Speculative Decoding and Language Diffusion Models

Alireza Mousavi-Hosseini

University of Toronto

Language Model Inference

- Inference from autoregressive models is slow
- Generating K tokens requires K serial runs
- Can we make it faster by using an approximate model?

Speculative Decoding

Fast Inference from Transformers via Speculative Decoding

Yaniv Leviathan [*1] **Matan Kalman** [*1] **Yossi Matias** [1]

Abstract

Inference from large autoregressive models like Transformers is slow - decoding K tokens takes K serial runs of the model. In this work we introduce *speculative decoding* - an algorithm to sample from autoregressive models faster *without any changes to the outputs*, by computing several tokens in parallel. At the heart of our approach lie the observations that (1) hard language-modeling tasks often include easier subtasks that can be approximated well by more efficient models, and (2) using speculative execution and a novel sampling method, we can make exact decoding from the large models faster, by running them in parallel on the outputs of the approximation models, potentially generating several tokens concurrently, and without changing the distribution. Our method can accelerate existing off-the-shelf models without retraining or architecture changes. We demonstrate it on T5-XXL and show a **2X-3X** acceleration compared to the standard T5X implementation, with identical outputs.

developed to make inference from them faster. Some approaches aim to reduce the inference cost for *all* inputs equally (e.g. Hinton et al., 2015; Jaszczur et al., 2021; Hubara et al., 2016; So et al., 2021; Shazeer, 2019). Other approaches stem from the observation that not all inference steps are born alike - some require a very large model, while others can be approximated well by more efficient models. These *adaptive computation* methods (e.g. Han et al., 2021; Sukhbaatar et al., 2019; Schuster et al., 2021; Scardapane et al., 2020; Bapna et al., 2020; Elbayad et al., 2019; Schwartz et al., 2020) aim to use less compute resources for easier inference steps. While many of these solutions have proven extremely effective in practice, they usually require changing the model architecture, changing the training-procedure and re-training the models, and don't maintain identical outputs.

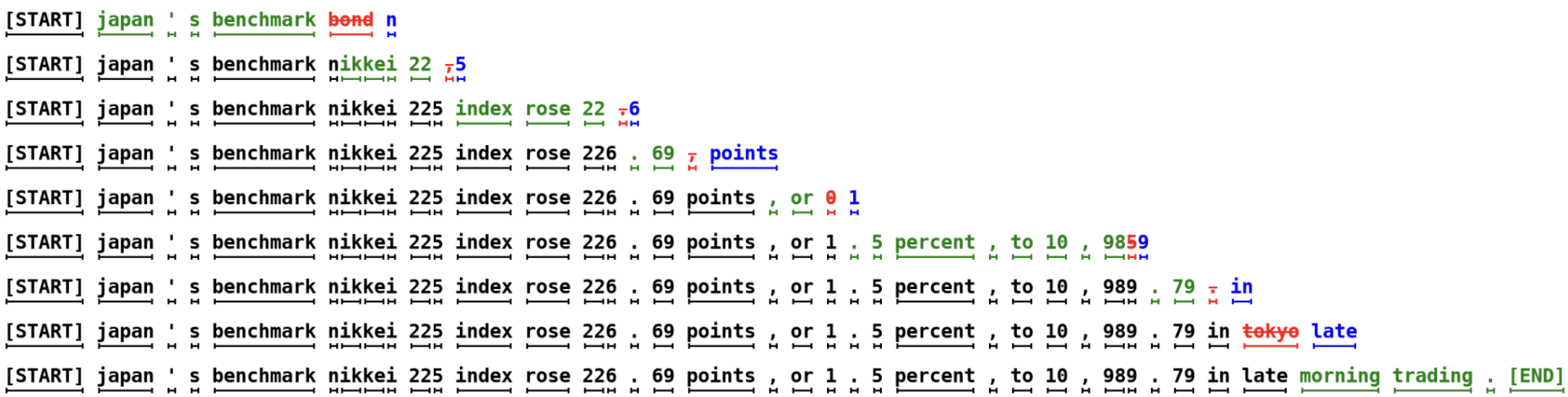
The key observation above, that some inference steps are "harder" and some are "easier", is also a key motivator for our work. We additionally observe that inference from large models is often not bottlenecked on arithmetic operations, but rather on memory bandwidth and communication, so additional computation resources might be available. There-

