Journal of Software Engineering and Applications, 2024, 17, 23-42
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Smaller & Smarter: Score-Driven Network Chaining of Smaller Language Models

Gunika Dhingra[1], Siddansh Chawla[1], Vijay K. Madisetti[2], Arshdeep Bahga[3]

[1]School of Computer Science Engineering & Technology, Bennett University, Greater Noida, India
[2]School of Cybersecurity and Privacy, Georgia Institute of Technology, Atlanta, USA
[3]Cloudemy Technology Labs, Chandigarh, India
Email: 12gunika@gmail.com, chawlasiddansh@gmail.com, vkm@gatech.edu, arshdeep@cloudemy.io

How to cite this paper: Dhingra, G., Chawla, S., Madisetti, V.K. and Bahga, A. (2024) Smaller & Smarter: Score-Driven Network Chaining of Smaller Language Models. *Journal of Software Engineering and Applications*, **17**, 23-42.
https://doi.org/10.4236/jsea.2024.171002

Received: December 1, 2023
Accepted: January 20, 2024
Published: January 23, 2024

Abstract

With the continuous evolution and expanding applications of Large Language Models (LLMs), there has been a noticeable surge in the size of the emerging models. It is not solely the growth in model size, primarily measured by the number of parameters, but also the subsequent escalation in computational demands, hardware and software prerequisites for training, all culminating in a substantial financial investment as well. In this paper, we present novel techniques like supervision, parallelization, and scoring functions to get better results out of chains of smaller language models, rather than relying solely on scaling up model size. Firstly, we propose an approach to quantify the performance of a Smaller Language Models (SLM) by introducing a corresponding supervisor model that incrementally corrects the encountered errors. Secondly, we propose an approach to utilize two smaller language models (in a network) performing the same task and retrieving the best relevant output from the two, ensuring peak performance for a specific task. Experimental evaluations establish the quantitative accuracy improvements on financial reasoning and arithmetic calculation tasks from utilizing techniques like supervisor models (in a network of model scenario), threshold scoring and parallel processing over a baseline study.

Keywords

Large Language Models (LLMs), Smaller Language Models (SLMs), Finance, Networking, Supervisor Model, Scoring Function

1. Introduction

LLMs are the neural networks that have the capability of understanding and generating human-like text. As an LLM grows to be better at delivering state of the

art results, it still lacks the ability to solve complex problems in various specialized domains, like finance, healthcare, and law, with precision and accuracy. While the attempts to improve the result quality intensify, these LLMs become extensively intricate and resonate a vast network of intensive parameters instigating a deeper level of complexity in terms of its architecture, processing and even the computational requirements of resources. Such larger language models often require highly efficient clusters of high performance GPUs which consume significant time, costs, and electricity.

In context of their performance and diverse tasks that they handle, it is important to note that finetuning [1] is considered an essential aspect. This process employs training the model on a smaller and task-specific dataset to tailor its potential to a particular use case, making it more contextually relevant and efficient. Additionally, recent and advanced techniques like Chain of Thought prompting [2] enable a Large Language Model to get better at reasoning and eventually cultivate results with markedly higher accuracy. However, this approach (typically) works effectively on models with about a 100 billion parameters or more [3]. The prospect of considering a smaller language model to carry through a complex task still remains a concern.

For the scope of this paper, we refer to smaller large language models (or SLMs) as models with a relatively low number of parameters, typically less than 10 billion. We propose an approach of incorporating a supervisor model in a network of multiple smaller language models (SLMs) as well as inducing an assistive model, for confronting complex tasks to yield effectiveness by the mechanism of the supervisor model employed and the helper mechanism induced. In order to refine our scope towards a target application and experimentally substantiate the hypothesis, this paper is directed towards the task of financial reasoning and arithmetic calculation of nested operations, domains where even prominent and heavy-duty models often struggle to perform adequately.

The objective of this research is to propose an assembly (or, network) of smaller models that when subjected to specialized finetuning, not only prove to be a better system for exercising penetrability and adding a sense of accountability, but also increase the quality of the outcomes. The interconnected workflow suggests a network of SLMs, wherein models shall be paired up with their supervisors, wherever necessary, we refer to this as the Threshold-Based approach. The supervisor model employed can be portrayed like a checking mechanism. Together, their cumulative efforts are aimed at upholding and ensuring the highest possible quality of the final outcome. This collaborative network model not only streamlines the process but also establishes a structured framework for achieving superior results in diverse tasks.

While the second proposed approach involves a similar setting it is different in scope. In this approach, two versions of the same SLM [with variations either in the dataset or hyperparameters, selection of which is task-dependent] perform the task concurrently to generate results. Further, the one that has the most contextually relevant output is chosen to be taken forward in the chain. We refer

this approach as the Parallel Processing method. This setting can be visualized as a competitive game-theoretic approach, where multiple agents solve the same problem and the best answer is chosen out of the lot. While this approach involves parallel computation, it also ensures that no sub-optimal results are taken forward in the chain.

By the means of a network of SLMs, a user can particularly handle more sophisticated tasks without exhausting much of the resources. Our suggested approaches can be implemented in similar way and adapted to other complex tasks in diverse specialized fields, to achieve desired results.

The key research contributions and novel aspects of the paper are as follows:

• Proposes an approach to enhance the performance of a chain of smaller language models (SLMs) for complex tasks by introducing a corresponding supervisor model that incrementally corrects errors.

• Suggests utilizing two smaller language models in a network performing the same task and retrieving the best relevant output, ensuring peak performance.

• Introduces a threshold-based approach where a supervisor SLM audits and corrects errors from the main SLM to optimize efficiency.

• Proposes a parallel processing method with two versions of an SLM trained with variations either in the dataset or hyperparameters, selection of which is task-dependent.

• Demonstrates the application of these approaches for financial reasoning and arithmetic calculation tasks using tailored scoring functions to choose optimal outputs.

• Compares performance of large language model, network of SLMs with supervisor model, and parallel processing on chosen domain of tasks.

• Aims to solve issues like high computational demands of large models while retaining performance for complex tasks by employing network of SLMs.

2. Problem Statement

Large Language Models (LLMs) exhibit substantial dimensions, characterized by both a high parameter count and extensive computational requirements. Managing storage, employing the required computational units and incurring the associated costs elevate as the scale of the model increases. Hence, such LLMs pose challenges across various aspects including hosting, training, and finetuning. To learn and understand patterns, large language models usually require large datasets for training purposes and hence more computing resources for training. While this places them at a superior level for returning outcomes for generalized tasks, they often lack the ability to do so in the specialized domains.

