A Dataset Details

A.1 Dataset on Arithmetic Tasks

Pre-training Dataset. The training dataset for pre-training arithmetic model is created with a Python script. The dataset includes a variety of arithmetic expressions, encompassing different types of arithmetic operations such as addition, subtraction, multiplication, division, and exponentiation. Each expression in the dataset is composed of various types of numbers, including integers, decimals, fractions, percents, and negative numbers. The training dataset consists of approximately 50 million arithmetic sequences. To investigate the impact of dataset scale on the arithmetic performance, we also create multiple datasets of varying sizes, including 1 million, 5 million, 10 million, and 25 million. This diverse representation of numbers ensures that the model can handle a wide range of numerical formats encountered in real-world arithmetic problems.

To facilitate the learning of underlying calculation rules, the arithmetic expressions are designed to be more complex than simple two-number calculations. Instead, each expression in the dataset involves multiple steps of calculations, ranging from 2 to 10 steps. By creating multi-step expressions, the model is exposed to more intricate mathematical reasoning and is better equipped to handle complex arithmetic problem-solving. The details of expressions is presented as follows. Table 1 demonstrates examples from the arithmetic dataset.

- Operations involving integers up to 10,000 that combine addition, subtraction, multiplication, and division.
- Exponentiation tasks using an integer base up to 10,000 and an integer exponent capped at 100.
- Bracketed expressions that include integers up to 10,000, combined with operations such as addition, subtraction, multiplication, and division.
- Lengthy arithmetic expressions that incorporate brackets and blend various numerical types, including integers, decimals, percentages, and negative numbers. These sequences utilize operations such as addition, subtraction, multiplication, and division.
- Arithmetic expressions involving fractions combined with various operations, including addition, subtraction, multiplication, and division.

Validation Dataset. Our evaluation dataset, which comprises 9,592 test cases, is generated from the same distribution as the training dataset, yet remains distinct and is excluded from the training process. This carefully generated suite of datasets serves as a comprehensive benchmark to evaluate and quantify MathGLM's computational prowess across a wide variety of arithmetic tasks.

A.2 Validation Dataset on MWP

In the field of math word problems (MWP), the performance of MathGLM is measured using the Ape210K test dataset (Zhao et al., 2020), which contains a collection of 5,000 test math problems. Additionally, we introduce the K6 dataset, which is designed to cover math word problems suitable for elementary school students across 6 different grade levels. The primary purpose of the K6 dataset is to assess the mathematical abilities of LLMs in comprehending and solving general-purpose math reasoning problems. By evaluating MathGLM on the K6 dataset, we are able to gauge its effectiveness in handling mathematical word problems of varying complexity and across a range of grade levels. We collect math word problems from Chinese elementary schools in collaboration with the renowned educational institution, TAL AI Lab. The dataset consists of math problems for each grade level, with each grade containing approximately 100 problems. The wide-ranging nature of these math word problems empowers us to gauge the model's efficacy across an array of difficulty gradients and academic grades. To illustrate the diversity and complexity of the K6 dataset, we present some exemplary math word problems in Table 2. These examples show the range of mathematical concepts covered and the varying levels of difficulty present in the dataset.

Table 1: Examples from the arithmetic dataset where “+”, “-”, “*”, “/”, “^” denotes addition, subtraction, multiplication, division, and exponentiation respectively.

Types	Arithmetic Expression
Integre mixing operation	1+8/1*10+2=1+8*10+2=1+80+2=81+2=83
	53-2+23+51*56=53-2+23+2856=51+23+2856=74+2856=2930
	214-792*509*260*556=214-403128*260*556=214-104813280*556=214-58276183680=-58276183466
	1912*6800*6022-7250-1624=13001600*6022-7250-1624=78295635200-7250-1624=78295627950-1624=78295626326
Exponentiation	5170^0=1, 1^8756=1
	3^9=19683, 93^18=270827695297250208363869180422467849
	100^13=1000000000000000000000000000
Expression of fractions	((49/24)*-(8/70))/-(34/80)=(+(49/24)*(8/70))/(34/80)=(392/1680)/(34/80)=
	(7/30)/(34/80)=(7/30)*(80/34)=(560/1020)=28/51
	(9947/9276)+(4411/9276)=14358/9276=2393/1546
Expression with brackets	-7805+(4383/7377)=-7805+0.5941439609597398=-7804.40585603904
	8371*(-1945+8878)=8371*(-1945+8878)=8371*6933=58036143
Lengthy arithmetic expressions	(-2090-5457.35697)*73.0=-7547.35697*73.0=-550957.05881
	-4457+(-7823/5483%)*-3338=-4457+(-7823/54.83)*-3338=-4457+(-142.6773664052526)*-3338=-4457+-142.6773664052526*-3338=-4457+142.6773664052526*3338=-4457+476257.0490607332=471800.0490607332

Grade	Example
K1	李老师买了20颗糖果,送给小丽5颗,送给小刚8颗,还剩多少颗糖果?
K2	一个乘数是4,另一个乘数是7,积是多少?
K3	乐乐家养了36只小鸡,其中1/4是公鸡,母鸡是公鸡的3倍,公鸡和母鸡各有多少只?
K4	公益小组的同学为敬老院的老人们制作香囊(náng),12个组共制作了864个,每组都有9人,平均每人制作了几个?
K5	东、西两城相距180千米,甲、乙两车分别从东、西两城同时出发,相向而行,1.2小时后两车可相遇.实际甲车出发0.4小时后因故障停车,乙车又走了2小时才和甲车相遇,求乙车每小时行多少千米?
K6	甜甜读一本小说,第一天读了这本书的3/8,正好是180页,第二天又读了这本书的1/6,第2天读了多少页?

Table 2: Examples from the K6 dataset to demonstrate the diversity and complexity of this dataset.