ICML 2023

Speculative Decoding

Key idea:

1. Use a more efficient model to generate several completions
2. Evaluate all guesses with the target model *in parallel*, *accept* the ones that lead to an identical distribution
3. Sample an additional token from an adjusted distribution to fix the first *rejected* token



Warm-up: Sample 1 Token

- Target model: $p(x_t \mid x_{<t})$, Approximate model: $q(x_t \mid x_{<t})$.
- To sample $x \sim p$:
  - Sample $x \sim q$. Accept the sample with probability $\min(1, p(x)/q(x))$.
  - If rejected, sample from $p'(x) = \text{norm}(\max(0, p(x) - q(x)))$.
  - The final sample will have distribution p.
- Similar to *rejection sampling*, but the distribution we sample from after rejection is different.

Why does it work?

Let X be a sample generated by this procedure. Note that

$$X = \mathbf{1}[R \leq p(\tilde{X})/q(\tilde{X})]\tilde{X} + \mathbf{1}[R > q(\tilde{X}/p(\tilde{X})]X',$$

where $R \sim \mathrm{Unif}(0,1), \tilde{X} \sim q, X' \sim p'$ independently. Then

Why does it work?

Let X be a sample generated by this procedure. Note that

$$X = \mathbf{1}[R \leq p(\tilde{X})/q(\tilde{X})]\tilde{X} + \mathbf{1}[R > q(\tilde{X}/p(\tilde{X})]X',$$

where $R \sim \mathrm{Unif}(0,1), \tilde{X} \sim q, X' \sim p'$ independently. Then

$$\begin{aligned}
\mathbb{P}[X = x] &= \mathbb{P}[R \leq p(x)/q(x)]q(x) + \mathbb{P}[R > p(\tilde{X})/q(\tilde{X})]p'(x) \\
&= \min(p(x)/q(x), 1)q(x) + \mathbb{E}[1 - \min(p(\tilde{X})/q(\tilde{X}))]p'(x) \\
&= \min(p(x), q(x)) + \big(1 - \sum_{\tilde{x}} \min(p(\tilde{x}), q(\tilde{x}))\big)p'(x)
\end{aligned}$$

Plugging in p' implies $\mathbb{P}[X = x] = p(x)$.

Sample More Tokens

Algorithm 1 SpeculativeDecodingStep

Inputs: $M_p, M_q, prefix$.
▷ Sample γ guesses $x_{1,\ldots,\gamma}$ from M_q autoregressively.
for $i = 1$ **to** γ **do**
  $q_i(x) \leftarrow M_q(prefix + [x_1, \ldots, x_{i-1}])$
  $x_i \sim q_i(x)$
end for
▷ Run M_p in parallel.
$p_1(x), \ldots, p_{\gamma+1}(x) \leftarrow$
  $M_p(prefix), \ldots, M_p(prefix + [x_1, \ldots, x_\gamma])$
▷ Determine the number of accepted guesses n.
$r_1 \sim U(0,1), \ldots, r_\gamma \sim U(0,1)$
$n \leftarrow \min(\{i - 1 \mid 1 \leq i \leq \gamma, r_i > \frac{p_i(x)}{q_i(x)}\} \cup \{\gamma\})$
▷ Adjust the distribution from M_p if needed.
$p'(x) \leftarrow p_{n+1}(x)$
if $n < \gamma$ **then**
  $p'(x) \leftarrow norm(max(0, p_{n+1}(x) - q_{n+1}(x)))$
end if
▷ Return one token from M_p, and n tokens from M_q.
$t \sim p'(x)$
return $prefix + [x_1, \ldots, x_n, t]$

Expected Number of Tokens in One Serial Evaluation of p

- Let $z_i = 1$ if sample i is accepted and $z_i = 0$ otherwise.
- # of generated tokens $= \mathbf{1}[z_1] + \mathbf{1}[z_1, z_2] + \ldots + \mathbf{1}[z_1, \ldots, z_\gamma]$
- Let $\alpha := \mathbb{E}[z_i] = \mathbb{P}[R_i < p(\tilde{X}_i)/q(\tilde{X}_i)] = \sum_x \min(p(x), q(x))$.
- Then

$$\mathbb{E}[\#\text{ of generated tokens}] = \frac{1 - \alpha^{\gamma+1}}{1 - \alpha}$$