To handle these situations, there exist two primary approaches for imparting knowledge to the language models: prompt-engineering, which involves in-context learning and the other being "finetuning". Fine-tuning encounters limitations in scale: as the model expands and continuously generates even larger amounts of data, it becomes infeasible to retrain the large language model on all newly

available information which refrains the likelihood of ideal performance at intricate tasks.

The objective of the study is to develop and assess an approach of employing a network of smaller language models, to address the previously mentioned challenges, reflecting an aim to resolve these issues.

3. Literature Survey

3.1. Prompt Engineering

Working with Large Language Models often results in very high computational and cost usage, yet they fail to deliver the desired results for complex tasks. However, there are multiple methods and techniques which can enhance the model output with the help of reasoning, one such being the Chain of Thought prompting method (CoT) [2]. It is a simple technique that enables the large language models to work around a complex problem just like a human would approach to solve it, hence it carefully reasons appropriate steps and accordingly takes the desired move forward. There have been prominent and promising results shown by the chain of thought prompting technique. A recent study [4] carried out with the GSM8K dataset, indicates that the results obtained from the CoT approach while employing the largest models in the GPT and PaLM family (with 60 billion parameters or more), were more than doubled their baseline comparative results. This reflection is attributed to the influence of the "chain of thought" prompts which induces reasoning, thereby making a language model interpret it own steps, leading to significant enhancements in the future results.

3.2. Effect of Distilled Learning on Reasoning Tasks

Even though the Chain of Thought (CoT) operational procedure seems to hold great potential, it harbours an issue which arises when the approach is tested on the language models, particularly with less than 100B parameters [3]. The experimental findings demonstrate that the Chain of Thought prompting techniques often yielded near zero performance even on the foundational reasoning tasks, making it a significant challenge for the SLMs. To devise a solution to this, it was proposed that instead of employing the SLMs to work on a variety of tasks and aim for an overall performance, it would be much more effective to implement distilled learning, a technique where the model is equipped to focus on a very specific task using all of its computational power and somehow losing the ability to perform on generic task. The approach used here is finetuning a model on a dataset, which consists of CoT rationales, which are step-by-step solutions either generated by an LLM (like GPT3.5 or GPT-4) or curated by an expert. To illustrate the potential, the paper [5] produced a fine tuned version of FLAN-T5 model which evidently reflected an increase of 11 points on reasoning tasks using the CoT approach and a decrease of 27.8 on the generic tasks. The reasoning results were comparable to those of the PaLM-60B model.

3.3. Learning from Mistakes

To further improve the ability of LLMs to handle and work with mathematical data, authors of the paper LeMa [6] introduced a novel approach to enhance the reasoning capabilities by emulating human error-driven learning processes. Known as Learning from Mistakes, this approach revolved around fine-tuning LLMs using CoT data with mistake-correction data pairs generated by GPT-4. The process involves collecting inaccurate reasoning paths from various LLMs, leveraging GPT-4 as a "corrector" to identify, explain, and rectify mistakes, ultimately producing the final answer. Experimental results showcase the effectiveness of LeMa, consistently improving the performance of multiple backbone LLMs across two mathematical reasoning tasks when compared to fine-tuning on CoT data alone. The paper underscores the potential of error-driven learning processes to enhance LLM reasoning capabilities, acknowledging that models like Llama 2 still encounter challenges, leaving room for improvement. Furthermore, comparisons indicate that larger LLMs are better at learning from mistakes.

3.4. Advocating the Usage of Smaller Models

3.4.1. Financial Reasoning

A recent study [7] laid rich conclusions in this domain by exhibiting that financial reasoning ability is first shown in language models with at least 6B parameters. Their ability to answer, can be further improved by either employing instruction tuning or the usage of larger corpora. As detailed in a scholarly exploration, Meta's Llama 13B model was able to establish an average ROUGE-L score of 0.273 along with an average BERTScore of around 0.845 on the financial reasoning task. While a similar test when conducted with GPT-J 6B, achieved the average metric of ROUGE-L of 0.122 and BERTScore of 0.788. The results are quite promising for the seed stage of working. Taking into consideration the results of this research, we employ a consistent approach in the preliminary phase which utilizes the Llama-2 model with 7 billion parameters.

3.4.2. Arithmetic Calculation

Another research performed on a small language model's arithmetic ability, resulted in creation of Goat [8], a fine-tuned version of the Llama 7B model that outperformed GPT-4 (about a trillion parameters) in variety of BigBench [7] testing metrics over multiple dimensions. It used the strategy to decompose a complex mathematical problem using the basic arithmetic principles. This study also reflected that when fine-tuned on the same dataset, models such as Bloom, OPT, GPT-NeoX and Pythia cannot match Llama's capability to work with numerical data. Due to the fact that Llama's tokeniser splits digits of a number individually, hence employing it for arithmetic tasks is perceived as a favourable aspect. While this study has provided substantial results in the realm of arithmetic data and its processing, our aim is to draw inspiration from its findings and delve into the exploration of numerical answering to the arithmetic nested

operations-an area that, to the best of our knowledge, remains largely untouched.

3.5. LLM Cascading

Following the idea of minimising the cost of the usage of Language models, a recent framework, FrugalGPT [9] offers a collection of techniques for building Large Language Model (LLM) applications with budget constraints, aiming to reduce costs while improving performance. It mainly suggests three main techniques: Prompt adaption, LLM Approximation and LLM Cascades. While all three approaches exhibit merit, the relevance of LLM cascades stands out in our study. LLM cascade sends a query to a list of LLM APIs sequentially. If one LLM API's response is reliable, then its response is returned, and no further LLMs in the list are needed. The remaining LLM APIs are queried only if the previous APIs' generations are deemed insufficiently reliable. While this approach is designed to minimize costs, the introduction of a heavy-duty model like GPT-4 in instances where no LLM API delivers satisfactory results can potentially disrupt the cost structure. The LLMs are arranged in ascending order of size, leading to increased computational requirements. Notably, this approach does not propose to rectify the mistakes of the preceding model. Each query passed to a subsequent model with higher parameter count is treated as a fresh query, without leveraging insights from the previous models.

4. Dataset

4.1. Baseline Dataset Curation

4.1.1. Dataset for Financial Reasoning

As a preliminary step of this research to prove the hypothesis, we utilized the FinQA [10] dataset which is a large scale corpus of data of the earning reports of S&P500 companies. The entries are hence based on its proposed approach of retrieval and program generator, which resulted into multiple specialized finance question and answer pairs. To outline the dataset hierarchy, FinQA has text labels for both pre/post of a table as well as it includes complex financial questions with their respective answers, logic program, explanation and model inputs. The data was cleaned by fixing the formatting by catering to the issues such as missing values in some features like explanation and answer. The instances which reflected evidence of such shortcomings were omitted to ensure good quality results. A comprehensive description of the dataset can be found in **Appendix A**.

