Reasoning on a Toaster: Enhancing Small Vision-Language Models via Simplified Chain-of-Thought Fine-tuning

Aniket Suhash Borkar   Anuj Singhal   Apoorva Gupta   Divyansh   Rajeev Kumar   Sandeep Nitharwal

***Abstract*—Vision-Language Models (VLMs) have shown amazing skills in understanding both images and text. However, the really powerful ones are huge and need a lot of computing power, making them unsuitable for everyday devices like smartphones or laptops. Smaller and more efficient VLMs exist, but often struggle with tasks that require complex reasoning, such as answering questions about charts, maths, planning,etc. This paper explores how to improve the reasoning abilities of these small VLMs without making them too big for edge devices. We focus on the ChartQA benchmark, a task that requires both visual perception and logical steps. Our main approach uses Supervised Fine-Tuning (SFT) on reasoning steps (Chain-of-Thought, or CoT) generated by a large teacher model (Google Gemini). Critically, we found that *simplifying* these CoT traces was key to effective learning for the smaller models. We also investigate the surprising effectiveness of a simple prompt modification (”*Let's think step by step*”) during inference. Although we explored advanced Reinforcement Learning (RL) methods like DPO and GRPO, we found them challenging to apply effectively in this setting. Our results, validated through automated accuracy metrics and human preference ELO scores, show that SFT with simplified CoT, especially combined with prompt engineering, significantly improves the reasoning performance of small VLMs (SmolVLM-256M and 500M) on ChartQA, making them smarter without needing a supercomputer.**

***Index Terms*—Vision-Language Models, Edge Computing, Chain-of-Thought, Supervised Fine-Tuning, Visual Question Answering, ChartQA, Prompt Engineering, Small Models**

I. Introduction

Vision-Language Models (VLMs) are pretty cool AI systems that can understand the world using both pictures and words [1]. They can do things like write captions for images, answer questions about what's in a picture (Visual Question Answering, or VQA), or even generate images from text descriptions. Big VLMs, like GPT-4V or Gemini, are incredibly smart, but they need powerful computers to run. This makes it hard to use them directly on devices most people have, like phones or personal laptops – devices with limited processing power and memory, which we jokingly call ”toasters”.

While there are smaller VLMs designed to be lightweight [2], they often lack the sophisticated reasoning abilities of their larger cousins. This becomes a problem for tasks that require not just seeing things in an image, but also thinking logically about them. A good example is understanding charts and graphs. Answering a question like ”Which product had the biggest sales increase between 2022 and 2023?” based on a bar chart requires extracting specific numbers and then performing a calculation – a multi-step reasoning process.

Researchers have worked on making big models smaller using techniques like distillation or quantization [3]. However, our goal wasn't just to shrink a model, but to specifically improve the reasoning capability of models that are already small. How can we make these ”toaster-friendly” VLMs better thinkers, especially for chart-based VQA?

Our main idea was inspired by how large language models (LLMs) learn to reason using Chain-of-Thought (CoT) [4]. We wondered: could we teach a small VLM to reason by showing it examples of reasoning steps generated by a much smarter, larger model? We decided to use Google's Gemini model as our ”teacher” to generate CoT reasoning traces for the ChartQA dataset [5]. We then used Supervised Fine-Tuning (SFT) to train small VLMs (specifically, SmolVLM-256M and SmolVLM-500M [2]) on these reasoning examples.

Interestingly, our initial attempts with detailed reasoning traces didn't work well. We found that simplifying the reasoning steps was crucial. We also experimented with Reinforcement Learning (RL) techniques like Direct Preference Optimization (DPO) [6] and Group Relative Policy Optimization (GRPO) [7] which have shown promise in aligning LLMs, but found them difficult to adapt effectively for boosting reasoning in our small VLMs. Finally, we stumbled upon a surprisingly simple trick: just adding ”Let's think step by step” to the input question significantly improved reasoning performance.

This paper details our journey:

- Section II discusses related work in VLMs, model compression, and reasoning enhancement.
- Section III explains our methodology, focusing on simplified CoT generation and SFT.
- Section IV describes our experimental setup, including the datasets and models.
- Section V presents our results using both automated accuracy and human evaluation.
- Section VI discusses our findings, limitations, and ideas for future work.
- Section VII concludes the paper.

II. Related Work

Making VLMs smaller and smarter involves several research areas.

VLM Architectures: Most VLMs combine a vision encoder (to process the image, often a Vision Transformer (ViT) [8] or Convolutional Neural Network (CNN)) and a language model (to process text and generate responses, usually a Transformer variant). The key challenge is effectively fusing the information from both modalities [1].