B Training Details

B.1 Overview of MathGLM on Arithmetic Tasks

Training Parameters for MathGLM. Table 3 reports an overview of all the models with different model parameters, including hidden dimensions, the number of attention heads, and the total number of layers employed in the model. Besides, we offer detailed training steps to facilitate the reproduction of our MathGLM.

Table 3: Model sizes and architectures of MathGLM.

Model	Dimension	Heads	Layers	Parameters	Training Steps
MathGLM-10M	256	32	15	10M	120,000
MathGLM-100M	512	32	35	100M	155,000
MathGLM-500M	1024	32	40	500M	135,000
MathGLM-2B	2048	32	40	2B	155,000

Tokenization for Arithmetic Tasks. The arithmetic operations in our MathGLM involve numbers from 0 to 9, and the calculating signs comprise addition (+), subtraction (-), multiplication (*), division (/), and exponentiation (^). Symbols that represent forms in the data include the decimal point (.), percent sign (%), negative sign (-), fraction delimiter (/), brackets such as '(' and '[', and the equal sign (=). To achieve a consistent tokenization process, we adopt the unified tokenization tool *icetk* proposed in CogView2 (Ding et al., 2022). By leveraging this methodology, we tokenize each digit as a distinct token. For instance, the numeral "12345" is tokenized into the set $\{1, 2, 3, 4, 5\}$. To allocate singular tokens to the other mentioned symbols, we disengage the continuous representation symbols within icetk throughout the tokenization procedure.

Table 4 shows some tokenization examples employed in MathGLM. This tokenization approach ensuers that every element in the arithmetic expression is adequately represented and can be efficiently processed by the MathGLM, facilitating MathGLM to excute comprehensive arithmetic tasks. Owing to the variable lengths of arithmetic expressions, it becomes imperative to standardize their lengths for efficient training of the MathGLM. A straightforward method, like padding each input to a fixed length, might damage training efficacy. To circumvent this, we adopt a more efficient strategy, where multiple arithmetic expressions are concatenated until they achieve a predefined fixed length.

Table 4: Some examples of tokenization in MathGLM.

Input	Tokenization
12345+345=	['_', '1', '2', '3', '4', '5', '+', '3', '4', '5', '='] [20005, 20009, 20010, 20013, 20016, 20015, 20065, 20013, 20016, 20015, 20054]
1234-45678=	['_', '1', '2', '3', '4', '-', '4', '5', '6', '7', '8', '='] [20005, 20009, 20010, 20013, 20016, 20011, 20016, 20015, 20021, 20025, 20023, 20054]
34*678=	['_', '3', '4', '*', '6', '7', '8', '='] [20005, 20013, 20016, 20032, 20021, 20025, 20023, 20054]
1.2/2=	['_', '1', '.', '2', '/', '2', '='] [20005, 20009, 20007, 20010, 20026, 20010, 20054]
(1.2*3%)/2+[(12+3)*5]=	['_', '(', '1', '.', '2', '*', '3', '%', ')', '/', '2', '+', '[', '(', '1', '2', '+', '3', ')', '*', '5', ']', '='] [20005, 20020, 20009, 20007, 20010, 20032, 20013, 20040, 20014, 20026, 20010, 20065, 20052, 20020, 20009, 20010, 20065, 20013, 20014, 20032, 20015, 20042, 20054]

B.2 BACKBONE MODELS

General Language Model (GLM) is a Transformer-based language model that combines autogressive blank infilling with bidirectional attention mechanisms. Different from decoder-only language models that primarily rely on unidirectional attention, GLM integrates bidirectional attention on unmasked contexts. This innovative approach empowers it with heightened proficiency in both comprehension and generative tasks.

Pre-Training Objectives. To amplify its linguistic understanding and generative abilities, GLM incorporates a dual pre-training strategy: 1) *Autoregressive Blank Infilling* involves predicting missing tokens within spans of corrupted text, wherein segments are arbitrarily supplanted with a [MASK] token. 2) *Multi-Task Pretraining* is utilized to endow GLM text generation ability, which aims to generate longer text by sampling random-length span from document-level or sentence-level text.

Model Sizes. GLM offers a diverse of models with various model parameters, including GLM-Large, GLM-6B, GLM-10B, GLM2-6B, ChatGLM-6B, and ChatGLM2-6B. Comprehensive specifics concerning the hyperparameters for each model variant can be found in Table 5. GLM-Large model is specifically tailored for Chinese language processing tasks equipped with 335M model parameters, while GLM-10B, GLM-6B, and GLM2-6B are equipped with 10 billion, 6 billion, and 6 billion parameters, respectively, enabling them to handle a wide range of NLP tasks with varying complexities. Augmenting the series are bilingual conversational models: ChatGLM-6B and ChatGLM2-6B, both tailored for Chinese-English bilingual dialogue tasks. The ChatGLM-6B model, having 6.2 billion parameters, undergoes fine-tuning using Chinese Q&A and dialogue datasets. In contrast, ChatGLM2-6B emerges as an evolved iteration of ChatGLM-6B, marking enhancements in performance, extended context handling, optimized inference, and broader applicability.

Table 5: Hyperparameters of the backbone models.

Model	Dimension	Heads	Layers	Parameters
GLM-Large	1024	24	16	335M
GLM-10B	4096	64	48	10B
GLM-6B	4096	32	28	6.2B
GLM2-6B	4096	32	28	6.2B
ChatGLM-6B	4096	32	28	6.2B
ChatGLM2-6B	4096	32	28	6.2B

C EVALUATION METRIC

To measure the ability of MathGLM on arithmetic tasks, we adopt the following metrics to evaluate the outputs.

Accuracy is typically measured by comparing the output of the MathGLM and the ground truth answer. In our experiments, we adhere to standard rounding rules, constraining the generated answers to precisely two decimal places. When the correctly rounded answer aligns with the answer generated by the MathGLM, we classify this outcome as a correct answer.