This dataset is employed by the models to perform the task of financial reasoning from a given set of financial information data. The required data was extracted from various files provided in the FinQA dataset. The selected training entries comprised of 810 rows and 2 input features and 2 output features. In the initial step, we added two more output features by further evaluating one of the features. In addition to this, data_points, a new attribute, was created using GPT-4 and GPT3.5-Turbo. This contained the necessary data points which were extracted from the input feature. This entirely was elaborated to obtain a Chain of Thought Rationale which was further utilized to train the model. The final

dataset consisted around 2 input features and 4 output features which when cumulated behaved as an CoT rationale. To compare the scores of our approaches with the baseline, we further compiled a dataset consisting 100 random testing examples from the FinQA.

4.1.2. Dataset for Arithmetic Calculation

We crafted a synthetic dataset to make the language model learn how to approach when exposed to arithmetic tasks. The justification for generating this dataset primarily stems from the evident observations that LLMs struggle with basic arithmetic computation and often make arithmetic mistakes, even though the reasoning process is correct. Considering the practice used by the authors in the publication, titled "Goat" [8], we instilled a similar process to generate random complex nested arithmetic expressions with its simplified mathematical operation, the steps to solve the expression and eventually the final answer. The dataset generated presents a graded range of expressions, progressing from basic to complex, thereby promoting enhanced retention and learning. The nested expressions comprises both standalone instances as well as combination of the four foundational operations: addition, subtraction, multiplication, division. The data is generated using a python script and has about 2K entries following the set standards laid down for it. While there is an assurance against the duplication of generated data due to the random numbers generated by the script, however the numbers with particularly fewer digits may occasionally recur. Detailed information of the dataset is extensively covered in **Appendix B**. The process of dataset curation has been visually summed up in **Figure 1**.

4.2. Dataset for Proposed Approach

Our proposed approach of supervision requires curating a dataset for a supervisor model. This can done by inferencing the finetuned model on a test dataset and comparing its solutions with the gold solutions. By inferencing the model on large test dataset, we can generate enough training labels that cumulate into the dataset for training the supervisor model. Therefore, the supervisor SLM dataset maps the false predictions by the model. This process is conducted for curating the datasets for both the target applications.

5. Methodology

To solve complex tasks like financial reasoning and arithmetic calculations using language models has proven to be difficult for not only small language model, but also for models that are huge in size with parameters in higher billions and trillions. To improve the efficacy of smaller language model on such tasks, we propose to implement the following approaches.

5.1. Baseline Approach

The baseline approach reflects the usage of a large language model to lay the foundation of comparative analysis to assess the performance of our proposed

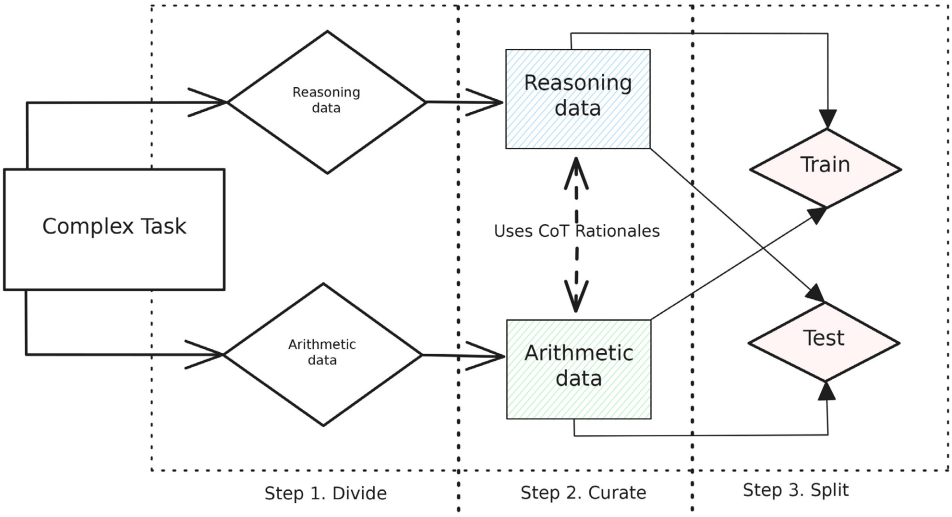

Figure 1. Dataset Curation for Financial reasoning and Arithmetic Calculation.

approaches. We utilized the Llama2-13B model [11] to devise the baseline results. This framework utilizes the Language Models which are made adept to the problem in the bigger aspect by using multiple techniques. For the scope of this research, the study revolves around the tasks in the financial and arithmetic domain, further extending it to the realm of question and answering with quantitative data and nested mathematical expressions. These chunks of tasks can be easily handled independently by language models when they are finetuned for the specific purpose on a particular dataset. The models are fine tuned using distilled learning strategies which are made to instill the Chain-of-Thought rationales. This process of refinement using such strategies when repeated on a base model for different tasks with respective dataset results into a new fine tuned model.

Fine Tuning Using Distillation

The strategies implemented differ for each model due to difference in task and the dataset. Both the models were quantized in 4-bit, then fine tuned using QLoRA [12] technique using bf16 on a Nvidia RTX a 5000 GPU. Configuration for both were taken as low rank as 64, alpha equal to 64 with 0.05 as the dropout rate and the learning rate was set as 0.0002. The hyperparameters were decided by working out multiple combinations while also reading multiple sources [13] on task specific QLoRA parameter efficiency. The distinct point of difference were the training steps for each model, 1600 and 3400 for financial reasoning and arithmetic calculation, respectively. This was determined using the complexity of task in hand the amount of data which was used for training. Both the models were fine tuned using Supervised Fine tuning trainer and were able to achieve satisfactory results during inference testing.

5.2. Our Proposed Approaches

The implementation of our approaches requires a selection of a Smaller Language Model (SLM), for the purpose of financial reasoning and arithmetic cal-

culation which is chosen to be Llama2-7B. We follow a similar baseline implementation and finetune (2x) Llama2-7B models for our tasks. The finetuning process was the replica of what we did in the baseline methodology. The further sections uses the word "SLM-x", which essentially denotes finetuned Llama2-7B model for task-1 and task-2 *i.e.*, financial reasoning and arithmetic calculation respectively.

