resource cost to be close to the supervised finetuning counterparts.

Fast and Efficient Inference We primarily use quantization, dynamic batching, and Paged Attention for improving decoding speed and memory usage. We use quantization to decrease both the memory footprint and computation demand. By 4-bit model quantization [81] and 8-bit KV cache quantization [18], we are able to achieve significant GPU memory saving with near-zero performance degradation (e.g., less than 1% accuracy drop in MMLU/CMMLU benchmark). We use dynamic batching [86] to minimize the response time and improve batching efficiency. We use PagedAttention[41] to improve memory utilization and improve decoding.

Long-context Window Support We implement and improve computation-communication overlapping, sequence parallelism, and communication compression to support up to 200K context length continue pretraining and finetuning. Our method to scale the context length to 200K is *solely* based on engineering, that is to say, we do not modify the model architecture like sparse, local, or sliding window attention – the model remains using the full attention even the input is 200K.

5 Safety

To enhance the model's trustworthiness and safety, we develop a full-stack Responsible AI Safety Engine (RAISE). RAISE ensures safe pretraining, alignment, and deployment. This section discusses our safety measures in the pretraining and alignment stages.

Safety in Pretraining Aligning with standard pretraining data safety practices [5, 58, 77], we build a set of filters based on heuristic rules, keyword matching, and learned classifiers to remove text containing personal identifiers and private data, and reduce sexual, violent, and extremist content.

Safety in Alignment Informed by existing research in [24, 35], we first build a comprehensive safety taxonomy. This taxonomy covers a broad spectrum of potential concerns, including environmental disharmony, superstitious, religious sensitivities, discriminatory practices, substance abuse, violent behavior, illegal activities, hate speech, ethical violations, privacy breaches, self-harm, sexually explicit content, mental health issues, and cybersecurity threats. We curated datasets reflecting these categories for a robust alignment, and mix them with our dialog SFT data. We also include a targeted set of prompts simulating attack scenarios in the alignment phase, which effectively improved the model's resilience against malicious use.

6 Evaluations

Our evaluation demonstrates that the Yi model family achieves inspiring performance on a wide range of tasks and delivers close to GPT-3.5 user preference rate. We first report the base model performance on standard benchmarks, then we discuss the chat model performance and its user preference rate.

6.1 Base Model Performance

6.1.1 Main Results

Here we present the results for our base models and several other well-known base models across standard academic benchmarks. While benchmarking open-source models, we observed a disparity between the results generated by our pipeline and those reported in public sources. Upon conducting a more in-depth investigation of this difference, mostly because different models use different prompts, post-processing strategies, and sampling techniques. These differences may potentially induce significant variations in the outcomes. Our prompt and post-processing strategy remains consistent with the default settings of the original benchmarks[2, 4, 7, 8, 10–12, 27, 28, 42, 50, 61–63, 72, 74, 75, 89, 90]. We use greedy decoding without any post-processing for the generated content. For scores that were not reported publicly (or scores reported with different settings), we try to get results with our pipeline. For scores that can be found publicly, we directly report the existing numbers. We use the following benchmarks, largely following the practice of LLaMA 2 [77]:

Commonsense Reasoning: We included PIQA[4], SIQA[63], HellaSwag[89], WinoGrande [62], ARC[11], OpenBookQA(OBQA)[50], and CommonsenseQA(CSQA)[75] to assess common sense reasoning. CSQA was exclusively tested using a 7-shot setup, while all other tests were conducted with a 0-shot configuration.

Reading Comprehension: For reading comprehension, we report the 0-shot average on SQuAD[61], QuAC[8], and BoolQ[10].

Math: We report the average of the GSM8K[12] (8 shot), and MATH[28] (4 shot) benchmarks with pass@1 accuracy without any specific prompting strategy (e.g. Chain-of-Thought prompting) and other ensemble technique (e.g., majority voting).

Model	Size	GSM8k	MATH	Human-Eval pass@1	MBPP pass@1
GPT-3.5	-	57.1	14.0	48.1	61.4
GPT-4	-	**92.0**	**40.2**	**67.0**	**63.6**
Falcon	180B	54.4	-	0.61	47.0
Qwen	7B	51.7	11.6	29.9	34.0
	14B	61.3	24.8	32.3	48.9
Baichuan 2	7B	24.5	5.6	18.3	28.3
	13B	22.1	10.1	20.7	26.1
LLaMA 2	7B	16.7	3.3	12.8	14.8
	34B	42.2	6.2	22.6	33.0
	70B	56.8	13.5	31.7	45.0
Mistral	7B	47.5	11.3	30.5	47.5
InternLM	20B	62.9	10.9	28.1	41.4
Skywork	7B	55.8	7.8	13.4	22.8
Yi	6B	32.5	4.6	15.9	26.3
	34B	67.2	14.4	23.2	41.0

Table 3: Comparison of models on GSM8k, MATH, Human-Eval, and MBPP.

Code: We report the average pass@1 scores of our models on HumanEval[7] (Chen et al., 2021) and MBPP[2] (Austin et al., 2021).

Popular Aggregated Benchmark: We report the overall results for MMLU[27](5-shot), CMMLU[42] (5-shot), Gaokao-Bench[90] (5-shot), and BigBench[72] Hard (BBH[74]) (3-shot).

By training on a significantly larger number of tokens (3.1T) compared to prior work (usually $\leq$ 2T), we have observed a substantial performance gain across benchmarks, as shown in Table 2. However, it is important to note that there are still discernible disparities between our model and existing open-source and close-source models, particularly in tasks related to mathematics and coding. As performance in these domains can be significantly improved by continual pretraining and instruction fine-tuning, we have refrained from incorporating extensive mathematical and coding content in the pretraining corpus when making the initial design choices. We do plan to release models with enhanced math and coding capabilities in the future.

6.1.2 Discussions

Gain from Model Scale. We observe that Yi-34B has substantial performance improvement compared to Yi-6B, though they utilized the same pretrain corpora. Larger model size leads to higher performance gain on Code and Math benchmarks, referring to Tab. 3, compared to benchmarks focusing on Commonsense Reasoning, Reading Comprehension, or Knowledge.

Data Quality. Smaller models of higher quality pretrain data, like Yi-34B or Qwen-14B, usually demonstrate better performance than models of larger size but (presumably) lower quality data, such as Falcon-180B (though the focus of Falcon-180B might be more on the scaling side, which is definitely of important value on its own).

Gap between GPT-4 and Open-source LLMs. Based on Tab. 2, we note that open-source LLMs still lag behind the performance of GPT-4 and GPT-3.5 on various benchmarks. Yet representative bilingual LLMs, e.g. Qwen-14B and Yi-34B, can match or even surpass the performance of GPT-4 on Chinese knowledge related benchmarks, including C-Eval [31], CMMLU [42], and Gaokao [90]. However, there is still a huge gap between GPT-4 and open-source models on reasoning-related benchmarks like BBH [72], code (HumanEval), and math (MATH).