Relative Error is another important metric used to evaluate the effectiveness of MathGLM, which quantifies the difference between the output generated by MathGLM and the correct answer. The relative error (RE) is quantified using the following formula:

$$RE = |\frac{\hat{y} - y}{y}| \tag{1}$$

where $\hat{y}$ and y denote the generated answer and the correct answer respectively. For our evaluation purposes, we utilize a relative error threshold of 1%. This threshold serves as a criterion for determining the acceptability of the answers generated by the MathGLM, where any relative error falling within this threshold range is considered an accurate outcome.

D Additional Exeriments on Arithmetic Tasks

D.1 Results on Test-100

Additionally, we conduct a performance comparison of arithmetic tasks among different prominent large language models (LLMs) including GPT-4, ChatGPT, text-davinci-003, code-davinci-002, Galactica, LLaMA, OPT, BLOOM, and GLM. For this comparison, we randomly extract a compact arithmetic dataset *Test-100* containing 100 test cases from the larger dataset discussed earlier. The results of this comparison arithmetic performance are presented in Table 6. Upon analyzing the results, it is evident that MathGLM achieves a high accuracy of 93.03% with 2 billion model parameters, surpassing all other LLMs. In addition to leading models like GPT-4 and ChatGPT, the large science model Galactica exhibits better performance in arithmetic tasks. This can be attributed to Galactica's training on a large scientific corpus, enabling it to learn the languages of science and comprehend the intricacies of arithmetic tasks. By leveraging the unique characteristics of this dataset, Galactica is able to enhance its understanding and handling of arithmetic tasks, resulting in improved performance. These findings emphasize the significance of domain-specific training and leveraging specialized datasets to enhance model performance. Besides, a step-by-step solution strategy, which involves decomposing complex arithmetic expressions into individual steps, has proven to be effective in improving arithmetic performance. The outstanding performance of MathGLM shows that the language model coupled with a specialized dataset and the step-by-step solution strategy can achieve remarkable performance in arithmetic tasks.

Table 6: Overall performance comparison on various LLMs in term of Accuracy.

Model	ACC	RE
GPT-4	22.22%	-
ChatGPT	13.25%	-
text-davinci-003	9.79%	-
text-davinci-002	4.08%	-
Galactica-120b	7.97%	-
Galactica-30b	7.02%	-
LLaMA-65b	5.02%	-
OPT-175B	3.83%	-
BLOOM-176B	3.96%	-
GLM-130B	3.06%	-
MathGLM-10M	64.29%	97.96%
MathGLM-100M	73.47%	98.23%
MathGLM-500M	89.80%	98.82%
MathGLM-2B	94.90%	98.98%

D.2 Grouped Results

To clearly evaluate the arithmetic ability of MathGLM among different operations, we design a series of extended experiments. Specifically, we design small test datasets comprising 100 test cases to respectively evaluate the arithmetica performance of MathGLM in various arithmetic operations, including addition, subtraction, multiplication, and division. These datasets encompass different data formats, such as integers, decimals, percents, fractions and negative numbers. Here, we compare MathGLM with several well-known chat-type LLMs, such as GPT-4, ChatGPT, ChatGLM, and Bard. The arithmetic performance comparison among these different language models is demonstrated in Table 7. Analyzing the results, we can observe that the majority of LLMs exhibit commendable accuracy levels exceeding 90% across diverse data formats for elementary arithmetic operations like addition and subtraction. However, as the complexity escalates to operations like multiplication and division, a divergence in performance manifests across different models. For instance, the accuracy levels of the most powerful model GPT-4 also show a trend towards zero, especially when dealing with decimal and percentile data formats. In contrast, MathGLM consistently shows superior performance in multiplication operations across various data formats, surpassing the capability of GPT-4. This demonstrates the effectiveness and capabilities of MathGLM in handling complex arithmetic tasks, even outperforming a prominent model like GPT-4 in specific operations. Notably, even the smaller variant of MathGLM, MathGLM-10M, with only 10 million training parameters, also achieves remarkable arithmetic performances, further emphasizing the arithmetic capabilities of our MathGLM.

D.3 Results on BIG-bench

We also evaluate MathGLM using BIG-bench arithmetic dataset (Srivastava et al., 2022), which is commonly used to evaluate basic arithmetic capabilities of language models by performing n-digit addition (ADD), subtraction (SUB), multiplication (MUL), and division (DIV). Table 8 reports

Task	Format	GPT-4	ChatGPT	ChatGLM	Bard	MathGLM-10M	MathGLM-2B
ADD	Int	100%	100%	94%	96.0%	100%	100%
	Dec	100%	98%	76%	87%	96%	100%
	Frac	43.33%	17.02%	32.98%	14.2%	60.64%	100%
	Perc	100%	90.0%	1%	9.6%	100%	100%
	Neg	100%	98%	91%	95%	100%	100%
SUB	Int	100%	97%	89%	91%	98%	100 %
	Dec	100%	94%	82%	85%	98%	100%
	Frac	52.48%	18.81%	3%	24.24%	68.32%	96.04%
	Perc	100%	100%	18%	0%	99%	100%
	Neg	100%	97%	44%	78%	100%	100%
MUL	Int	9%	4%	1%	2%	77%	84%
	Dec	0%	0%	0%	0%	3%	33%
	Frac	5.63%	2.82%	1.41%	1.41%	67.61%	85.92%
	Perc	0%	0%	1%	0%	81%	97%
	Neg	7%	2%	0%	0%	76%	98%
DIV	Int	92%	91%	24%	68%	99%	100%
	Dec	93%	88%	60%	60%	97%	98%
	Frac	33.44%	29.69%	7.81%	1.56%	73.44%	96.88%
	Perc	97%	80%	19%	15%	88%	100%
	Neg	97%	90%	50%	52%	96%	100%