5.2.1. Proposed Approach 1: Threshold Based Method

This methodical approach is implemented with additional refinements and augmentations including a supervisor language model trained for enhancing the output of the used SLM, in our use case. The role of this supervisor is to act as the corrective mechanism for the employed language model. It works on a similar strategy implemented in LeMa [6] while the distinguishing factor is that the supervisor model in our approach is a separate entity in the network. A sample supervisor prompt would take into consideration the previous model solution, the question and a prompt stating to correct the former solution. It would then rectify the step where the previous model went wrong and simultaneously correct the final answer. The supervisor ensures that its corresponding SLM adheres to delivering the right kind of output, offering corrective feedback and guidance to optimize the overall efficiency of the system. **Figure 2** illustrates the proposed approach. Unlike the other approaches employed [9], where the size of the large language models incrementally increases in the chain, our approach utilizes the same size of SLM as the supervisor model.

Figure 3 denotes the process of curating a dataset required for finetuning a supervisor model for the specific task. For the tasks, we inferred the SLMs using a validation set and mapped the output with the gold solutions [the correct solution to a given problem] using various NLP techniques and the algorithm devised for the desired output. The supervisor model is therefore made adept to learn the mistakes made by its corresponding SLM thereby ensuring a supervisory, corrective process akin to a corrector providing guidance and corrections.

Chaining of finetuned models: Followed by the process of fine tuning, the language models were first loaded from hugging face using pipeline function of the transformers module. Each model had its own LLMChain [14] setup with the help of LangChain [15]. The second LLMChain was only called when the output from the first LLMChain did not pass the scoring criteria. The final step of the approach involved inferencing of the the chains on the test data, similar to what is used for inferencing from Large language model in the baseline approach.

Cost saving: To further optimize the usage in terms of invoking cost, we employed a scoring function which acts as a filter for the network. All the outputs a SLM are subjected to a scoring function. The supervisor is only called when the scoring function determines the output of a SLM is insufficiently lagging reason or has predicted a wrong outcome.

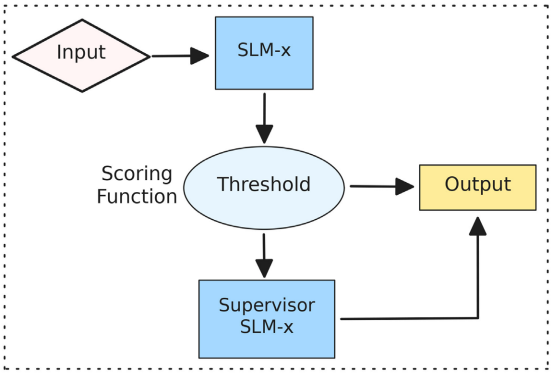

Figure 2. Proposed network for threshold-based approach. SLM and supervisor SLM here denotes the finetuned version of Llama2-7B model. This unit can be replicated for other tasks and when chained together, can potentially solve a complicated problem.

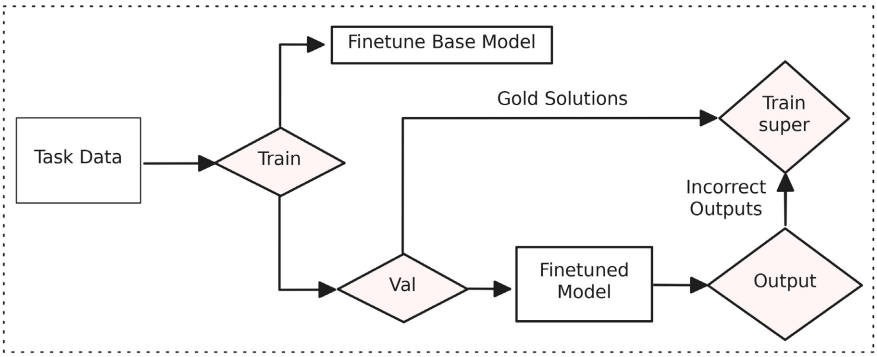

Figure 3. Dataset Curation for the incorporation of the supervisor model as proposed in our study.

Scoring function: A key component of the chain is the scoring function attached to the SLM in the network. The introduction of the scoring function is grounded on the principle that any below-par performance by the predecessor language model, results in the subsequent degradation of the entire chain's effectiveness. It is developed on the basis of our targeted application of financial numerical reasoning, hence it is worth noting that different use cases might have different metrics of evaluating the performance of the language models.

1) Financial reasoning: The comprehensive scoring metric, tailored for evaluating financial reasoning within the Model 1 and its supervisor chain for conducting financial numerical reasoning, encompasses a sequential workflow. This involves deriving an explanation from the context information and question, fetching necessary data points, and generating a program used for arithmetic calculations. The scoring metric evaluates parameters such as the exact match of numbers and operators, quantifies deviations, and checks for correct operator usage, aligning with the information provided in the questions. Additionally, it ensures accurate placement of operators, especially within nested operations, through a recursive function. Both approaches are equally weighted in calculat-

ing the final score, ranging between 0 and 1. For a deeper understanding, Algorithm 1 detailing the function can be referenced.

Algorithm 1: Model Output Analysis → Checking operations, numbers & relevant factors

Input: syn_dict, R_nums, W
Output: score
Data: question(Q), logic(L), data_points(D), steps(S)

$R \leftarrow$ `ReverseMapping`(syn_dict)
$ops\_Q, ops\_L, nums\_D \leftarrow emptysets$
while $\neg converged$ **do**
  $ops\_Q \leftarrow extract(Q, R)$ `// extraction through reverse mapping`
  $ops\_L \leftarrow extract(L, R)$
  $nums\_D \leftarrow extract(D)$ `// fetch numerical values from data_points`
  $ops\_S, nums\_s \leftarrow extract(S)$
  $ops\_QL \leftarrow ops\_Q \cap ops\_L$ `// conjunction of operators in question and logic`
  $nums\_DS \leftarrow nums\_D \cap nums\_S$ `// conjunction of operators in data_points and steps`
  $s\_logic \leftarrow ops\_QL/ops\_Q$
  $s\_steps \leftarrow |ops\_S \cap ops\_L|/|ops\_L| - max(0, |ops\_S| - |ops\_L|)/|ops\_S|$
  $s\_data \leftarrow |nums\_DS|/|nums\_D|$
  `/* cumulative score achieved by summing weighted partial scores */`
  $score \leftarrow (0.5 * s\_logic + 0.25 * s\_steps + 0.25 * s\_data)$
return *score*

2) Arithmetic calculation: The scoring function, akin to the one utilized for the financial reasoning task, evaluates the performance of SLM that performs arithmetic calculation. It assesses the performance of the model in the sequence by quantifying its accuracy by checking the closeness of two values based on predefined relative and absolute thresholds. In our study, this threshold is taken to be 5% The algorithm returns True if the absolute difference is within the absolute threshold, or if the second value falls within a relative range of the first value, as defined by the relative threshold. If neither condition is met, it concludes the values are not close and returns False. Refer to Algorithm 2 for getting an in-depth analysis of the utilized function.