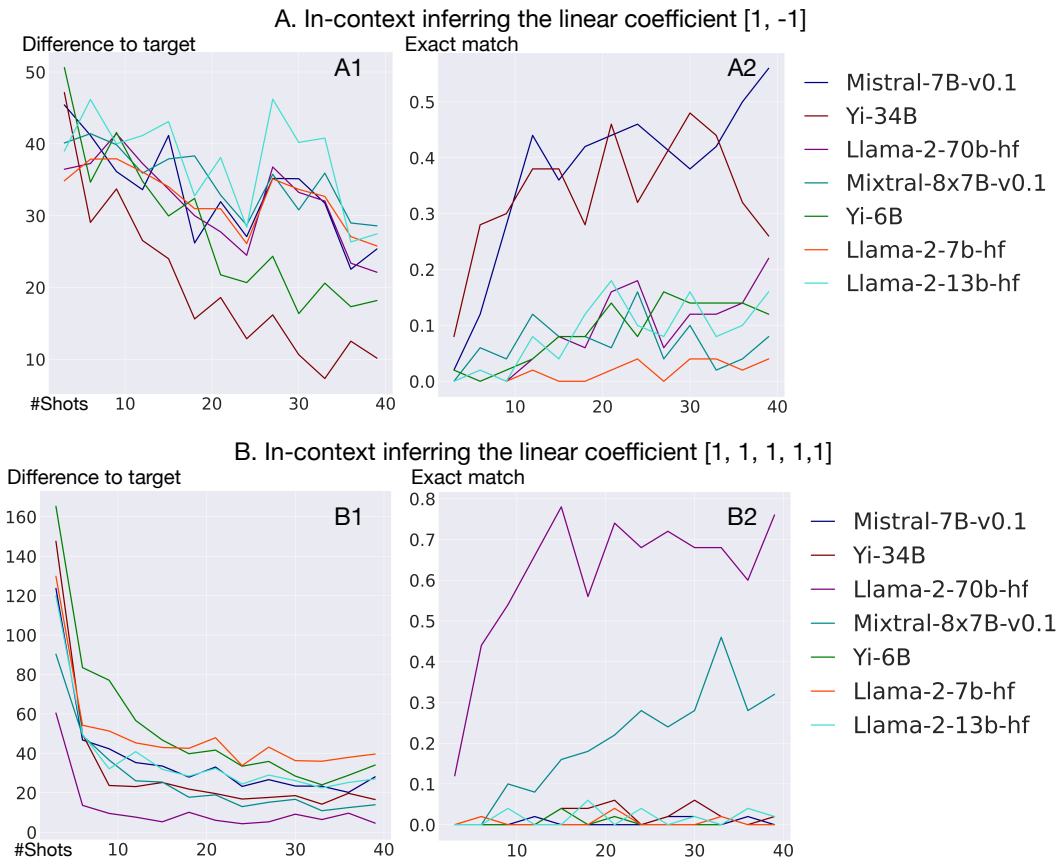

Figure 3: Evaluating language model's in-context learning capability by inferring the linear coefficients of a weighted sum. Considering the discussions of whether emergent ability is an artifact of measurement [65], we use difference to the target (target number - model prediction) as a continuous measure, and exact match (target number == model prediction) as a discontinuous measure. A: when there is two linear coefficients, Yi-34B performs the best when measuring by the difference to the target number. B: increasing the number of linear coefficients to 5, only models that are large enough (LLaMA2 70B and Mixtral 8x7B) can achieve meaningful exact match, showing that in-context learning complex functions is an emergent ability.

6.1.3 In-Context Learning Study

We further investigate the in-context learning capability, i.e., the capability of inferring the underlying function given the few-show input-output demonstrations. We consider the task of inferring the linear coefficient of a weighted sum. Specifically, define $y = w_1x_1 + w2x_2 + ... + w_nx_n$, our few-shot demonstration is $x_1, x_2, ..., x_n, y$, and we ask the model to (implicitly) infer $w_1, w_2, ..., w_n$ by predicting the y given a new set of input x. We use (a). the absolute difference between model prediction y and the ground truth y^*, i.e., $|y - y^*|$ as a continuous measure, and use (b). the exact match $y == y^*$ as a discontinuous measure. We further note that most of the models perform reasonably well on addition and subtraction, so the ability to do arithmetic, as a confounding factor, can be ruled out.

The results are shown in Figure 3. When setting the linear coefficients of be [1, -1], we see that Yi 34B and LLaMA-2 70B performs the best in-terms of answer exact match. If we increase the number of the linear coefficients to be [1, 1, 1, 1, 1], we observe the emergent behavior that only large models (LLaMA-2 70B and Mixtral) can achieve good scores on exact match, although the differences to target is more continuous. These observations give side evidence for Yi-34B's performance on in-context learning and indicates that further scaling may allow the model to infer more complicated functions by in-context learning.

6.2 Chat Model Performance

In this section, we report the automatic and human preference evaluation of the Chat Model. We use greedy decoding to generate responses. For the automatic evaluation benchmarks, we extract answers from the model's generated outputs and calculate accuracy. During the evaluation process, we observed that different prompts have varying influence on results. Therefore, for the same set of questions, we use identical prompts to evaluate all models, aiming to ensure as fair and unbiased results as possible.

6.2.1 Automatic Evaluations

For automatic evaluation, we use the same benchmarks as is for the base model, detailed in Sec. 6.1.1. We use both zero-shot and few-shot methods but generally, zero-shot is more suitable for chat models. Our evaluation involves generating responses while following instructions explicitly or implicitly (such as the format in the few-shot examples). We then isolate relevant answers from the generated text. Unlike the base model, for the zero-shot evaluations on the GSM8K and BBH datasets, we employ the Chain-of-Thought (CoT) approach to guide the model in deliberation before reaching an answer.

The results shown in Tab. 4 demonstrate the effectiveness of our chat models in understanding human instructions and generating appropriate instruction-following responses. We particularly highlight the 4-bit quantization results, as 4-bit quantization substantially reduces the memory requirement while the model performance nearly does not drop. This observation serve as the foundation of serving the model on consumer-grade devices.

In line with Goodhart's principle, when a measurement metric becomes the target of our pursuit, it ceases to serve as a reliable standard of assessment. Consequently, the outcomes of our evaluations on benchmarks are exclusively employed for ensuring that our alignment training does not detrimentally impact the foundational knowledge and capabilities of the base model. We do not engage in targeted optimization of our chat model with the objective of enhancing benchmark performance.