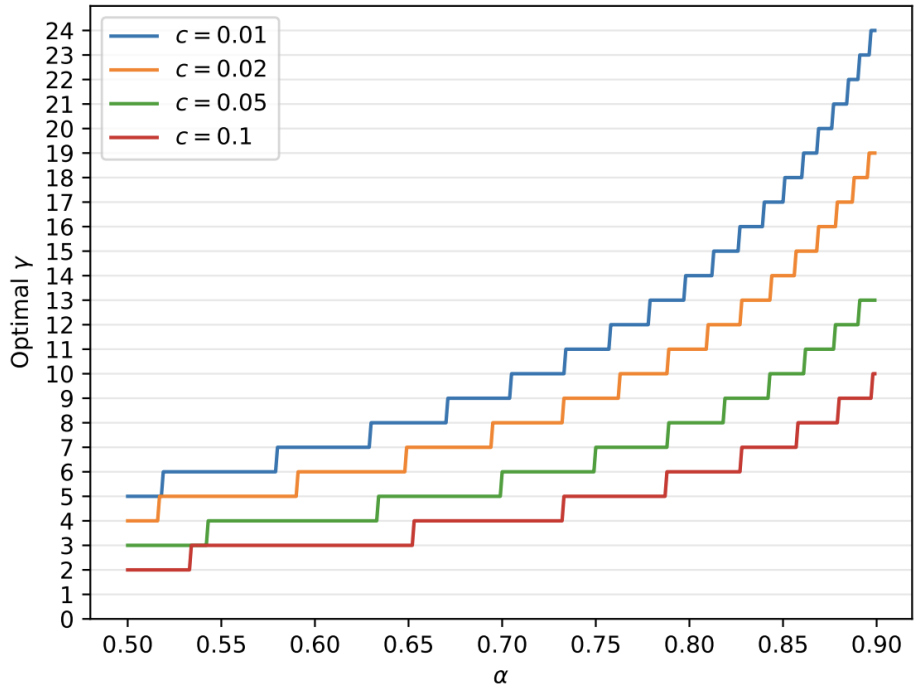
How to choose γ?

- Suppose the ratio between the time of a single run of q to p is $c \in (0, 1)$.
- The ratio between the time of one speculative decoding step and one step of p is $\gamma c + 1$.
- The improvement factor in walltime by speculative decoding is

$$\frac{\#\text{ of generated tokens}}{\text{runtime}} = \frac{1 - \alpha^{\gamma+1}}{(1 - \alpha)(\gamma c + 1)}.$$

- Optimal γ minimizes walltime.

How to choose γ?



Optimal γ as a function of α for different c

Language Diffusion Models

Large Language Diffusion Models

Shen Nie [1 * †] **Fengqi Zhu** [1 * †] **Zebin You** [1 †] **Xiaolu Zhang** [2 ‡] **Jingyang Ou** [1] **Jun Hu** [2 ‡] **Jun Zhou** [2]
Yankai Lin [1 ‡] **Ji-Rong Wen** [1] **Chongxuan Li** [1 ‡ ¶]

Abstract

Autoregressive models (ARMs) are widely regarded as the cornerstone of large language models (LLMs). We challenge this notion by introducing **LLaDA**, a diffusion model trained from scratch under the pre-training and supervised fine-tuning (SFT) paradigm. LLaDA models distributions through a forward data masking process and a reverse process, parameterized by a vanilla Transformer to predict masked tokens. By optimizing a likelihood bound, it provides a principled generative approach for probabilistic inference. Across extensive benchmarks, LLaDA demonstrates strong *scalability*, outperforming our self-constructed ARM baselines. Remarkably, LLaDA 8B is competitive with strong LLMs like LLaMA3 8B in *in-context learning* and, after SFT, exhibits impressive *instruction-following* abilities in case studies such as multi-turn dialogue. Moreover, LLaDA addresses the reversal curse, surpassing GPT-4o in a reversal poem completion task. Our findings establish diffusion models as a viable and promising alternative to ARMs, challenging the assumption that key LLM capabilities discussed above are inherently tied to ARMs. Project page and codes: https://ml-gsai.github.io/LLaDA-demo/.