Small VLMs for Edge Devices: Recognizing the need for efficiency, researchers have developed smaller VLMs. Examples include the SmolVLM family (256M, 500M parameters) [2] and models like MiniCPM-V [10]. While efficient, these models often lag behind larger ones in complex reasoning tasks involving math or multi-step logic, as noted in their own documentation and our observations (Slide 5).

Model Compression Techniques: Standard ways to shrink models include:

- **Knowledge Distillation:** Training a smaller ”student” model to mimic the output (response-based) or internal representations (feature-based) of a larger ”teacher” model. Often uses Kullback–Leibler (KL) divergence to match output distributions [9].
- **Pruning:** Removing less important weights or connections in the model.
- **Quantization:** Reducing the precision of the model’s weights (e.g., from 32-bit floats to 8-bit integers), sometimes using Quantization-Aware Training (QAT).
- **Low-Rank Factorization:** Approximating large weight matrices with smaller ones, for example using Singular Value Decomposition (SVD).

These techniques focus primarily on reducing size while preserving existing performance, rather than explicitly teaching new reasoning skills.

Improving Reasoning in LLMs: Significant progress has been made in enhancing LLM reasoning:

- **Fine-tuning on Specialized Data:** Training models on datasets focused on specific reasoning types, like math (e.g., DeepSeek-Math [11]).
- **Chain-of-Thought (CoT) Prompting/Fine-tuning:** Encouraging models to output intermediate reasoning steps before the final answer [4]. This can be done via prompting (”Let’s think step by step”) or by fine-tuning on CoT data.
- **Advanced Reasoning Strategies:** Methods like Tree of Thoughts (ToT) [12] explore multiple reasoning paths. Inference-time search and verification (like in Gemini [13]) generate multiple answers and pick the best one.
- **Reinforcement Learning from Human Feedback (RLHF):** Using human preferences to fine-tune models. DPO [6] learns directly from preference pairs (good vs. bad answer). GRPO [7] extends this to handle ranked lists of answers, potentially offering finer control. DeepSeek-R1 [14] used GRPO effectively for LLMs.

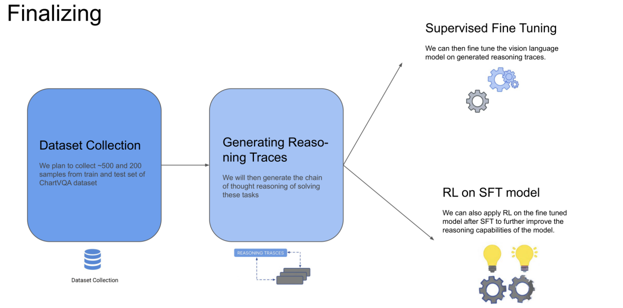
Improving Reasoning in VLMs: Applying these reasoning techniques to VLMs is an active research area.

- A recent work [15] used CoT data generated by GPT-4o to fine-tune VLMs using SFT, followed by DPO. Their focus was on larger VLMs (e.g., 7B parameters). Our work differs by targeting much smaller models (256M/500M), using Gemini as the teacher, and finding simplified CoT crucial. We also faced more challenges making DPO effective.
- Another paper, R1-VL [16], proposed StepGRPO, a specialized RL framework using rule-based rewards for step-wise feedback in VLM reasoning tasks, particularly math-related ones. This inspired our attempt to use GRPO, but we encountered practical hurdles implementing it for general chart VQA with our small models.

Challenges and Our Approach: Existing small VLMs lack strong reasoning. Directly applying techniques from large LLMs, like complex CoT or RLHF methods (DPO/GRPO), proved difficult for teaching new reasoning skills to very small VLMs, especially given resource constraints and potential implementation complexities for vision tasks. Our work focuses on a pragmatic approach: using SFT with simplified CoT traces distilled from a large model (Gemini) as an effective and resource-friendly way to boost reasoning in toaster-sized VLMs, complemented by the simple power of prompt engineering.

III. Proposed Methodology

Our goal was to make small VLMs better at reasoning, specifically for answering questions about charts, so they could potentially run on edge devices. Our approach involved distilling reasoning steps from a large model and using them to fine-tune smaller ones.



Finalizing

Dataset Collection

We plan to collect ~500 and 200 samples from train and test set of ChartVQA dataset

Generating Reasoning Traces

We will then generate the chain of thought reasoning of solving these tasks

Supervised Fine Tuning

We can then fine tune the vision language model on generated reasoning traces.

RL on SFT model

We can also apply RL on the fine tuned model after SFT to further improve the reasoning capabilities of the model.

Fig. 1. Overall pipeline: Generate simplified CoT traces from a large model (Gemini) using ChartQA samples, then use this data for Supervised Fine-Tuning (SFT) of small base VLMs (SmolVLM). Optional RL phase was explored but faced challenges.