Table 7: Arithmetic comparison between MathGLM and other LLMs among different operations. Int denotes integers, Dec denotes decimals, Frac denotes fractions, Perc denotes percents, and Neg denotes negative numbers.

the experimental results of GPT-4 and MathGLM on various arithmetic operations with different numbers of digits. GPT-4 exhibits near-perfect (100%) accuracy in low-digit arithmetic tasks. However, as the digits escalate, the performance gradually diminishes, particularly pronounced in the multiplication task. In contrast, MathGLM consistently maintains high accuracy levels even in high-digit arithmetic tasks, illustrating its outstanding ability to handle complex arithmetic tasks effectively. The performance trends of different MathGLM variants reveal a consistent pattern of improvement as model size increases. For ADD and SUB tasks, the accuracy remains consistently high across all model sizes with slight variations. There is a tendency for larger models to achieve higher accuracy compared to smaller models but the differences in performance between different model sizes are relatively small. In the MUL task, accuracy rises distinctly with larger model sizes. Smaller models exhibit relatively lower accuracy, while larger counterparts demonstrate enhanced accuracy, particularly in tasks involving higher digit numbers. A similar tendency can be observed in the DIV task. Overall, the evaluation results demonstrate that MathGLM outperforms GPT-4 in high-digit arithmetic tasks, and the performance generally inproves with larger model sizes.

D.4 Results on MATH 401

Table 9 shows a comprehensive evaluation of the arithmetic performance of MathGLM on the MATH 401 dataset (Yuan et al., 2023). This dataset offers a new set of arithmetic problems, allowing for a deeper exploration into MathGLM's proficiency in addressing a wide variety of arithmetic tasks. By evaluating MathGLM's performance on this dataset, we observe that MathGLM consistently outperforms all other large language models with a substantial number of model parameters.

D.5 Analysis on Arithmetic Errors

Despite achieving an impressive overall accuracy of 93.03% with its 2 billion model parameters, a thorough analysis is conducted to comprehend instances where MathGLM fails to generate accurate answers. Consider the example $3468 * 4046/7424$, MathGLM generate an answer of $468 * 4046/7424 = 14031528/7424 = 1889.901400862069$, while the true answer is

Table 8: Overall performance comparison on GPT-4 and MathGLM on BIG-bench Arithmetic sub-task.

Task		GPT-4	MathGLM-10M	MathGLM-100M	MathGLM-500M	MathGLM-2B
ADD	1D	100%	84%	100%	100%	100%
	2D	100%	97.2%	100%	100%	100%
	3D	99.6%	99.3%	100%	100%	100%
	4D	98.8%	99.9%	99.9%	100%	100%
	5D	94.1%	99.2%	100%	99.6%	99.4%
SUB	1D	100%	92%	100%	100%	100%
	2D	100%	98.5%	99.8%	100%	100%
	3D	99.2%	98.8%	99.9%	100%	99.9%
	4D	98.9%	98.4%	99.6%	99.7%	99.8%
	5D	92.4%	98.0%	99.3%	99.5%	98.9%
MUL	1D	100%	91%	100%	99%	100%
	2D	99.4%	85.8%	99.7%	99.9%	99.9%
	3D	30.3%	77.8%	91.4%	93.7%	98.3%
	4D	5.3%	79.7%	80.4%	90.0%	94.9%
	5D	0.0%	41.6%	55.6%	59.6%	89.9%
DIV	1D	100%	87.0%	100%	100%	100%
	2D	100%	89.5%	100%	100%	100%
	3D	94.5%	90.2%	100%	99.6%	99.4%
	4D	90.9%	90.5%	99.5%	99.6%	100%
	5D	53.4%	82.2%	92.9%	93.6%	94.9%

$468 * 4046/7424 = 14031528/7424 = 1890.0226293103$. Upon comparing the generated results with the true answers, it is obviously observed that the multiplication operation for $468 * 4046$ is correct but the division operation for $14031528/7424$ is incorrect. One possible reason for this discrepancy is that MathGLM's pre-training primarily encompasses numbers in the 5-digit range, thereby causing inaccuracies when tackling division tasks involving 12-digit and 4-digit numbers. Upon thorough analysis of the errors made by MathGLM, it's important to highlight that the inaccuracies in the generated answers are remarkably close to the correct evaluations.

Table 10 provides some examples to analyze the failures of MathGLM on performing arithmetic tasks. Through careful examination of these examples, we can observe several patterns and trends in the MathGLM's errors. Firstly, MathGLM appears to grapple with intricate arithmetic expressions, particularly those combining several operations and large numbers. For instance, the expression 14031528/742: the division of an 8-digit number by a 4-digit one proves problematic for MathGLM, leading to miscalculations in the outcome. Secondly, MathGLM tends to encounter difficulties when dealing with long sequences of numbers and operations. As the expression length increases, the model's ability to accurately perform arithmetic calculations diminishes, leading to inaccurate results. For example, expression involving multiplication among two large numbers like 3626 * 8919 and calculation with a decimal and large integer number like 1.610311 * 7691. These errors generated by MathGLM usually have only one calculation result error, indicating that the MathGLM's mistakes mainly occur at specific calculation steps rather than affecting the entire expression.

D.6 Step-by-step Analysis

To delve deeper into the impact of the step-by-step strategy on MathGLM, we conduct extended experiments that directly calculate the answer of each arithmetic expression without employing the step-by-step approach. Figure 1 shows performance comparison between employing the step-by-step strategy and bypassing it for different models. We can observe that a significant improvement in the peformance of MathGLM when the step-by-step strategy is applied. For instance, in the case of MathGLM-500M, the accuracy rises from 31.96% to 89.57%, while for MathGLM-2B, it increases

Table 9: Overall performance comparison on various LLMs in term of Accuracy.