To further evaluate the generalizability of our model's capabilities, we conducted assessments of its mathematical computation proficiency by subjecting it to the 2023 Hungarian high school mathematics final exam questions, first proposed by the xAI Grok team then reproduced by Paster [55]. This evaluation was undertaken with the aim of determining whether our model exhibited signs of overfitting to training datasets that are mathematically oriented. The results in Fig. 4 show that Yi-34B-Chat performs inspiringly on both the GSM8K and the Hungarian mathematics exam. However, note that Yi-6B-Chat does not exhibit strong mathematical capabilities (on both GSM8K and the Hungarian mathematics exam). We speculate that smaller models may require more data to activate their corresponding abilities during the SFT stage.

6.2.2 Human Evaluations

In this section we conducted an assessment of the model's conversational abilities, considering aspects to ensure its effectiveness and safety. We have compiled a collection of open-source evaluation datasets from the community, such as alpaca-eval[21], Belle-eval [88], and MT-bench[93]. Additionally, we have established our own helpful and harmless evaluation dataset by gathering and constructing data of varying difficulty levels, for the purpose of comprehensively assessing the conversational abilities of chat models.

However, whether it is a public evaluation set or a self-built evaluation set, the evaluation results are strongly influenced by the assessment criteria and the design of the prompt. Our internal evaluation results may be unfair to other models, making it difficult to accurately represent the true capability level of our model. Therefore, here we only present external evaluation results to demonstrate the current conversational abilities of our chat model. We consider: (1). AlapcaEval[1] [44], which is designed to assess the English conversation capabilities of models by comparing the responses of a specified model to reference replies from Davinci003 [21] in order to calculate a win-rate; (2). LMSys[2] [93] Chatbot Arena, which showcases the responses of different models through a dialogue platform, then asks users to make selections based on their preferences, then computes the Elo score;

[1] https://tatsu-lab.github.io/alpaca_eval/

[2] https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard

Model	Size	MMLU	CMMLU	C-Eval(val)	TruthfulQA	BBH	GSM8K
		0-shot / 5-shot	0-shot / 5-shot	0-shot / 5-shot	0-shot	0-shot / 3-shot	0-shot / 4-shot
LLaMA2-Chat	13B	50.9 / 47.3	27.5 / 35.1	27.9 / 35.9	36.8	32.9 / 58.2	36.9 / 2.7
	70B	59.4 / 59.9	36.1 / 41.0	35.0 / 41.3	54.0	42.4 / 58.5	47.1 / 58.7
Baichuan2-Chat	13B	55.1 / 50.1	58.6 / 59.5	56.0 / 54.8	49.0	38.8 / 47.2	45.7 / 23.3
Qwen-Chat	14B	64.0 / 65.0	67.7 / 70.6	66.1 / 70.1	52.5	49.7 / 55.0	59.5 / 61.2
InternLM-Chat	20B	55.6 / 57.4	53.6 / 53.8	51.2 / 53.6	51.8	42.4 / 36.7	15.7 / 43.4
AquilaChat2	34B	65.2 / 66.7	67.5 / 70.0	**83.0 / 89.4**	**64.3**	20.1 / 34.3	11.5 / 48.5
Yi-Chat	6B	58.2 / 61.0	69.4 / 74.7	68.8 / 74.2	50.6	39.7 / 47.2	38.4 / 44.9
Yi-Chat-8bits(GPTQ)	6B	58.3 / 61.0	69.2 / 74.7	69.2 / 73.9	49.9	40.4 / 47.3	39.4 / 44.9
Yi-Chat-4bits(AWQ)	6B	56.8 / 59.9	67.7 / 73.3	67.5 / 72.3	50.3	37.7 / 43.6	35.7 / 38.4
Yi-Chat	34B	**67.6** / 73.5	**79.1 / 81.3**	77.0 / 78.5	62.4	51.4 / **71.7**	**71.7 / 76.0**
Yi-Chat-8bits(GPTQ)	34B	66.2 / **73.7**	79.1 / 81.2	76.8 / 79.0	61.8	**52.1** / 71.0	70.7 / 75.7
Yi-Chat-4bits(AWQ)	34B	65.8 / 72.4	78.2 / 80.5	75.7 / 77.3	61.8	48.3 / 69.4	70.5 / 74.0

Table 4: Overall performance on automatic benchmarks compared to open-source chat models. We highlight the 4-bit quantization results, as 4-bit quantization substantially reduces the memory requirement while the model performance nearly does not drop. This observation serve as the foundation of serving the model on consumer-grade devices, e.g., RTX4090.

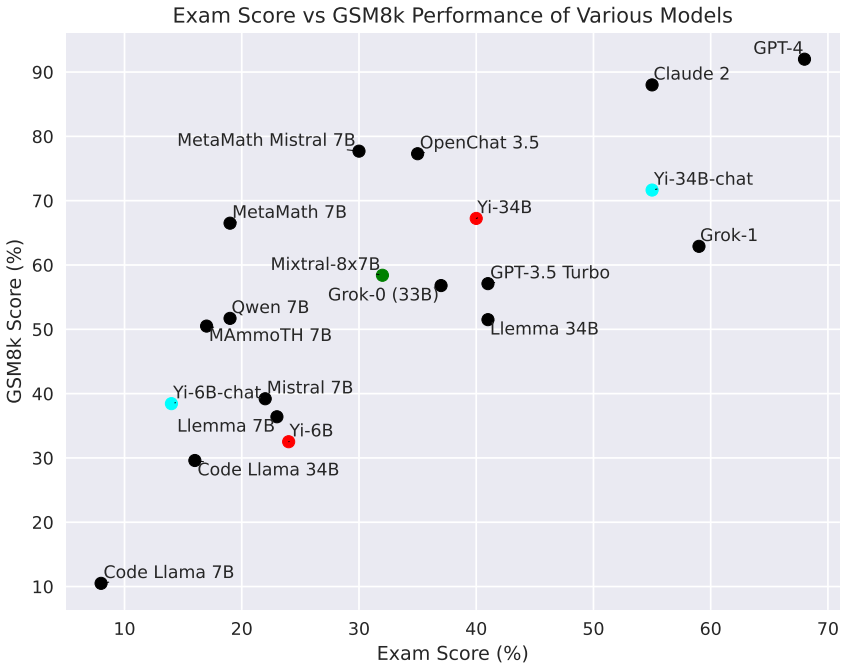

Figure 4: Yi's result of Hungarian mathematics exam.

(3). SuperClue[3], on the other hand, is a leaderboard aimed at comprehensively evaluating the Chinese language capabilities of models.

Tab. 5 presents the performance results of Yi-34B-Chat in the three third-party evaluations we consider, with the cutoff date for the results being December 21, 2023. The data demonstrates that, although there is still a gap compared to GPT-4, our model exhibits proficient bilingual (Chinese and English) dialogue capabilities and aligns well with user preferences. Additional comparative results of various models are accessible for review on the official website.