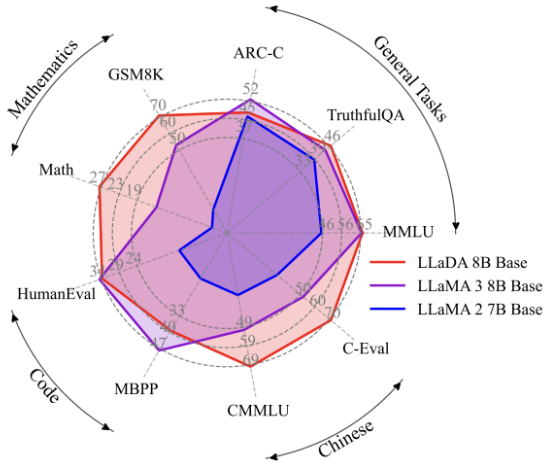
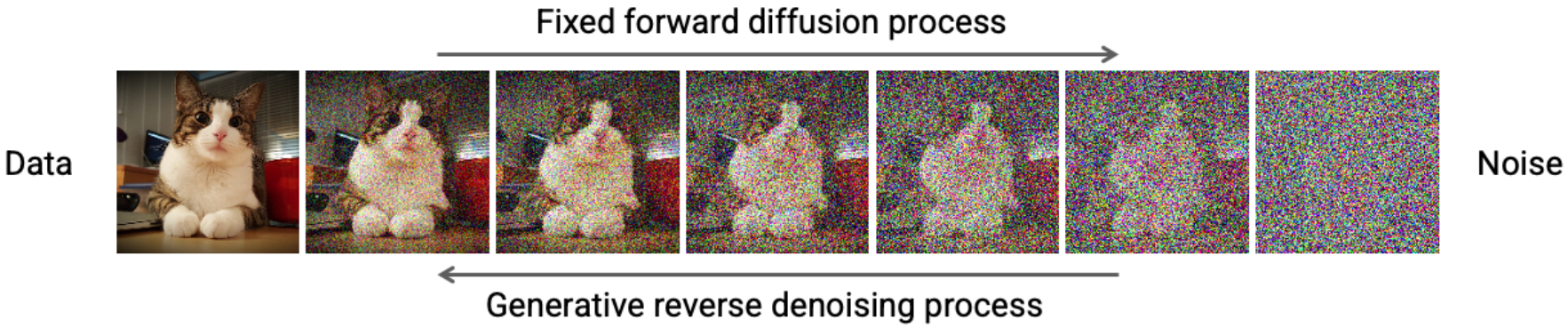


Figure 1. **Zero/Few-Shot Benchmarks.** We scale LLaDA to an unprecedented size of 8B parameters from scratch, achieving competitive performance with strong LLMs (Dubey et al., 2024).

distribution $p_{\text{data}}(\cdot)$ by optimizing a model distribution $p_\theta(\cdot)$ through maximum likelihood estimation, or equivalently KL divergence minimization between the two distributions:

$$\underbrace{\max_\theta \mathbb{E}_{p_{\text{data}}(x)} \log p_\theta(x) \Leftrightarrow \min_\theta \mathrm{KL}(p_{\text{data}}(x) || p_\theta(x))}_{\text{Generative modeling principles}}. \quad (1)$$

What Are Diffusion Models?



In diffusion models, we *learn to predict original data conditioned on noisy versions*, i.e. to perform denoising.