Model	ACC
GPT-4	83.54%
GPT-3.5-turbo	75.06%
text-davinci-003	56.61%
text-davinci-002	42.89%
code-davinci-002	21.70%
Galactica-120b	45.14%
Galactica-30b	45.14%
Galactica-6.7b	34.41%
LLaMA-65b	28.43%
LLaMA-30b	30.17%
LLaMA-13b	27.68%
LLaMA-7b	21.96%
OPT-175B	21.70%
OPT-66B	20.70%
OPT-30B	15.96%
OPT-13B	15.21%
OPT-6.7B	14.46%
BLOOM-176B	22.44%
BLOOM-7.1B	7.23%
BLOOM-3B	4.24%
BLOOM-1.7B	5.24%
GLM-130B	25.94%
GLM-10B	14.96%
MathGLM-0.5B	85.48%
MathGLM-2B	89.44%

Table 10: Some failed examples generated by MathGLM.

Input	Output	Ground Truth
3468*4046/7424=	14031528/7424=1889.901400862069	14031528/7424=1890.0226293103449
(3626*8919)/8861=	32330294/8861=3648.605574991536	32340294/8861=3649.7341157882856
7715/4791*7691-1968*9155=	1.610311*7691-1968*9155=12384.801801-1968*9155=12384.801801-18017040=-18004655.198199	1.610311*7691-1968*9155=12384.9018993-1968*9155=12384.9018993-18017040=-18004655.098100606
(4059+7011.8718)-4038.22*847.15907=	(4059+7011.8718)-4038.22*847.15907=11070.8718-4038.22*847.15907=11070.8718-3420014.6996554=-3408943.8278554	(4059+7011.8718)-4038.22*847.15907=11070.8718-4038.22*847.15907=11070.8718-3421014.6996554=-3409943.8278554003
7499-5747.91007/-5438*-439=	7499-5747.91007/5438*439=7499-1.057008104082383 2*439=7499-464.0265576921662=7034.973442307834	7499-5747.91007/5438*439=7499-1.056989715*439=7499-464.0184848713=7034.981515128724
3868*6735*5755+3741-7533=	26050980*5755+3741-7533=159923389900+3741-7533=159923393641-7533=159923386108	26050980*5755+3741-7533=149923389900+3741-7533=149923393641-7533=149923386108

from 40.76% to 93.03% for MathGLM-2B, all attributable to the incorporation of the step-by-step strategy. Similarly, the relative error accuracy exhibits a similar positive trend, escalating from 89.29% to an exceptional 99.41% for MathGLM-500M, and from 94.26% to an outstanding 99.71% for MathGLM-2B with the implementation of the step-by-step strategy. These results demonstrate the effectiveness of the step-by-step strategy in enhancing MathGLM's ability to accurately perform arithmetic operations. The step-by-step approach enables MathGLM to better understand and solve intricate arithmetic tasks, leading to significant improvements in accuracy and relative error accuracy metrics.

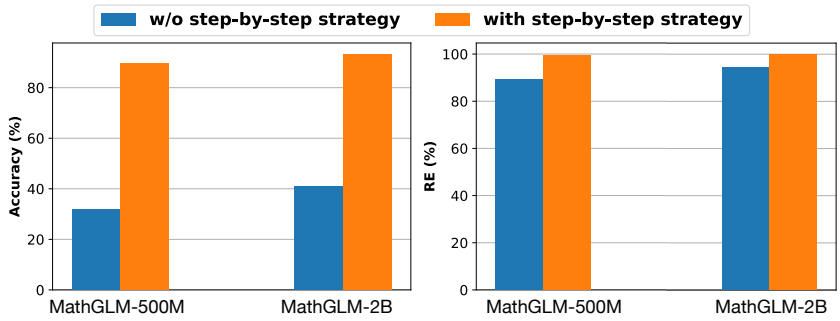

Figure 1: Performance comparison of MathGLM with and without the step-by-step solution.

D.7 Case Study

Figure 2 demonstrates examples generated by MathGLM with 2B model parameters on addition, subtraction, multiplication, division, exponentiation, and mixing operations tasks.

MathGLM-2B

Addition	Subtraction	Multiplication
5678+9865= 5678+9865=15543	**76869-98654=** 76869-98654=-21785	**1234*7809=** 1234*7809=9636306
67548+908653= 67548+908653=976201	**46455762897-9867553=** 46455762897-9867553=46445895344	**8757755*907644=** 8757755*907644=7948923779220
8659054861+7564369655= 8659054861+7564369655=16223424516	**879800.75432-767.427934=** 879800.75432-767.427934=879033.326386	**9765.89755*42653.9773065=** 9765.89755*42653.9773065=416554372.4753039
7878.547+98065.344901= 7878.547+98065.344901=105943.891901	**-7634-8904=** -7634-8904=-16538	**-453*75479=** -453*75479=-34191987
176.67+2.3%= 176.67+2.3%=176.67+0.23=176.9	**5423%-638.0965=** 5423%-638.0965=54.23-638.0965=-583.8665	**7645%*975301=** 7645%*975301=76.45*975301=74561761.45
(13/14)+(18/19)= (13/14)+(18/19)=247/266+252/266=499/266	**(3/5)-(12/19)=** (3/5)-(12/19)=57/95-60/95=-3/95	**(123/897)*(6753/16)=** (123/897)*(6753/16)=(830619/14352)=276873/4784
-32+7854= -32+7854=7822	**21-(19/21)=** 21-(19/21)=21-0.9047619047619048=20.095238095238	**875*(646/67)=** 875*(646/67)=875*9.64179104477612=8436.567164179