Figure 1 shows our overall pipeline.

A. Base Vision-Language Models

We chose the SmolVLM family [2] as our base models because they are specifically designed for efficiency. We experimented with two sizes:

- **SmolVLM-256M:** A smaller, ultra-efficient model.

- **SmolVLM-500M:** A slightly larger model offering better performance.

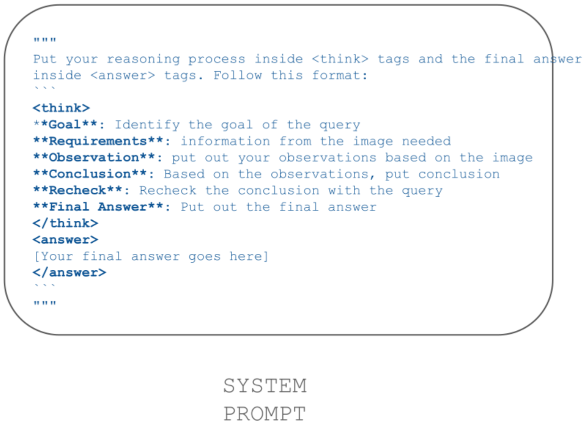
These models consist of a vision encoder based on SigLIP [17] and a language decoder based on Phi-1.5 [18], connected by a simple projection layer.

B. Chain-of-Thought (CoT) Data Generation

We needed reasoning examples to teach our small VLMs. We used a large, capable model, Google Gemini (specifically, ‘gemini-2.0-flash-thinking‘ accessed via API), as a ”teacher” to generate Chain-of-Thought (CoT) traces for questions from the ChartQA dataset [5].

Phase 1: Structured CoT (Initial Attempt): Initially, we prompted Gemini to generate detailed, structured reasoning steps, including identifying the goal, required information, observations from the image, conclusions, and checks (See Figure 2). We generated about 1,000 such examples. However, when we fine-tuned SmolVLM on this data, performance decreased significantly (accuracy dropped from 45% to 17% on a test set, see Figure 3). We suspected two reasons:

1) Learning complex structures might be too hard for small models.
2) The long CoT traces might exceed the effective context length SmolVLM could handle, leading to confusion or ”hallucination” (making things up).

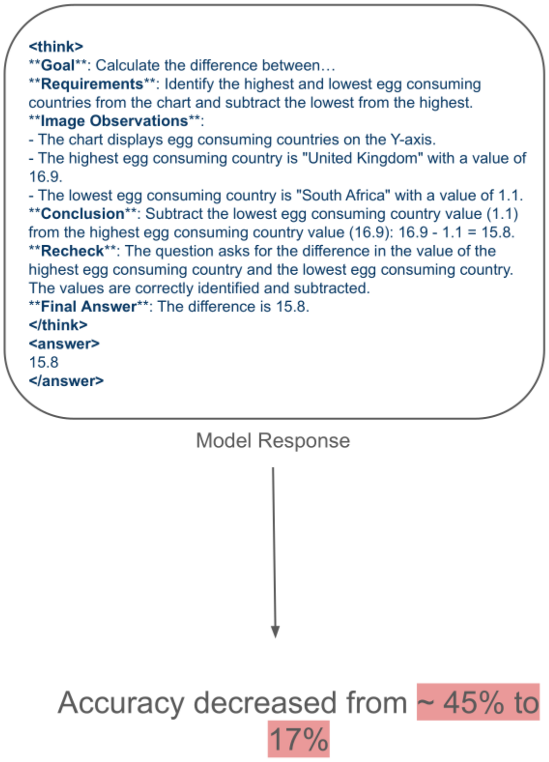


~~~
"""
Put your reasoning process inside <think> tags and the final answer
inside <answer> tags. Follow this format:
```
<think>
**Goal**: Identify the goal of the query
**Requirements**: information from the image needed
**Observation**: put out your observations based on the image
**Conclusion**: Based on the observations, put conclusion
**Recheck**: Recheck the conclusion with the query
**Final Answer**: Put out the final answer
</think>
<answer>
[Your final answer goes here]
</answer>
```
"""
~~~

Fig. 2. Phase 1: Generating structured, detailed CoT traces using Gemini. This approach proved ineffective for fine-tuning small VLMs.