Language Diffusion Models

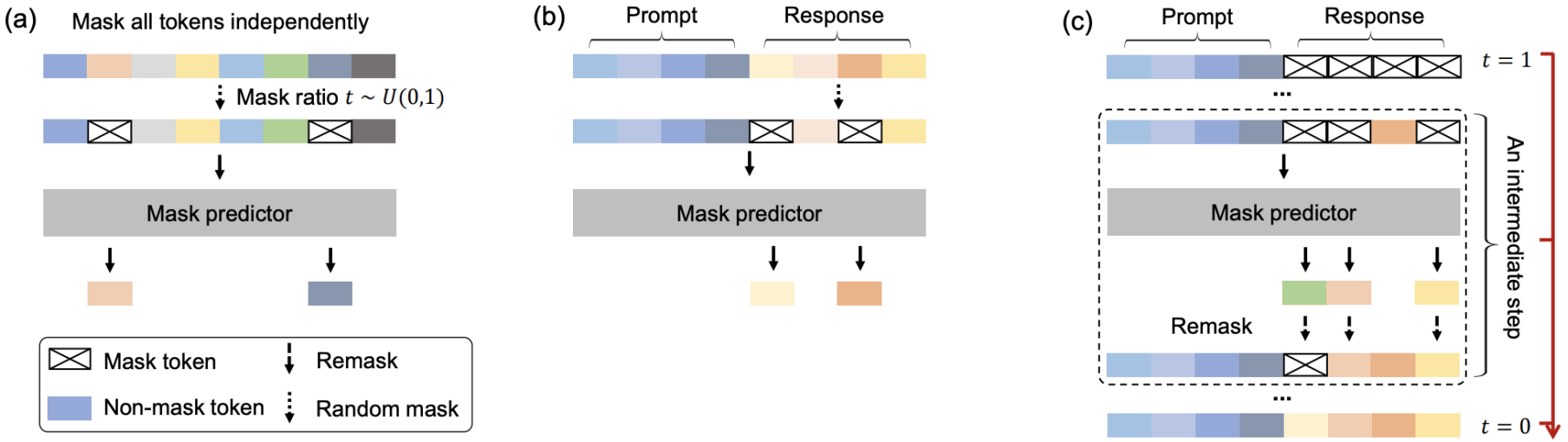


Figure 2. **A Conceptual Overview of LLaDA.** (a) Pre-training. LLaDA is trained on text with random masks applied independently to all tokens at the same ratio $t \sim U[0, 1]$. (b) SFT. Only response tokens are possibly masked. (c) Sampling. LLaDA simulates a diffusion process from $t = 1$ (fully masked) to $t = 0$ (unmasked), predicting all masks simultaneously at each step with flexible remask strategies.

From the LLaDA paper

LLaDA

- LLaDA replaces the noise with *masks*.
- The forward process gradually masks tokens from $t = 0$ to $t = 1$, x_0 is original data, x_1 is all masked.
- The core idea is to train a mask predictor that predicts *all masked tokens simulatenously*:
  - Minimize the loss

$$\mathcal{L}(\theta) = -\mathbb{E}_{t,x_0,x_t}\left[\frac{1}{t}\sum_{i=1}^{L}\mathbf{1}[x_t^i = M]\log p_\theta(x_0^i \mid x_t)\right]$$

  - The loss is only calculated on masked tokens.

Pretraining, Finetuning, and Inference

- Pretraining loss:

$$-\mathbb{E}_{t,x_0,x_t}\left[\frac{1}{t}\sum_{i=1}^{L}\mathbf{1}[x_t^i = M]\log p_\theta(x_0^i \mid x_t)\right]$$

- Supervised Finetuning loss:

$$-\mathbb{E}_{t,p_0,r_0,r_t}\left[\frac{1}{t}\sum_{i=1}^{L}\mathbf{1}[r_t^i = M]\log p_\theta(r_0^i \mid p_0, r_t)\right]$$

- Likelihood evaluation:

$$-\mathbb{E}_{l,r_0,r_l}\left[\frac{1}{l}\sum_{i=1}^{L}\mathbf{1}[r_l^i = M]\log p_\theta(r_0^i \mid p_0, r_l)\right]$$

where l is randomly sampled from $\{1, \ldots, L\}$.