[3]https://www.superclueai.com/

Model	Size	AlpacaEval	LMSys Chatbot Arena	SuperClue
GPT-4-Turbo	-	**97.7**	**1243**	**89.79**
GPT-3.5-Turbo	-	89.37	1117	59.39
LLaMA2-Chat	70B	92.66	1077	-
Yi-Chat	34B	94.08	1110	71.87

Table 5: Human evaluation comparison with other open-source chat models.

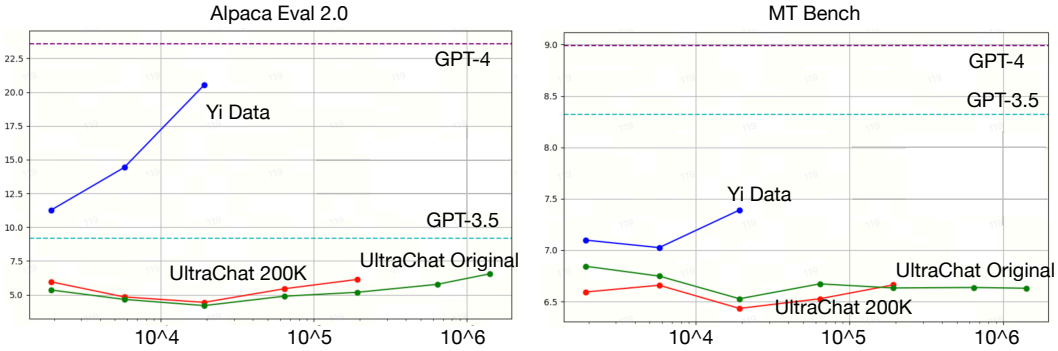

Figure 5: SFT data scaling curve. Compared with UltraChat and its cleaned version UltraChat 200K, our SFT data demonstrates clear scaling advantages. We attribute its steep slope to the data quality.

We further demonstrate the data quality by comparing the speed of preference increase during data scaling. As is shown in Fig. 5, when compared with UltraChat [19] and its cleaned version UltraChat 200K, we see a clear tendency of performance improvements when scaling up the Yi data.

7 Capability Extension

In this section, we discuss our post-training methods to extend the Yi base model to 200K long-context, equip it with visual understanding capability, and enhance the 6B model by depth upsacaling.

7.1 Long Context Modeling

Our long-context solution consists of a continual pretraining and a finetuning phase, both are light-weight. We hold the basic hypothesis that the potential of utilizing information anywhere within the 200K input context is already exist in the base model (same as Fu et al. 22), the continue pretraining phase "unlocks" such capability, evidenced by a strong performance on Needle-in-a-Haystack test, then the finetuning phase further adapt the style of response to follow human instruction and preference.

Continue Pretraining We continue pretrain the full-attention model using sequence parallelism [43] and distributed attention. This is to say, we do not use any sparse or linear attention, but use a brute force implementation of the full attention. We continue pretrain the Yi 6B/ 34B base model on the data mixture of (1). original pretraining data, as is introduced in section 2; (2). length-upsampled long-context data, where the long documents are mostly from books; (3). multi-document question-answering synthetic data, where we construct QA pairs where the answer contains a recitation of the related paragraph before the answer. Our data approach mostly follows the data engineering practice in Fu et al. [22] and Yu et al. [87]. We continue pretrain the model on 5B tokens with 4M batch size, which translate to 100 optimization steps. Aligning with the concurrent work from Fu et al. [22], we observe that such light-weight continue pretraining is already able to enable a strong performance on Needle-in-a-Haystack test, as we will show in Figure 6.

Supervised Finetuning We mix our short-context SFT data with long-context document question-answering data. We use model-assisted automated methods (i.e., synthetic data) to construct document QA. Specifically, we randomly concatenate multiple documents into a sequence, sample one or more paragraphs from the long sequence, and ask a chat model to construct question and answer pairs

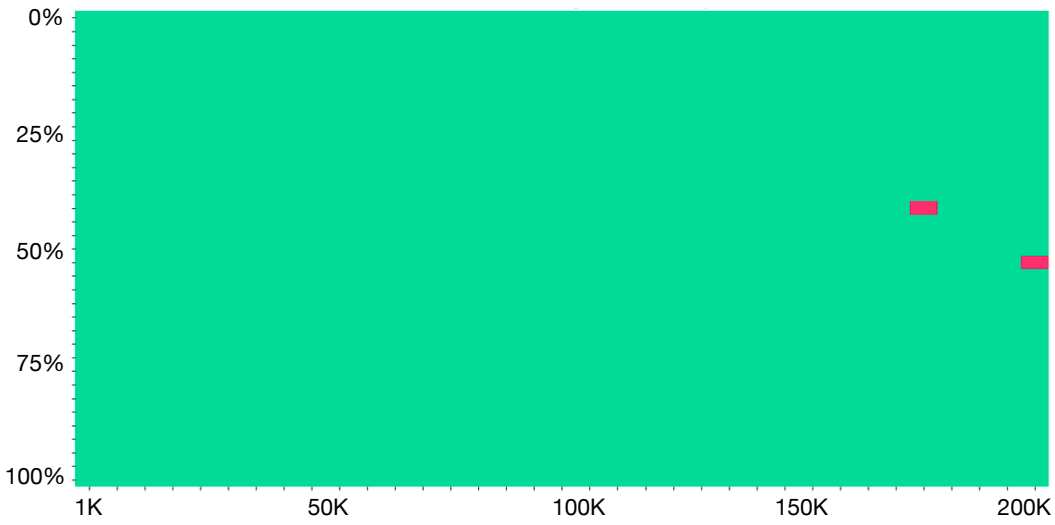

Figure 6: Needle-in-a-Haystack performance of Yi-34B-200K. X-axis means length of the document, and Y-axis means the depth of the needle sentence within the document. We continue pretrain the model on 5B tokens long-context data mixture and demonstrates a near-all-green performance.

Model	**Average**	**Humanity**	**STEM**	**Social Science**	**Other**
Yi-6B 4K	63.24	59.10	53.15	73.83	69.26
Yi-6B 200K	61.73	56.17	52.36	72.54	68.94
Yi-34B 4K	76.32	73.17	68.03	85.11	80.78
Yi-34B 200K	75.56	72.20	66.83	84.76	80.40

Table 6: Performance on MMLU after 200K adaptation. Extending the context length to 200K does not significantly change the short context capability.

based on the sampled paragraph. One important detail is recitation and rephrasing: before giving the answer, we ask the model to recite or paraphrase the original paragraph. This data format encourages the model's retrieval behavior and consequently discourages the hallucination behavior: given a question, the model is more likely to use the information within the input to construct the answer, rather than use its internal knowledge, which may be related but inaccurate. Our finetuned model is deployed at `www.wanzhi01.com`, and we encourage the readers to try it out.