Algorithm 2: Model Output Analysis → Checking proximity of numeric values

Input: v1, v2, rel_thresh, abs_thresh
Output: Proximity
Data: rel_thresh, abs_thresh

$v1, v2 \leftarrow round(float(3))$
$abs\_dif \leftarrow |v2 - v1|$ `// calculate the absolute difference between the values`
`/* generating a dynamic absolute threshold */`
$dyn\_abs\_thresh \leftarrow max(abs\_thresh * max(v1, v2), abs\_thresh)$
if $abs\_dif \leq dyn\_abs\_thresh$ **then**
  return *True*
if *v2 ≤v1*(1+rel_thresh) & v2 ≥v1*(1-rel_thresh)* **then**
  return *True* `// assessing the proximity above and below the threshold`
else
  return *False*

5.2.2. Proposed Approach 2: Parallel Processing Method

This approach draws its fundamentals from the concept of utilizing two models performing the same task and choosing the best output from the two. It reflects the addition of another model of the same size and type to the model utilized for making the SLM. Both the models parallelly perform to deliver the results of the particular task at hand. Subsequently, the network determines the most relevant

output from both the models and selects the one that is the most contextually relevant, which is attained by employing the scoring function. Figure 4 portrays the methodology of this approach.

The second versions of an SLM is actually dependent upon the type of task, for financial reasoning we found out that by utilizing different parameters, primarily the QLoRa target modules and the training epochs, for the task 1 SLM-v2, we were able to fine-tune one model to concentrate on a particular task with higher success ratio. While the other model did not show any improvement on that particular sub task, it was efficient in solving other variety of questions. The dataset used for both the SLM-1 models were the same to maintain a level playing field for the reasoning task.

While the hyperparameters do play a crucial role, but for task like arithmetic calculation, it was found out that utilizing a dataset consisting higher precision of numbers, for example, a CoT rationale consisting calculation till 3 decimal places instead of 2, made it a much better candidate than finetuning the base model with different parameters. So for the finetuning of task 2 SLM-v2 of arithmetic calculation we utilised the above approach and kept the hyperparameters same as the version 1 model. Hence, it is noteworthy that these can be two approaches while adapting the versions of the models to deliver tasks for this approach.

Building the network: After fine-tuning both the versions of the models, they were loaded from hugging face using the similar strategy as discussed before. Each model has its own LLMChain setup. The chains were combined using LangChain's RunnableParallel module [16]. Following this, a scoring function was attached to the combined chain. This architecture when executed returns a dictionary of the output of both chains, which when further passed to the scoring function, retrieves the final output.

Our proposed approach of parallel processing resonates a competitive network wherein both the models concurrently solve the problem and the best is chosen to be taken forward. It is essential to ensure that the scoring function curated for a task is robust enough to catch all the essential aspects. In our study, we have utilized the same scoring function as employed in the above proposed approach or the threshold technique, as we had the same tasks at hand.

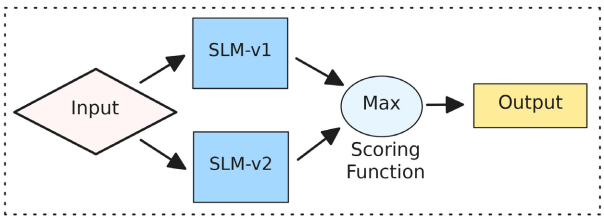

Figure 4. Workflow for parallel-processing approach. Llama2-7B is utilized at all places. Both versions, v1 and v2 execute the task concurrently, and the scoring function selects the most relevant outcome. The models are identical, differing only in their hyperparameters or dataset aspect, akin to the original, and are referred to as two versions.

6. Results

This section addresses the results obtained by carrying out the above proposed methodologies and frameworks. The aim of this study was to lay task specific performance by 1) the baseline model 2) network of multiple SLMs with a supervisor model for auditory purposes 3) network of SLMs with parallel processing utilizing a friend model. We utilized a test dataset consisting 100 testing prompts, keeping them same across all the proposed and baseline methodologies. The final score (accuracy) is calculated as [Total number of Correct Answers/Total Prompts (*i.e.* 100)].

The application of these approaches has been demonstrated on the chosen target applications: financial reasoning and arithmetic calculation of nested mathematical expressions. It is prominent to note that while testing on math prompts, the answers were tallied correctly if they were within 4% of the real answer.

All the proposed approaches involved the usage of smaller language models [in our study they are Llama2-7B] moulded to cater to particular tasks by means of supervised fine-tuning. The extensive discussion of each framework is present in the further subsections.

6.1. Task-Specific Model Performance

6.1.1. Parallel Processing Approach

This proposed pipeline involves the usage of two same SLMs in the chain for each sub-task that needs to be performed. The scoring function applied to the outputs generated by both the models assesses and rates them accordingly. The one that attains the maximum score out of the two is chosen to be taken forward in the chain. In Figure 5 below we display two plots for distinct tasks, financial reasoning and arithmetic calculation respectively. In each plot there are scores for two different versions of a model reflected individually with the scores of the proposed approach. The plot illustrates when individual models are subjected to a network where they are linked to their assisting models, they process concurrently and achieve maximum accuracy.

Figure 5 displays the scores of two versions of Llama2-7B model fine-tuned with a different strategy and an accumulated score utilizing our proposed approach. The max scoring function used, takes into consideration the outputs generated by both the models and finalises one of them using the scoring criteria. The plots portray that the suggested parallel-processing approach achieves the highest accuracy for both tasks, registering 57% and 85%, respectively.

6.1.2. Threshold-Based Approach

The proposed approach of utilizing smaller language models in form of a network with employing the usage of supervisor model revolved around invoking the supervisor wherever (and whenever) necessary. This supervisor SLM acts as a control mechanism that methodically controls the errors encountered in the output of the main SLM. This approach yielded comparative results to employing a Large language Model for a specific task. The discussion of the result is

present in section 6.2 and its visual depiction can be referred in Figure 6.

6.2. Performance Analysis

The plots in Figure 6 demonstrate the scores of all the approaches implemented for each task separately. Llama2-7B and Llama2-13B denotes the fine-tuned version of the base model designed to execute our desired tasks following the same strategy. Threshold-Based and Parallel-Processing Approach comprise of two Llama2-7B models each, as demonstrated in the methodology. The following graphs show that our approach, having similar number of parameters, provides more accuracy than a finetuned Llama2-13B model.