Performance: LLaDA vs. Autoregressive (Pretrained) Models

	LLaDA 8B*	LLaMA3 8B*	LLaMA2 7B*	Qwen2 7B†	Qwen2.5 7B†	Mistral 7B†	Deepseek 7B¶
Model	Diffusion	AR	AR	AR	AR	AR	AR
Training tokens	2.3T	15T	2T	7T	18T	-	2T
General Tasks							
MMLU	**65.9** (5)	65.4 (5)	45.9 (5)	70.3 (5)	74.2 (5)	64.2 (5)	48.2 (5)
BBH	49.8 (3)	**57.6** (3)	37.3 (3)	62.3 (3)	70.4 (3)	56.1 (3)	39.5 (3)
ARC-C	47.9 (0)	**53.1** (0)	46.3 (0)	60.6 (25)	63.7 (25)	60.0 (25)	48.1 (0)
Hellaswag	72.5 (0)	**79.1** (0)	76.0 (0)	80.7 (10)	80.2 (10)	83.3 (10)	75.4 (0)
TruthfulQA	**46.4** (0)	44.0 (0)	39.0 (0)	54.2 (0)	56.4 (0)	42.2 (0)	-
WinoGrande	74.8 (5)	**77.3** (5)	72.5 (5)	77.0 (5)	75.9 (5)	78.4 (5)	70.5 (0)
PIQA	74.4 (0)	**80.6** (0)	79.1 (0)	-	-	-	79.2 (0)
Mathematics & Science							
GSM8K	**70.7** (4)	53.1 (4)	14.3 (4)	80.2 (4)	85.4 (4)	36.2 (4)	17.4 (8)
Math	**27.3** (4)	15.1 (4)	3.2 (4)	43.5 (4)	49.8 (4)	10.2 (4)	6.0 (4)
GPQA	**26.1** (5)	25.9 (5)	25.7 (5)	30.8 (5)	36.4 (5)	24.7 (5)	-
Code							
HumanEval	33.5 (0)	**34.2** (0)	12.8 (0)	51.2 (0)	57.9 (0)	29.3 (0)	26.2 (0)
HumanEval-FIM	**73.8** (2)	73.3 (2)	26.9 (2)	-	-	-	-
MBPP	38.2 (4)	**47.4** (4)	18.4 (4)	64.2 (0)	74.9 (0)	51.1 (0)	39.0 (3)
Chinese							
CMMLU	**69.9** (5)	50.7 (5)	32.5 (5)	83.9 (5)	-	-	47.2 (5)
C-Eval	**70.5** (5)	51.7 (5)	34.0 (5)	83.2 (5)	-	-	45.0 (5)

Performance: LLaDA vs. Autoregressive (Finetuned) Models

	LLaDA 8B*	LLaMA3 8B*	LLaMA2 7B*	Qwen2 7B†	Qwen2.5 7B†	Gemma2 9B†	Deepseek 7B¶
Model	Diffusion	AR	AR	AR	AR	AR	AR
Training tokens	2.3T	15T	2T	7T	18T	8T	2T
Post-training	SFT	SFT+RL	SFT+RL	SFT+RL	SFT+RL	SFT+RL	SFT+RL
Alignment pairs	4.5M	-	-	0.5M + -	1M + 0.15M	-	1.5M + -
General Tasks							
MMLU	65.5 (5)	**68.4** (5)	44.1 (5)	-	-	-	49.4 (0)
MMLU-pro	37.0 (0)	**41.9** (0)	4.6 (0)	44.1 (5)	56.3 (5)	52.1 (5)	-
Hellaswag	74.6 (0)	**75.5** (0)	51.5 (0)	-	-	-	68.5 (-)
ARC-C	**88.5** (0)	82.4 (0)	57.3 (0)	-	-	-	49.4 (-)
Mathematics & Science							
GSM8K	**78.6** (4)	78.3 (4)	29.0 (4)	85.7 (0)	91.6 (0)	76.7 (0)	63.0 (0)
Math	26.6 (0)	**29.6** (0)	3.8 (0)	52.9 (0)	75.5 (0)	44.3 (0)	15.8 (0)
GPQA	31.8 (5)	**31.9** (5)	28.4 (5)	34.3 (0)	36.4 (0)	32.8 (0)	-
Code							
HumanEval	47.6 (0)	**59.8** (0)	16.5 (0)	79.9 (0)	84.8 (0)	68.9 (0)	48.2 (-)
MBPP	34.2 (4)	**57.6** (4)	20.6 (4)	67.2 (0)	79.2 (0)	74.9 (0)	35.2 (-)