The performance of the 200K models is shown in figure. 6 and table 6. Specifically, Figure 6 shows the famous Needle-in-a-Haystack test of Yi-34B-200K, though we tend to view that this level of retrieval is relatively easy for long-context LLMs. Table 6 shows that our context scaling does not significantly influence the short-context generic capability.

7.2 Vision-Language

In the burgeoning field of multimodal research, the integration of image understanding capabilities into large language models has become increasingly viable. Drawing inspiration from the open-sourced LLaVA [46, 47], we present Yi Vision Language (Yi-VL) models, *i.e.*, Yi-VL-6B and Yi-VL-34B, based on Yi-6B-Chat and Yi-34B-Chat language models. The architecture of Yi-VL models, as illustrated in Figure 7, comprises three primary modules. The Vision Transformer (ViT), used for image encoding, is initialized with CLIP ViT-H/14 model [33]. A Projection Module, designed to align image features with text feature spcae, consists of a two-layer Multilayer Perceptron (MLP) with layer normalizations. Finally, the large language model, initialized with the Yi-Chat models, demonstrating exceptional proficiency in understanding and generating both English and Chinese. To enhance the performance of Yi-VL models in bilingual multimodal understanding and generation, we leverage a rich dataset of bilingual image-text pairs.

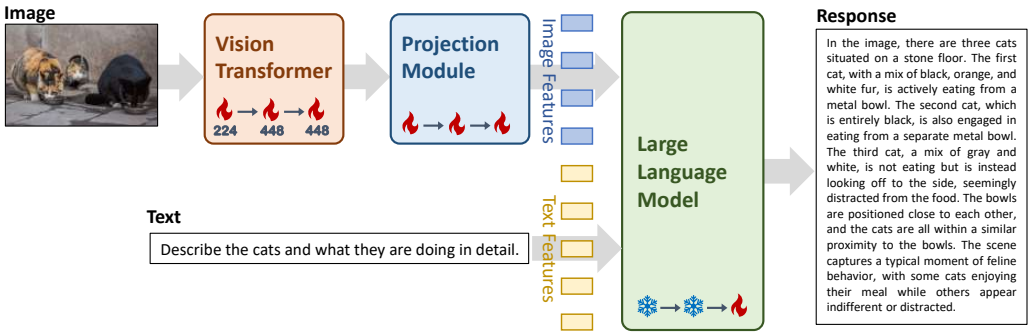

Figure 7: Architecture of Yi-VL models. Symbols are used to denote the training status of various modules at three training stages: a fire icon (🔥) indicates the parameters of the module are trainable, while a snowflake icon (❄) signifies that parameters are frozen. The image resolution used in ViT at each stage, either 224^2 or 448^2, is also marked.

Yi-VL models undergo a three-stage training process:

Stage 1: we train the parameters of the ViT and the projection module using an image resolution of 224^2. The training leverages a substantial dataset comprising 100 million image-text pairs from LAION-400M [66]. The primary objective is to enhance the ViT's knowledge acquisition within our specified architecture and to achieve better alignment between the ViT and the LLM.

Stage 2: we scale up the image resolution of ViT to 448^2, aiming to further boost the model's capability for discerning intricate visual details. The dataset used in this stage includes 20 million image-text pairs derived from LAION-400M. Additionally, we incorporate around 4.8 million image-text pairsn from diverse sources, *e.g.*, CLLaVA [45], LLaVAR [91], Flickr [85], VQAv2 [25], RefCOCO [37], Visual7w [95] and so on.

Stage 3: the parameters of the entire model are trained. The primary goal is to enhance the model's proficiency in multimodal chat interactions, thereby endowing it with the ability to seamlessly integrate and interpret visual and linguistic inputs. To this end, the training dataset encompasses a diverse range of sources, totalling approximately 1 million image-text pairs, including GQA [32], VizWiz VQA [26], TextCaps [71], OCR-VQA [51], Visual Genome [39], ShareGPT4V [6] and so on. To ensure data balancing, we impose a cap on the maximum data contribution from any single source, restricting it to no more than $50,000$ pairs.

In Stage 1 and 2, we set the global batch size, the learning rate, the gradient clip and the number of epoch to 4096, $1e{-}4$, 0.5 and 1, respectively. In Stage 3, these parameters are adjusted to 256, $2e{-}5$, 1.0 and 2. The training consumes 128 NVIDIA A100 GPUs. The total training time amounted to approximately 3 days for Yi-VL-6B and 10 days for Yi-VL-34B.

Table 7 shows the MMMU test set leaderboard by Yi-VL's release. We note that this area is currently actively under research, aligning with the community's advances, we will continuously improve the update Yi-VL's performance.

7.3 Depth Upscaling

Recent studies on scaling laws [29, 30, 36] have underscored the predictable improvement in model performance with increases in computational budget, model size, and data size. Yet, identifying the most effective distribution of resources between model and data sizes upon expanding the computational budget remains a formidable challenge in the field of scaling laws. Additionally, research conducted by DeepSeek-AI et al. [17] has highlighted that the allocation of an increased computational budget towards model scaling should be proportional to the quality of the data available. In light of these insights, we propose a novel approach aimed at dynamically adjusting the resource allocation between data and model sizes through a series of staged training processes. This strategy iteratively fine-tunes the balance between data characteristics and model size according to scaling laws, enhancing both model training efficiency and performance.

Model	Overall	Art	Business	Science	Health	Society	Engineering
GPT-4V	55.7	65.3	64.3	48.4	63.5	76.3	41.7
Yi-VL-34B	41.6	56.1	33.3	32.9	45.9	66.5	36.0
Qwen-VL-PLUS	40.8	59.9	34.5	32.8	43.7	65.5	32.9
Marco-VL	40.4	56.5	31.0	31.0	46.9	66.5	33.8
Yi-VL-6B	37.8	53.4	30.3	30.0	39.3	58.5	34.1
InfMIM-Zephyr-7B	35.5	50.0	29.6	28.2	37.5	54.6	31.1
SVIT	34.1	48.9	28.0	26.8	35.5	50.9	30.7
Emu2-Chat	34.1	50.6	27.7	28.0	32.4	50.3	31.3
BLIP-2 FLAN-T5-XXL	34.0	49.2	28.6	27.3	33.7	51.5	30.4
InstructBLIP-T5-XXL	33.8	48.5	30.6	27.6	33.6	49.8	29.4
LLaVA-1.5-13B	33.6	49.8	28.2	25.9	34.9	54.7	28.3
Qwen-VL-7B-Chat	32.9	47.7	29.8	25.6	33.6	45.3	30.2
SPHINX*	32.9	50.9	27.2	25.3	34.1	51.2	27.8
mPLUG-OWL2	32.1	48.5	25.6	24.9	32.8	46.7	29.6
BLIP-2 FLAN-T5-XL	31.0	43.0	25.6	25.1	31.8	48.0	27.8
InstructBLIP-T5-XL	30.6	43.3	25.2	25.2	29.3	45.8	28.6
CogVLM	30.1	38.0	25.6	25.1	31.2	41.5	28.9

Table 7: MMMU test set performance by the time of Yi-VL's release.