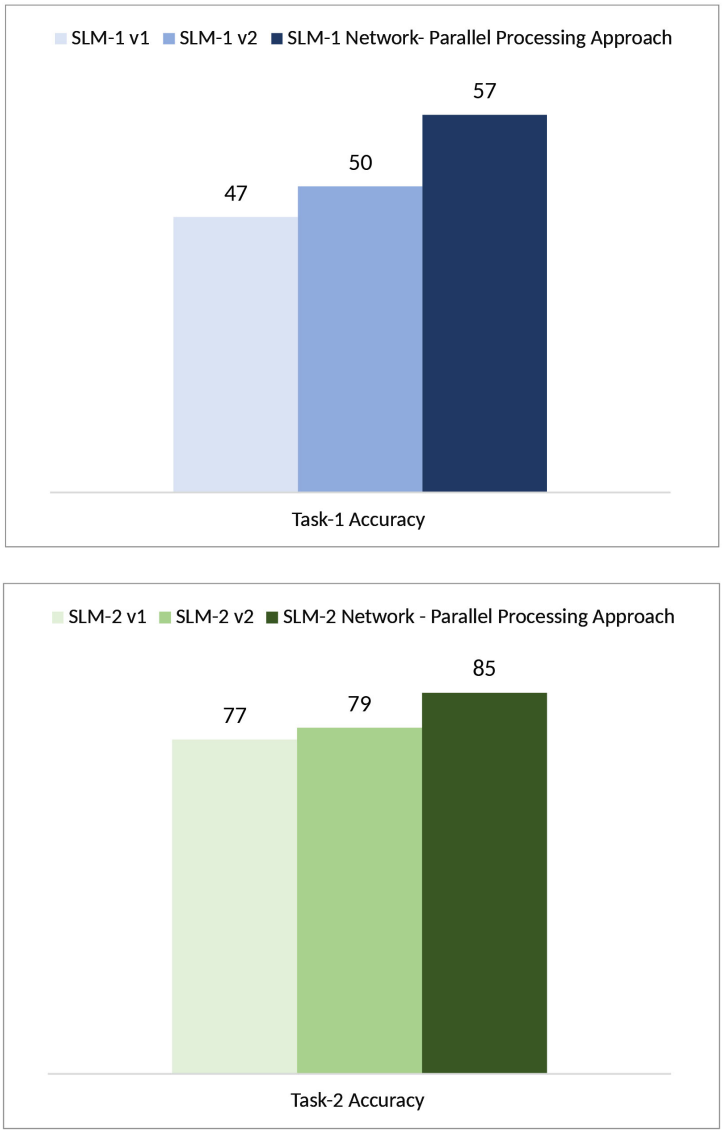

Figure 5. Accuracy analysis for the parallel processing approach. The plot on the left is the depiction of accuracy for the task of financial reasoning, referred as Task 1. While the graph on the right depicts the efficacy for the arithmetic calculation for complex mathematical expressions, mentioned as Task 2. V1 and V2 are the versions of the same SLM employed in the network. Refer section 5.2.2 for more details.

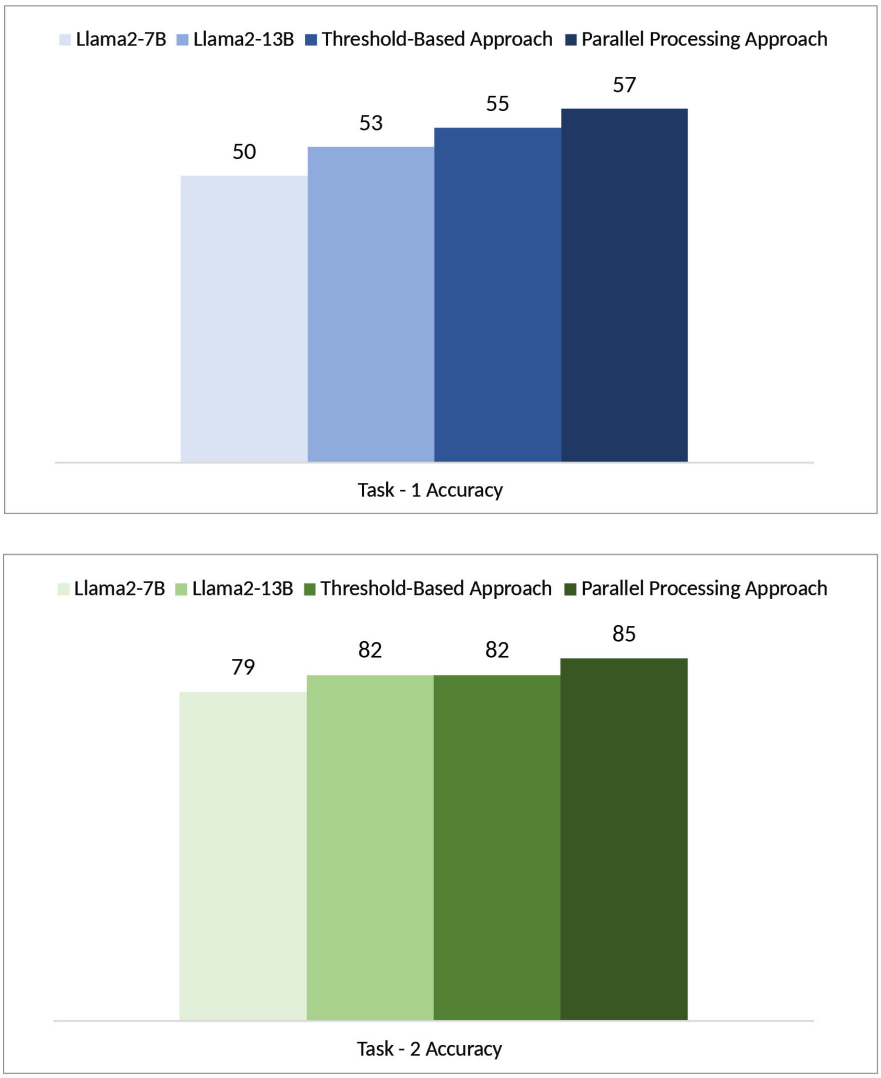

Figure 6. Approach specific accuracy and baseline comparison for the chosen tasks. Task1 refers to financial reasoning while Task2 denotes arithmetic calculation. The SLMs utilized in our proposed approaches is Llama2-7B.