Division	Exponentiation	Mixing Operations
7862/87= 7862/87=90.36781609195402	**756^0=** 756^0=1	**61+9104+6129-8755-9978*1335*2454-5404=** 61+9104+6129-8755-9978*1335*2454-5404 =61+9104+6129-8755-13320630*2454-5404 =61+9104+6129-8755-32688826020-5404 =9165+6129-8755-32688826020-5404 =15294-8755-32688826020-5404 =6539-32688826020-5404 =-32688819481-5404 =-32688824885
6574389173/86532= 6574389173/86532=75976.3922363981	**457^2=** 457^2=208849	
8753.75255/5427.67397= 8753.75255/5427.67397=1.6128000040872018	**786^1=** 786^1=786	
456%/243= 456%/243=4.56/243=0.01876543209876543	**12^4=** 12^4=20736	**8723%/-632+-169.63=** 8723%/-632+-169.63=87.23/-632+-169.63 =-87.23/632-169.63=-0.13802215189873418-169.63 =-169.76802215189873
(12/33)/(56/12)= (12/33)/(56/12)=(12/33)*(12/56)=(144/1848)=6/77	**386^5=** 386^5=8569125814176	
-129/-334= -129/-334=129/334=0.3862275449101796	**723^10=** 723^10=39028600001675809908107504649	**(-2714.9607*4215%)--7850=** (-2714.9607*4215%)--7850=(-2714.9607*42.15)--7850 =(-114435.593505)--7850=-114435.593505--7850 =-114435.593505+7850=-106585.593505

Figure 2: Examples of MathGLM's response on a variety of arithmetic tasks.

E Baseline Models for MWP

Here, we leverage a variety of popular LLMs that can address Chinese problems to compare the mathematical reasoning ability among these LLMs and our MathGLM. The details of each baseline LLM as follows.

- GPT-4 (OpenAI, 2023) is the most advanced generative language model that developed by OpenAI, which successfully achieves so many SOTA performances on a variety of downstream tasks.
- ChatGPT (OpenAI) is the predecessor of GPT4 and is constructed upon the success of InstructGPT (Ouyang et al., 2022), which is fine-tuned using instruction data with reinforcement learning from human feedback (RLHF), making it a powerful tool for natural language understanding and conversation.
- MOSS (Sun and Qiu) is an open-source LLM that consists of 16 billion model parameters. It utilizes 100 billion Chinese tokens and 20 billion English tokens to learn language patterns and semantic representations.
- Ziya-LLaMA-13B (Zhang et al., 2022) is a language model constructed on LLaMA-13B, which extends LLaMA-13B's character set to contain 7,000 Chinese characters and undergoes continual pre-training on a vast dataset of 110 billion Chinese tokens.
- Chinese-Alpaca-13B (Cui et al., 2023) is a Chinese language model with 13 billion parameters that is built upon LLaMA-13B. During the supervised instruction tuning, the Low Rank Adaptation (LoRA) (Hu et al., 2021) technique is utilized to fine-tune LLaMA-13B for Chinese language tasks.
- Baichuan-7B (inc.) shares similarities with LLaMA but is pre-trained from scratch on a massive dataset containing 1.2 trillion Chinese and English tokens.
- ChatGLM-6B (THUDM, a) and its successor ChatGLM2-6B (THUDM, b) are language models that share a unified transformer architecture named GLM (Du et al., 2021; Zeng et al., 2022). These models are pre-trained on a diverse dataset containing English and Chinese data, combined with the supervised instruction tuning, makes them powerful tools for understanding and generating text in both English and Chinese contexts.

F Additional Experiments on MWP

F.1 Comparison of Training Strategies

Here, we evaluate the mathematical reasoning ability of MathGLM with different training strategies: fine-tuning and continue training. To execute continue training, we amalgamate the Ape210K train dataset with instruction data released by Chinese-Vicuna (Chenghao Fan and Tian, 2023). We subsequently continue training MathGLM from the GLM-10B backbone. Table 11 shows the overall performance comparison of MathGLM employing different training strategies. We observe that directly fine-tuning on the specific dataset can achieves better performance.

Table 11: Overall performance comparison on various LLMs in term of Accuracy.

Training	w/o step-by-step strategy		with step-by-step strategy	
	Arithmetic$_{Acc}$	Answer$_{Acc}$	Arithmetic$_{Acc}$	Answer$_{Acc}$
Fine-tuning	71.38%	41.24%	69.08 %	58.68%
Continue training	70.16%	40.34%	67.02%	56.60%

F.2 Scaling Analysis

To explore the impact of scaling on MathGLM, we conduct a series of experiments encompassing varying dataset sizes and distinct model parameters. Table 12 demonstrates the results obtained from varying the dataset sizes within the range of $\{5K, 10K, 20K, 50K, 100K, 200K\}$. Furthermore,

to understand the impact of different model parameters, we incorporate various backbone models into MathGLM, including GLM-Large (335M), GLM-6B, and GLM-10B. The results consistently indicate that MathGLM's performance improves across all backbone models with the increase in dataset size. Such observation highlights the beneficial effects of enlarging the training data on bolstering MathGLM's proficiency in tackling math word problems. By accessing more extensive datasets, MathGLM is introduced to a wider array of problem types, resulting in better performance. Additionally, discernible differences in performance emerge among the various backbone models. Given sufficient dataset size, larger models like MathGLM-GLM-10B often outperform others, indicating the crucial role of model parameters in addressing intricate math word problems. These insights emphasize the significance of both dataset and model scaling. By augmenting dataset size and utilizing larger models, we can markedly boost MathGLM's capability to generate more accurate solutions, enhancing its overall efficacy in resolving math word problems.

Table 12: Performance comparison of MathGLM on different training dataset sizes and model parameters.