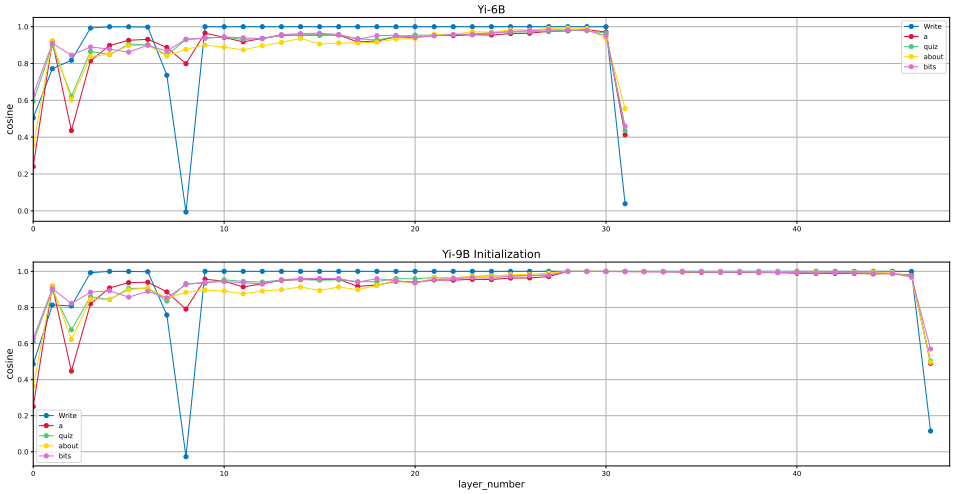

Figure 8: Input/output cosine similarity score of each token per layer for text "Write a quiz about bits". The cosine similarity scores of the 16 newly added layers(layers 28-44), as depicted in the lower figure, are observed to be nearly 1.

Method Following the methodology outlined by Kim et al. [38], our goal is to upscale our Yi-6B base model, which has 32 layers, to a 9B model named the Yi-9B base model, featuring 48 layers, by duplicating the original 16 middle layers 12-28. Depth up-scaling involves expanding the base model's depth and subsequently continuing the pretraining phase for the enhanced model.

Our investigations reveal that the decision on which layers to replicate could be informed by evaluating the cosine similarity scores between the inputs and outputs of each layer. Such an approach allows for targeted model scaling without necessitating additional pretraining, leading only to minimal performance impacts. This minimal impact on performance is attributed to the high cosine similarity, approaching one, between the inputs and outputs of the duplicated layers, as evidenced in Figure 8. This observation suggests that the replication of these layers does not significantly alter the output

Model	Arc-C	HellaSwag	MMLU	Winogrande	GSM8K	MATH	HumanEval	MBPP
Yi-6B	50.3	74.4	63.2	71.3	32.5	4.6	15.9	26.3
Yi-9B Init	52.1	73.3	63.0	69.4	31.3	4.1	12.8	25.8
Yi-9B	**55.6**	**76.4**	**68.4**	**73.0**	**52.3**	**15.9**	**39.0**	**54.4**

Table 8: Performance between Yi-6B and Yi-9B: Arc Challenge (25-shot), HellaSwag (10-shot) MMLU (5-shot), Winogrande (5-shot), GSM8K (5-shot), MATH (4-shot), HumanEval pass@1, MBPP pass@1(3-shot). Yi-9B Init is just depthwise upscaling from Yi-6B by duplicating layers 12-28 without further training.

logits produced by the original model. This method ensures the efficient scaling of the model by optimizing its architecture based on the internal processing dynamics of its layers.

Continual Training The dataset is composed of approximately 800 billion tokens across two stages, with around 70% having been recently collected and carefully selected. We have enhanced the code coverage in the final stage to improve code performance.

To optimize the training process, we maintain a constant learning rate of 3e-5, and adopt a strategic approach to gradually increase the batch size from 4M tokens whenever the model's loss plateaued. This incremental adjustment of the batch size, alongside maintaining all other parameters in alignment with the established Yi-6B base model configuration, was instrumental in navigating the challenges of training at scale.

The effectiveness of these strategies is demonstrated in Table 8, which details the Yi-9B base model's performance across a variety of benchmarks, including common sense, reasoning, knowledge, coding, and mathematics. It underscores the competitive advantages of Yi-9B base model in specific domains, illustrating the efficacy of our methodology in enhancing model performance by optimally adjusting the interplay between data characteristics and model size.

8 Final Discussions

In this report, we discuss the full-stack development of the Yi language model family. Yi-34B achieves GPT-3.5 matching performance and is deployable (thank to the 4/8-bit quantization) on consumer-grade devices, making it an ideal model for local deployment.

The key takeaways from the Yi pretraining procedure are about data quantity and quality: (1). training the model on a larger amount of data than the Chinchilla optimal delivers clear and consistent performance gain, which we highly recommend for all pretraining teams. Our model is trained on 3.1T tokens, yet we belive with larger amount of data, we can continue improve the model performance (i.e., the model have not saturated at 3.1T); (2). when it comes to the pretraining data quality, we believe the most critical two factors are the source of the data (e.g., whether the text is produced for professional usage or for casual social media posting) and the details of the data cleaning (e.g., the strength of filtering and deduplication). Since data cleaning is a very complicated pipeline and it is extremely difficult to conduct extensive grid-search styled optimizations, our current solution may still have room for improvements.

The key takeaways from the Yi finetuning procedure is to heavily iterate on a small amount of data ($\leq$ 10K), case by case, over multiple iterations, directly by the machine learning engineer, and improved from real user feedback. This approach clearly deviates from the instruction-scaling approach, initially introduced by the FLAN series [9] then followed by the UltraChat series [19].

As is demonstrated by our current results, the reasoning capability, which we view as the core capability for real-world deployment of language models, is strongly correlated with model scale when the amount of pretraining data is fixed. We believe that given our current results, continuing to scale up model parameters using thoroughly optimized data will lead to even stronger frontier models in our upcoming next versions.

A Author List and Contributions

Our team members contribute to the development of Yi from the following perspectives:

- Frontier Research
- Machine Learning Infrastructure
- Pretraining
- Finetuning and AI Alignment
- Multimodal
- Safety and Responsible AI
- Deployment

We list our team members in alphabetical order. All authors contributed equally to this work.