The plots demonstrate that our proposed approaches stand equal or superior to the baselines in both the aspects. Our methodologies substantially improve the accuracy of the Llama2-7B model in both target applications, surpassing Llama2-13B in financial reasoning while achieving parity in arithmetic calculations' capability.

6.3. Cost Analysis

In this section, we provide a comprehensive relative analysis of cost utilization for model inferencing when employing our proposed approaches and also for the established baselines. The following subsections provide a detailed breakdown of the cost analysis for each approach employed in tackling individual tasks. All the costs have been calculated utilizing the averaged length of tokens in the prompts [input + output] across the testing dataset.

• Threshold Based Approach: This proposed approach comprised of a network of SLMs with the working model attached with its supervisor SLM that incrementally corrected its errors to enhance the accuracy of the returned answer. The scoring function employed in the networks acted as a gatekeeper and passed only sub-optimal outcomes further. Hence, looking closely to the methodology it can be understood that the total cost essentially depends on the frequency of invoking of the supervisor model. For the task of financial reasoning, it can be experimentally established that the threshold-based method yielded 11% lower cost when compared with the baseline. While for the task involving arithmetic calculation of mathematical nested expressions we observed a substantial 55% cost reduction.

• Parallel Processing Approach: The proposed approach of parallel processing involved the network of SLMs wherein two models concurrently performed the same operation and the superior result out of the two is chosen as the final output. Upon estimating, it demonstrated that the parallel processing approach resulted in similar inferencing costs (+1.5%) and, therefore, gave no cost advantage for the task of financial reasoning. For arithmetic calculation, it yielded a decrease of 2.5% of the overall cost from the baseline. This minimal reduction in costs compared to the baseline can be explained because the process involves invoking 2 models simultaneously which means the total cost becomes double that of Llama2-7B. This approach is more focused towards leveraging the potential of both the models, and attains a better overall result by ensuring similar inferencing costs.

Therefore, it can be concluded that both of our proposed approaches incur substantially lower (or at par) than the cost of the baseline. While these costs have been established for target financial applications, it can considered as a harbinger of improvements that may be obtained in other specialized areas. Further elaboration can be referenced in **Appendix C**.

7. Limitations

While the study exhibits valuable insights, it also holds certain limitations that warrant careful considerations. The precision of answers may be affected due to the acceptance of rounded numerical value as correct, potentially impacting the overall accuracy and reliability of the results. This approach is consistent along all the accuracy assessments. While we illustrate a sequential pattern of the LLM network, it is noteworthy that a variety of other configurations are possible depending upon the complexity of the task at hand. We also do not include and discuss finance domain-specific LLMs, like BloombergGPT, in our work due to their limited public access.

8. Conclusion

The objective of this study was to conduct an analysis and lay out conclusions for the performance of model networks involving smaller language models and

eventually compare them to the efficacy of a larger LLM, which is chosen to be Llama2-13B. We introduced two new approaches: a threshold based and a parallel processing method. Both these approaches employ a network of SLMs to carve out the results for any complex task with an added supervisor model in the former and an assistive helper model in the latter approach. The introduced approaches employ Llama2-7B models this study differs from other studies wherein the models were kept increasing incrementally in size when arranged in chain. The efficiency of the proposed approaches was evaluated on two complex tasks namely: 1) financial reasoning and 2) arithmetic calculation with nested mathematical expressions, the domains that are largely untouched. Our quantitative analysis illustrated that our proposed networks of SLMs stand above or equal in comparison to the baseline established in terms of accuracy, while also giving better inferencing costs as compared to Llama2-13B when assessed on the test dataset. The proposed network provides the leverage of penetrability giving the option of molding the network wherever required as well as accounts to yield a benefit in terms of developer scalability, which can be attributed to both: low costs and size.

9. Future Work

The efficacy of the proposed approaches in their application to additional specialized applications can be measured in the future using similar techniques. Additionally, we can conduct experiments to test the alternate configurations of the network. We plan to test and compare our proposed network approaches against Language Models consisting of 50B+ parameters.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H. and Smith, N. (2020) Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. https://arxiv.org/abs/2002.06305

[2] Wei. J., Wang, X.Z., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. and Zhou, D. (2023) Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html

[3] LearnPrompting (2023) Chain of Thought Prompting. https://learnprompting.org/docs/intermediate/chain_of_thought

[4] Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C. and Schulman, J. (2021) Training Verifiers to Solve Math Word Problems. https://arxiv.org/abs/2110.14168

[5] Fu, Y., Peng, H., Ou, L.T., Sabharwal, A. and Khot, T. (2023) Specializing Smaller Language Models towards Multi-Step Reasoning. https://arxiv.org/abs/2301.12726

[6] An, S.N., Ma, Z.X., Lin, Z.Q., Zheng, N.N., Lou, J.-G. and Chen, W.Z. (2023) Learning from Mistakes Makes LLM Better Reasoner. https://arxiv.org/abs/2310.20689

[7] Srivastava, A., Rastogi, A., Rao, A., Md Shoeb, A.A., Abid, A., Fisch, A. and Brown, A.R., *et al.* (2023) Beyond the Imitation Game: Quantifying and Extrapolating the Capabilities of Language Models. https://arxiv.org/abs/2206.04615

[8] Liu, T.D. and Low, B.K.H. (2023) Goat: Fine-tuned LLaMA Outperforms GPT-4 on Arithmetic Tasks. https://arxiv.org/abs/2305.14201

[9] Chen, L.J., Zaharia, M. and Zou, J. (2023) FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. https://arxiv.org/abs/2305.05176

[10] Chen, Z.Y., Chen, W.H., Smiley, C., Shah, S., Borova, I., Langdon, D., Moussa, R., Beane, M., Huang, T.-H., Routledge, B. and Wang, W.Y. (2022) FinQA: A Dataset of Numerical Reasoning over Financial Data. https://arxiv.org/abs/2109.00122

[11] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., *et al.* (2023) Llama 2: Open Foundation and Fine-Tuned Chat Models. https://arxiv.org/abs/2307.09288

[12] Dettmers, T., Pagnoni, A., Holtzman, A. and Zettlemoyer, L. (2023) QLoRA: Efficient Finetuning of Quantized LLMs.

[13] Ligntning.ai (2023) Finetuning LLMs with LoRA and QLoRA: Insights from Hundreds of Experiments. https://lightning.ai/pages/community/lora-insights/

[14] LangChain (2023) Chains. https://python.langchain.com/docs/modules/chains/

[15] LangChain (2023) Developer Docs. https://python.langchain.com/docs/get_started/introduction

[16] LangChain (2023) Parallelize Steps. https://python.langchain.com/docs/expression_language/how_to/map

Appendix

This is the appendix of the paper: *Smaller & Smarter*: *Score-Driven Network Chaining of Smaller Language Models*.