Model Scale	MathGLM-GLM-Large	MathGLM-GLM-6B	MathGLM-GLM-10B
5K Problems	4.32%	12.84%	3.68%
10K Problems	7.14%	19.78%	6.36%
20K Problems	10.36%	21.89%	9.62%
50K Problems	18.32%	26.40%	16.78%
100K Problems	25.98%	31.44%	22.20%
200K Problems	35.68%	34.00%	38.10%

F.3 Failure Analysis on Math Word Problems

Figure 3 provides some failed examples generated by MathGLM-GLM-10B on solving math word problems. We can identify certain challenging scenarios where MathGLM-GLM-10B encounters difficulties in solving math word problems. One common issue is the misinterpretation of ambiguous language, leading to incorrect problem-solving approaches. For instance, ambiguous phrases such as "more than" or "less than" can be interpreted differently by the model, resulting in inaccurate solutions. Additionally, MathGLM-GLM-10B tends to struggle with problems that involve complex mathematical operations. As a result, it may provide partially correct arithmetic solutions but fail to arrive at the final correct answer.

F.4 Training Steps Analysis.

We explore the impact of training steps on the MathGLM's performance by analyzing its performance against varied training steps, as depicted in Figure 4. The results reveal that there is a consistent uptrend in performance as the number of training steps increases. With more training steps, MathGLM becomes increasingly adept at comprehending and resolving math word problems, which translates to a surge in accuracy. However, it is clearly observed that the performance gains of MathGLM start to plateau after a certain point, indicating potential diminishing returns with extended training. These findings highlight the significance of finding an optimal balance between training time and performance gains for MathGLM in solving math word prblems. Additionally, we observe that model undergoing instruction tuning requires a longer training duration to achieve consistent accuracy on math word problems.

F.5 Case study

Here, we present specific cases to demonstrate the solving process of the MathGLM on both arithmetic tasks and math word problems. As shown in Figure 5, these examples illustrate how the MathGLM leverages a step-by-step strategy to solve both arithmetic tasks and math word problems accurately.

For arithmetic tasks, the MathGLM breaks down complex calculations into multiple steps rather than a direct answer. The step-by-step strategy ensures that each intermediate result is accurately computed, leading to the final correct answer for each arithmetic expression.

Type	Question	Generated Answer	Ground Truth
Question Misunderstood Error	小红看一本小说，第一天看了总页数的25\%少17页，第二天看的比总页数的(1/8)还多16页，还剩下61页没看，这本小说一共有多少页?	x=(17+16+61)/(1−25%−(1/8))=(17+16+61)/(1−0.25−(1/8))=(33+61)/(1−0.25−(1/8))=94/(1−0.25−(1/8))=94/(1−0.25−0.125)=94/(0.75−0.125)=94/0.625=144.	x=(61−17+16)/(1−(1/8)−25%)=96
	牛牛用10元钱买了一个玩具，胖胖对他的玩具非常喜欢，花20元钱买了过去．过了几天，牛牛有些舍不得，又花30元钱从胖胖手里买了回来．又过了几天，胖胖花40元又把玩具从牛牛手里买走．在整个过程中，牛牛共赚了多少元钱?	x=20+30+40=50+40=90	x=40−30+20−10=20
Calculation Error	小马虎在甲数一道减法算式时，错把减数65看成了56，结果得到差是215，正确的差是多少?	x=215+(56−65)=215+9=222	x=215+56−65=206
	学校计划装修16间教室，工程队经过测算，每间教室需要铺地砖192块，工程队应该购买这样的地砖多少块?	x=1920*16=32640	x=192*16=3072
Knowledge Error	一个长方形的周长是88cm，长是30cm，它是面积是多少cm**2?	x=30*88/(30*2)=30*88/60=2640/60=44	x=30*(88/2−30)=420
	一个长方体的长和宽都是4米，高是5米，如果底面积扩大5倍，要使体积不变，高应该是多少厘米?	x=(4*5*5)/5=(20*5)/5=100/5=20	x=(5/5)*100=100
Common-sense Error	两地相距13千米，甲乙二人从两地同时出发相向而行，经过(4/3)小时相遇．甲每小时行5千米，乙每小时行多少千米?	x=(13−5*4/3)/(4/3)=4.75	x=(13−5*4/3)/4=(13−20/3)/4=(13−6.666666666666667)/4=6.333333333333334/4=1.666666667
	一艘客轮往返甲、乙两港，顺水速度是15千米/小时，逆水速度是的12千米/小时．现在甲港放一个木排顺水漂流到乙港，要用3天才能到达．那么，甲、乙两港的水路长多少千米.	x=((15−12)/2)*72=108	x=(15+12)*3=27*3=81

Figure 3: Some failed examples generated by MathGLM-GLM-10B on solving math word problems.

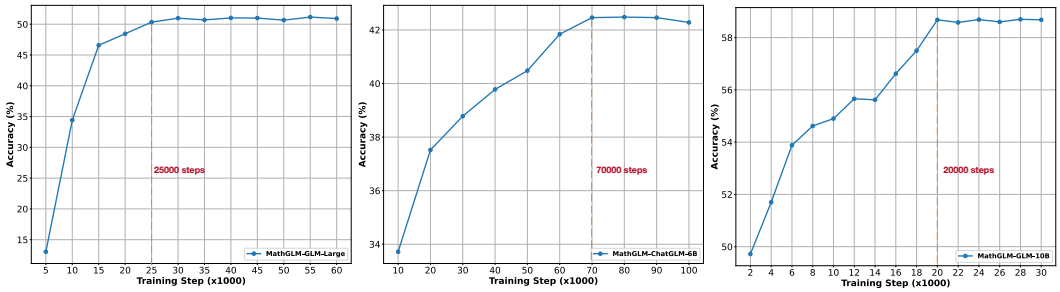

Figure 4: The impact of training steps on MathGLM with different backbone models. Fine-tuning on model undergoing instruction tuning requires a longer training steps.

In math word problems, the MathGLM utilizes LLMs's language understanding capabilities to understand the problem statement accurately. The MathGLM then applies the step-by-step strategy to seperately calculate the generated arithmetic expressions for each math problem, enabling it to obtain the correct answer for math word problems.