- Alex Young
- Bei Chen
- Chao Li
- Chengen Huang
- Ge Zhang
- Guanwei Zhang
- Heng Li
- Jiangcheng Zhu
- Jianqun Chen
- Jing Chang
- Kaidong Yu
- Peng Liu
- Qiang Liu
- Shawn Yue
- Senbin Yang
- Shiming Yang
- Tao Yu
- Wen Xie
- Wenhao Huang
- Xiaohui Hu
- Xiaoyi Ren
- Xinyao Niu
- Pengcheng Nie
- Yuchi Xu
- Yudong Liu
- Yue Wang
- Yuxuan Cai
- Zhenyu Gu
- Zhiyuan Liu
- Zonghong Dai

References

[1] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

[2] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program Synthesis With lLarge Language Models. *arXiv preprint arXiv:2108.07732*, 2021.

[3] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen Technical Report. 09 2023. URL `https://arxiv.org/pdf/2309.16609.pdf`.

[4] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about Physical Commonsense in Natural Language. *ArXiv*, abs/1911.11641, 2019. URL `https://api.semanticscholar.org/CorpusID:208290939`.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[6] 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.

[7] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating Large Language Models Trained on Code. *CoRR*, abs/2107.03374, 2021. URL `https://arxiv.org/abs/2107.03374`.

[8] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. QuAC : Question Answering in Context, 2018.

[9] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

[10] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions, 2019.

[11] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge, 2018.

[12] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021.

[13] Together Computer. Redpajama: an open dataset for training large language models, 2023. URL https://github.com/togethercomputer/RedPajama-Data.

[14] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. 2023.

[15] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

[16] Michiel de Jong, Yury Zemlyanskiy, Joshua Ainslie, Nicholas FitzGerald, Sumit Sanghai, Fei Sha, and William Cohen. FiDO: Fusion-in-Decoder Optimized for Stronger Performance and Faster Inference. *arXiv preprint arXiv:2212.08153*, 2022.

[17] DeepSeek-AI, :, Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, Ying He, Wenjie Hu, Panpan Huang, Erhang Li, Guowei Li, Jiashi Li, Yao Li, Y. K. Li, Wenfeng Liang, Fangyun Lin, A. X. Liu, Bo Liu, Wen Liu, Xiaodong Liu, Xin Liu, Yiyuan Liu, Haoyu Lu, Shanghao Lu, Fuli Luo, Shirong Ma, Xiaotao Nie, Tian Pei, Yishi Piao, Junjie Qiu, Hui Qu, Tongzheng Ren, Zehui Ren, Chong Ruan, Zhangli Sha, Zhihong Shao, Junxiao Song, Xuecheng Su, Jingxiang Sun, Yaofeng Sun, Minghui Tang, Bingxuan Wang, Peiyi Wang, Shiyu Wang, Yaohui Wang, Yongji Wang, Tong Wu, Y. Wu, Xin Xie, Zhenda Xie, Ziwei Xie, Yiliang Xiong, Hanwei Xu, R. X. Xu, Yanhong Xu, Dejian Yang, Yuxiang You, Shuiping Yu, Xingkai Yu, B. Zhang, Haowei Zhang, Lecong Zhang, Liyue Zhang, Mingchuan Zhang, Minghua Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, Qihao Zhu, and Yuheng Zou. Deepseek llm: Scaling open-source language models with longtermism. 2024.

[18] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*, 2022.

[19] Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*, 2023.

[20] Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. How abilities in large language models are affected by supervised fine-tuning data composition, 2023.

[21] Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback, 2023.

[22] Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. Data engineering for scaling language models to 128k context. *arXiv preprint arXiv:2402.10171*, 2024.

[23] Gemini 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.

[24] Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, et al. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*, 2022.

[25] 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*, pages 6904–6913, 2017.

[26] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3608–3617, 2018.

[27] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. *CoRR*, abs/2009.03300, 2020. URL `https://arxiv.org/abs/2009.03300`.

[28] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring Mathematical Problem Solving With the MATH Dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[29] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B. Brown, Prafulla Dhariwal, Scott Gray, Chris Hallacy, Benjamin Mann, Alec Radford, Aditya Ramesh, Nick Ryder, Daniel M. Ziegler, John Schulman, Dario Amodei, and Sam McCandlish. Scaling laws for autoregressive generative modeling. 2020.

[30] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[31] Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, et al. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *arXiv preprint arXiv:2305.08322*, 2023.

[32] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019.

[33] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. URL `https://doi.org/10.5281/zenodo.5143773`.

[34] Neel Jain, Ping-yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, Brian R Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Aniruddha Saha, et al. Neftune: Noisy embeddings improve instruction finetuning. *arXiv preprint arXiv:2310.05914*, 2023.

[35] Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Chi Zhang, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. Beavertails: Towards improved safety alignment of llm via a human-preference dataset, 2023.

[36] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. 2020.

[37] 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.

[38] Dahyun Kim, Chanjun Park, Sanghoon Kim, Wonsung Lee, Wonho Song, Yunsu Kim, Hyeonwoo Kim, Yungi Kim, Hyeonju Lee, Jihoo Kim, Changbae Ahn, Seonghoon Yang, Sukyung Lee, Hyunbyung Park, Gyoungjin Gim, Mikyoung Cha, Hwalsuk Lee, and Sunghun Kim. Solar 10.7b: Scaling large language models with simple yet effective depth up-scaling. 2023.

[39] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123:32–73, 2017.

[40] Taku Kudo and John Richardson. SentencePiece: A Simple and Language Independent Subword Tokenizer and Detokenizer for Neural Text Processing. *arXiv preprint arXiv:1808.06226*, 2018.

[41] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. *arXiv preprint arXiv:2309.06180*, 2023.

[42] Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. CMMLU: Measuring Massive Multitask Language Understanding in Chinese. *arXiv preprint arXiv:2306.09212*, 2023.

[43] Shenggui Li, Fuzhao Xue, Chaitanya Baranwal, Yongbin Li, and Yang You. Sequence parallelism: Long sequence training from system perspective. *arXiv preprint arXiv:2105.13120*, 2021.

[44] Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`, 2023.

[45] LinkSoul-AI. Chinese llava. `https://github.com/LinkSoul-AI/Chinese-LLaVA`, 2023.

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

[47] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

[48] Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. *arXiv preprint arXiv:2312.15685*, 2023.

[49] Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. #instag: Instruction tagging for analyzing supervised fine-tuning of large language models, 2023.

[50] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering, 2018.

[51] Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. Ocr-vqa: Visual question answering by reading text in images. In *2019 international conference on document analysis and recognition (ICDAR)*, pages 947–952. IEEE, 2019.

[52] Thuat Nguyen, Chien Van Nguyen, Viet Dac Lai, Hieu Man, Nghia Trung Ngo, Franck Dernoncourt, Ryan A Rossi, and Thien Huu Nguyen. CulturaX: A Cleaned, Enormous, and Multilingual Dataset for Large Language Models in 167 Languages. *arXiv preprint arXiv:2309.09400*, 2023.

[53] OpenAI. ChatML, 2022. URL `https://github.com/openai/openai-python/blob/e389823ba013a24b4c32ce38fa0bd87e6bccae94/chatml.md`.