A. Financial Reasoning Dataset

The FinQA dataset [10] is a large corpus of data formed by eleven financial experts who collectively worked on analysing the earning reports of S&P500 companies for the years ranging from 1999 to 2019. Based on its approach of retrieval and program generator, the dataset has around 8281 specialised finance question and answer pairs. Apart from being groundbreaking one of its kind, the dataset has the potential to be utilised in training large language models to retrieve high quality question and answering application. It consists of many attributes out of which we will be focusing on a few, namely "question", "gold_inds", "program_re", "explanation" and exe_ans".

B. Arithmetic Calculation Dataset

The aim of crafting this dataset is to finetune the second model of the chain of the LLMs and specialise it for the task of arithmetic calculation of the program_re expressions. The utilization of the fundamental mathematical operations: addition, subtraction, multiplication and division was driven by their frequent occurrence noticed in the financial documents. program_re depicts the logic utilized to solve the reasoning tasks. The dataset consists of two broad categories of the expression: basic and nested. Basic expressions resonate the usage on an individual standalone foundational operator on a set of numerals while the nested expressions are a combination of operations with multiple dimensions ranging from two to five. A breakdown of the split of the dataset can be referred in **Figure A1**.

C. Comprehensive Cost Analysis

This section comprises of an examination of the cost utilization of networks and models, presenting a comparative study between the proposed methodologies and established baselines.

The input and output cost of Llama2-7B are 0.25$/Million tokens each and for Llama2-13B are 0.5$/Million tokens each. The final cost incurred is directly proportional to the length of the prompt which comprises of both the input and the output

The basic cost without any network can be calculated by multiplying the unit cost of inputs with their total length and adding this to the product of the unit cost of outputs and their total length. Hence, this working can be employed while determining the cost for the baseline Llama2-13B. For the threshold based approach the final cost is the addition of the costs for the main model and the supervisor cost which essentially depends on the time of times the supervisor is invoked in the chain. For the task of financial reasoning this frequency was of

around 54 times while for arithmetic calculation it was around 20 times. Finally, for the parallel processing approach, the overall cost is twice the cost incurred for Llama2-7B model as it involves concurrent processing

The plots in Figure A2 depict the cost fluctuations and comparison across all the approaches.

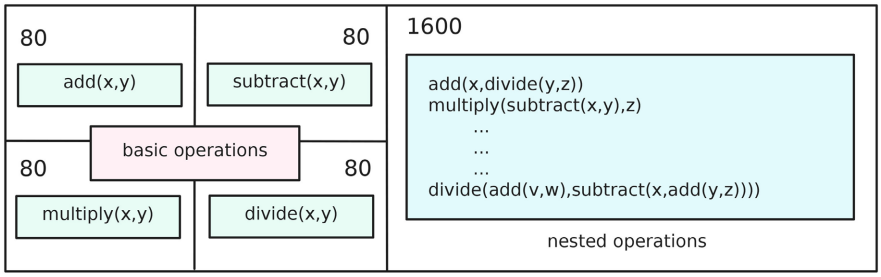

Figure A1. Breakdown of the synthetically generated arithmetic dataset. Each alphabet is employed as a variable, serving to denote numerical values, including both integers and decimals.

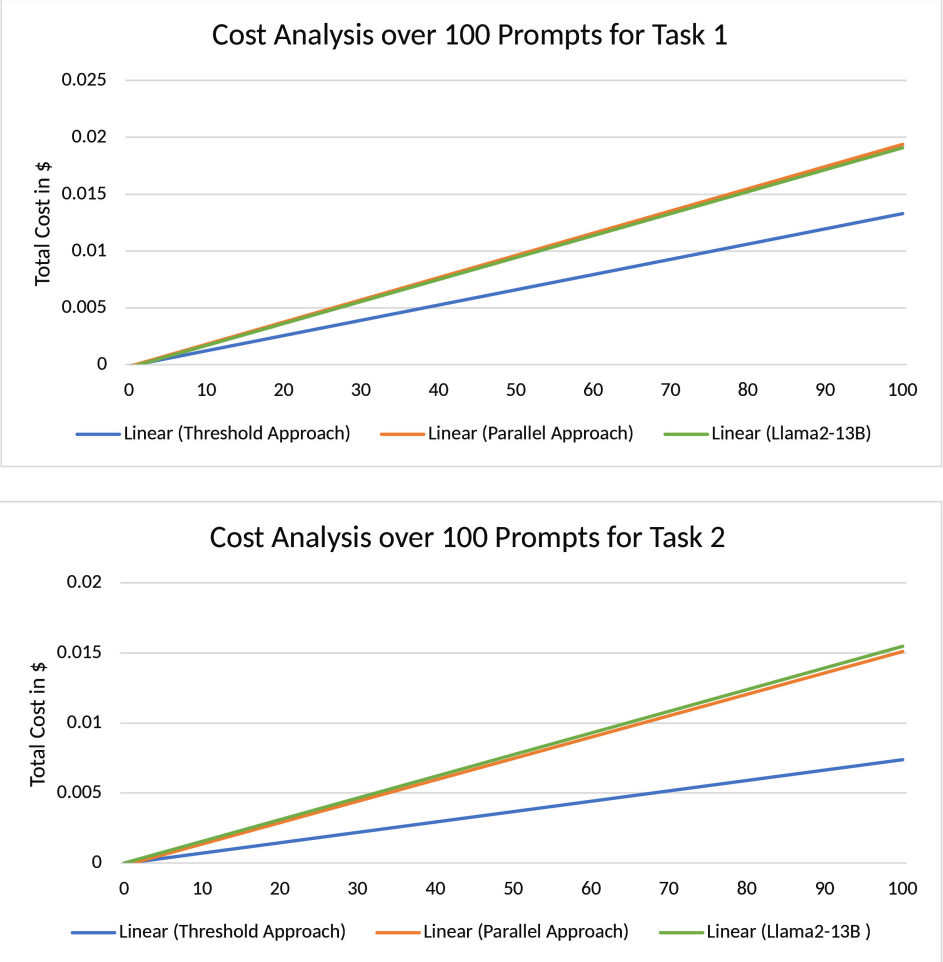

Figure A2. Plots depicting the fluctuations of cost for all the approaches across both the target applications. Task 1 (Top) illustrates financial reasoning while Task 2 (Bottom) depicts the arithmetic calculation application.