Math Word Problems		Arithmetic Tasks	
Input	Output	Input	Output
王艳家买了一台洗衣机和一台电冰箱,一共花了6000元,电冰箱的价钱是洗衣机的(3/5),求洗衣机的价钱.	答:x=6000/(1+(3/5))=6000÷(1+3/5)=6000÷(5/5+3/5)=6000÷8/5=30000/8=3750	-7439+(3156+1649%)+(-2307+-1112)=	-7439+(3156+16.49)+(-2307+-1112)=-7439+3172.49+(-2307+-1112)=-7439+3172.49+(-2307-1112)=-7439+3172.49+-3419=-7439+3172.49-3419=-4266.51-3419=-7685.51
甲乙两车从相距180千米的两地同时相向而行,甲车每小时行50千米,乙车每小时行40千米,几小时后两车相遇?	答:x=180/(50+40)=180/90=2		
两个因数相乘,其中一个因数扩大10倍,要使积不变,另一个因数要缩小为原来的多少?	答:x=(1/10)	511 4392.7*1127/[(6532--3588)+-1499.90071*-3871.5083]=	4392.7*1127/[(6532+3588)+-1499.90071*-3871.5083]=4392.7*1127/[10120+-1499.90071*-3871.5083]=4392.7*1127/(10120+1499.90071*3871.5083)=4392.7*1127/(10120+5806878.047940893)=4392.7*1127/5816998.047940893=4950572.899999999/5816998.047940893=0.8510528728389893
一个长方体的体积是48m**3,长是8m、宽是5m、高是多少m?	答:x=48/(8*5)=48/40=1.2		
李枫看一本书,第一周看了全书的45%,第二周看了全书的(2/5),还有60页没有看.这本书一共有多少页?	答:x=60/(1−(45%+(2/5)))=60/(1−(0.45+(2/5)))=60/(1−(0.45+0.4))=60/(1−0.85)=60/0.15=400		
教室长8m、宽6m、高3m,六1班有48名学生,平均每人占有的空间是多少?	答:x=8*6*3/48=48*3/48=144/48=3	2824/2202-[8553-(156-4493)/9780*4740]*8637=	2824/2202-[8553--4337/9780*4740]*8637=2824/2202-(8553+4337/9780*4740)*8637=2824/2202-(8553+0.4434560327198364*4740)*8637=2824/2202-(8553+2101.9815950920245)*8637=2824/2202-10654.981595092025*8637=1.2824699818346957-10654.981595092025*8637=1.2824699818346957-92027076.03680982=-92027074.75433984
加工一批零件,甲单独做要10天完成,乙单独做要8天完成.现在甲、乙两人合作,多少天能完成这些零件的一半?	答:x=(1/2)/((1/10)+(1/8))=1/2÷(1/10+1/8)=1/2÷(4/40+5/40)=1/2÷9/40=40/18=20/9		
一种商品原价780元,如打八折出售,那么价格降价多少?	答:x=780*(1−80%)=780*(1−0.8)=780*0.2=156		
每个书架有3层,每层都装50本书,有4个这样的书架,一共装了多少本书?	答:x=50*3*4=150*4=600	1360*5664+8698=	7703040+8698=7711738

Figure 5: Some cases generated by MathGLM on arithmetic tasks and math word problems.

REFERENCES

Zhenyi Lu Chenghao Fan and Jie Tian. Chinese-vicuna: A chinese instruction-following llama-based model. 2023. URL https://github.com/Facico/Chinese-Vicuna.

Yiming Cui, Ziqing Yang, and Xin Yao. Efficient and effective text encoding for chinese llama and alpaca. *arXiv preprint arXiv:2304.08177*, 2023.

Ming Ding, Wendi Zheng, Wenyi Hong, and Jie Tang. Cogview2: Faster and better text-to-image generation via hierarchical transformers. *Advances in Neural Information Processing Systems*, 35:16890–16902, 2022.

Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. Glm: General language model pretraining with autoregressive blank infilling. *arXiv preprint arXiv:2103.10360*, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Baichuan inc. Baichuan-7b. https://github.com/baichuan-inc/baichuan-7B/blob/main/README_EN.md.

OpenAI. Chatgpt. https://mkai.org/chatgpt-optimizing-language-models-for-dialogue/.

OpenAI. Gpt-4 technical report, 2023.

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.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Tianxiang Sun and Xipeng Qiu. Moss github. https://github.com/OpenLMLab/MOSS/blob/main/README_en.md.

THUDM. Chatglm-6b. https://github.com/THUDM/ChatGLM-6B/blob/main/README_en.md, a.

THUDM. Chatglm2-6b. https://github.com/THUDM/ChatGLM2-6B/blob/main/README_EN.md, b.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, and Songfang Huang. How well do large language models perform in arithmetic tasks? *arXiv preprint arXiv:2304.02015*, 2023.

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*, 2022.

Jiaxing Zhang, Ruyi Gan, Junjie Wang, Yuxiang Zhang, Lin Zhang, Ping Yang, Xinyu Gao, Ziwei Wu, Xiaoqun Dong, Junqing He, Jianheng Zhuo, Qi Yang, Yongfeng Huang, Xiayu Li, Yanghan Wu, Junyu Lu, Xinyu Zhu, Weifeng Chen, Ting Han, Kunhao Pan, Rui Wang, Hao Wang, Xiaojun Wu, Zhongshen Zeng, and Chongpei Chen. Fengshenbang 1.0: Being the foundation of chinese cognitive intelligence. *CoRR*, abs/2209.02970, 2022.

Wei Zhao, Mingyue Shang, Yang Liu, Liang Wang, and Jingming Liu. Ape210k: A large-scale and template-rich dataset of math word problems. *arXiv preprint arXiv:2009.11506*, 2020.