[54] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training Language Models to Follow Instructions with Human Feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[55] Keiran Paster. Testing language models on a held-out high school national finals exam. `https://huggingface.co/datasets/keirp/hungarian_national_hs_finals_exam`, 2023.

[56] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only, 2023.

[57] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently Scaling Transformer Inference. *Proceedings of Machine Learning and Systems*, 5, 2023.

[58] Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. Scaling Language Models: Methods, Analysis & Insights from Training Gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[59] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

[60] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory Optimizations Toward Training Trillion Parameter Models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.

[61] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text, 2016.

[62] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An Adversarial Winograd Schema Challenge at Scale, 2019.

[63] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. SocialIQA: Commonsense Reasoning about Social Interactions, 2019.

[64] Nikhil Sardana and Jonathan Frankle. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws. *arXiv preprint arXiv:2401.00448*, 2023.

[65] Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? *Advances in Neural Information Processing Systems*, 36, 2024.

[66] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021.

[67] Noam Shazeer. Fast Transformer Decoding: One Write-Head is All You Need. *arXiv preprint arXiv:1911.02150*, 2019.

[68] Noam Shazeer. GLU Variants Improve Transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[69] Yusuxke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, Takeshi Shinohara, and Setsuo Arikawa. Byte Pair Encoding: A Text Compression Scheme That Accelerates Pattern Matching. Technical report, Technical Report DOI-TR-161, Department of Informatics, Kyushu University, 1999.

[70] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[71] Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh. Textcaps: a dataset for image captioning with reading comprehension. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 742–758. Springer, 2020.

[72] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, Agnieszka Kluska, Aitor Lewkowycz, Akshat Agarwal, Alethea Power, Alex Ray, Alex Warstadt, Alexander W. Kocurek, Ali Safaya, Ali Tazarv, Alice Xiang, Alicia Parrish, Allen Nie, Aman Hussain, Amanda Askell, Amanda Dsouza, Ambrose Slone, Ameet Rahane, Anantharaman S. Iyer, Anders Andreassen, Andrea Madotto, Andrea Santilli, Andreas Stuhlmüller, Andrew Dai, Andrew La, Andrew Lampinen, Andy Zou, Angela Jiang, Angelica Chen, Anh Vuong, Animesh Gupta, Anna Gottardi, Antonio Norelli, Anu Venkatesh, Arash Gholamidavoodi, Arfa Tabassum, Arul Menezes, Arun Kirubarajan, Asher Mullokandov, Ashish Sabharwal, Austin Herrick, Avia Efrat, Aykut Erdem, Ayla Karakaş, B. Ryan Roberts, Bao Sheng Loe, Barret Zoph, Bartłomiej Bojanowski, Batuhan Özyurt, Behnam Hedayatnia, Behnam Neyshabur, Benjamin Inden, Benno Stein, Berk Ekmekci, Bill Yuchen Lin, Blake Howald, Bryan Orinion, Cameron Diao, Cameron Dour, Catherine Stinson, Cedrick Argueta, César Ferri Ramírez, Chandan Singh, Charles Rathkopf, Chenlin Meng, Chitta Baral, Chiyu Wu, Chris Callison-Burch, Chris Waites, Christian Voigt, Christopher D. Manning, Christopher Potts, Cindy Ramirez, Clara E. Rivera, Clemencia Siro, Colin Raffel, Courtney Ashcraft, Cristina Garbacea, Damien Sileo, Dan Garrette, Dan Hendrycks, Dan Kilman, Dan Roth, Daniel Freeman, Daniel Khashabi,

Daniel Levy, Daniel Moseguí González, Danielle Perszyk, Danny Hernandez, Danqi Chen, and Daphne Ippolito et al. (351 additional authors not shown). Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models, 2023.

[73] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced Transformer with Rotary Position Embedding. *arXiv preprint arXiv:2104.09864*, 2021.

[74] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging Big-Bench Tasks and Whether Chain-of-Thought can Solve Them. *arXiv preprint arXiv:2210.09261*, 2022.

[75] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge, 2019.

[76] 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*, 2023.

[77] 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, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models, 2023.

[78] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in Neural Information Processing Systems*, 06 2017. URL `https://arxiv.org/pdf/1706.03762.pdf`.

[79] Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. CCNet: Extracting High Quality Monolingual Datasets from Web Crawl Data. *arXiv preprint arXiv:1911.00359*, 11 2019. URL `https://arxiv.org/pdf/1911.00359.pdf`.

[80] Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. CCNet: Extracting High Quality Monolingual Datasets from Web Crawl Data. *arXiv preprint arXiv:1911.00359*, 2019.

[81] Xiaoxia Wu, Cheng Li, Reza Yazdani Aminabadi, Zhewei Yao, and Yuxiong He. Understanding int4 quantization for transformer models: Latency speedup, composability, and failure cases. *arXiv preprint arXiv:2301.12017*, 2023.

[82] Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, et al. Effective long-context scaling of foundation models. *arXiv preprint arXiv:2309.16039*, 2023.

[83] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023.

[84] Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, JunTao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang

Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. Baichuan 2: Open Large-scale Language Models. 09 2023. URL `https://arxiv.org/pdf/2309.10305.pdf`.

[85] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014.

[86] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A Distributed Serving System for Transformer-Based Generative Models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 521–538, 2022.

[87] Yijiong Yu, Zhe Zhou, Zhixiao Qi, and Yongfeng Huang. Paraphrasing the original text makes high accuracy long-context qa. *arXiv preprint arXiv:2312.11193*, 2023.

[88] Ji Yunjie, Deng Yong, Gong Yan, Peng Yiping, Niu Qiang, Zhang Lei, Ma Baochang, and Li Xiangang. Exploring the impact of instruction data scaling on large language models: An empirical study on real-world use cases. *arXiv preprint arXiv:2303.14742*, 2023.

[89] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a Machine Really Finish Your Sentence?, 2019.

[90] Xiaotian Zhang, Chunyang Li, Yi Zong, Zhengyu Ying, Liang He, and Xipeng Qiu. Evaluating the Performance of Large Language Models on GAOKAO Benchmark. *arXiv preprint arXiv:2305.12474*, 2023.

[91] Yanzhe Zhang, Ruiyi Zhang, Jiuxiang Gu, Yufan Zhou, Nedim Lipka, Diyi Yang, and Tong Sun. Llavar: Enhanced visual instruction tuning for text-rich image understanding. *arXiv preprint arXiv:2306.17107*, 2023.

[92] Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. Take a step back: Evoking reasoning via abstraction in large language models, 2023.

[93] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

[94] Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. Lima: Less is more for alignment, 2023.

[95] Yuke Zhu, Oliver Groth, Michael Bernstein, and Li Fei-Fei. Visual7w: Grounded question answering in images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4995–5004, 2016.