Phase 2: Simplified CoT (Successful Approach): Based on the failure of Phase 1, we drastically simplified the prompt for Gemini (See Figure 4). We asked for a very concise thinking process (just 2-3 lines) before the final answer. The format was simply: “‘ ¡think¿ [Short reasoning steps here] ¡/think¿ ¡answer¿ [Concise final answer here] ¡/answer¿ “‘ This simpler format proved much more effective. We scaled this process up, generating two datasets using this simplified approach:

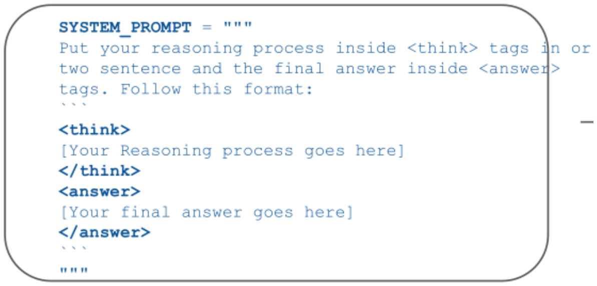
- **smolcot-3k:** Approximately 3,000 samples.
- **smolcot-7k:** Approximately 7,000-8,000 samples.



```
<think>
**Goal**: Calculate the difference between…
**Requirements**: Identify the highest and lowest egg consuming countries from the chart and subtract the lowest from the highest.
**Image Observations**:
- The chart displays egg consuming countries on the Y-axis.
- The highest egg consuming country is "United Kingdom" with a value of 16.9.
- The lowest egg consuming country is "South Africa" with a value of 1.1.
**Conclusion**: Subtract the lowest egg consuming country value (1.1) from the highest egg consuming country value (16.9): 16.9 - 1.1 = 15.8.
**Recheck**: The question asks for the difference in the value of the highest egg consuming country and the lowest egg consuming country. The values are correctly identified and subtracted.
**Final Answer**: The difference is 15.8.
</think>
<answer>
15.8
</answer>
```

Fig. 3. Phase 1 Results: Training loss decreased, but actual task accuracy plummeted when using structured CoT, indicating the model wasn’t learning the reasoning task effectively.

These datasets formed the basis for our successful fine-tuning experiments.



~~~
SYSTEM_PROMPT = """
Put your reasoning process inside <think> tags in or
two sentence and the final answer inside <answer>
tags. Follow this format:
```
<think>
[Your Reasoning process goes here]
</think>
<answer>
[Your final answer goes here]
</answer>
```
"""
~~~

Fig. 4. Phase 2: Generating simplified, concise CoT traces (2-3 lines) using a revised prompt. This data was effective for SFT.

C. Supervised Fine-Tuning (SFT)

The core of our method was SFT. We trained the SmolVLM-256M and SmolVLM-500M models on our ‘smolcot-3k‘ and ‘smolcot-7k‘ datasets. The models were trained to predict the entire text string containing both the ‘¡think¿‘ process and the final '¡answer¿', given the chart image and the user's question. We used the standard SFT setup from the Hugging Face TRL (Transformer Reinforcement Learning) library [19], which essentially performs next-token prediction loss minimization on the target text.

D. Reinforcement Learning (RL) Exploration

Inspired by recent works [15], [16], we initially planned to apply RL after SFT to further refine the models' reasoning.

- **DPO Attempt:** We experimented with Direct Preference Optimization (DPO) [6]. DPO learns from pairs of preferred and dispreferred responses. We tried using the Gemini-generated CoT+answer as the "preferred" response and the SFT model's own generated response as the "dispreferred" one. However, we found DPO difficult to train effectively for teaching complex reasoning, especially with the longer structured CoT traces. It seemed better suited for aligning outputs to a desired style rather than teaching fundamentally new reasoning steps to small models. Memory usage was also a significant challenge.
- **GRPO Consideration:** We considered Group Relative Policy Optimization (GRPO) [7], which handles ranked responses and could potentially offer finer control. However, GRPO involves multiple training stages (e.g., training a reward model implicitly or explicitly) and we couldn't find readily available implementations that integrated well with vision models and our specific setup within popular libraries like TRL or Unsloth [20] at the time.

Due to these practical difficulties and resource limitations, we focused our efforts on optimizing the SFT stage, which yielded significant improvements on its own.

E. Prompt Engineering during Inference

Based on the effectiveness of CoT prompting in LLMs [21], we tested a simple modification during inference (when evaluating the models). We appended the phrase "*Let's think step by step.*" to the end of the user's question before feeding it to the VLM. This was evaluated as a separate condition for both the baseline models and our SFT-finetuned models.

IV. Experimental setup

A. Hardware and Software

We conducted our experiments primarily using NVIDIA A40 GPUs. We utilized standard deep learning frameworks and libraries, including:

- PyTorch [22] for model implementation and training.
- Hugging Face 'transformers' [23] for accessing pre-trained models (SmolVLM) and processors.
- Hugging Face 'datasets' [24] for data handling.
- Hugging Face 'trl' [19] for the Supervised Fine-Tuning (SFT) process.
- 'accelerate' [25] for efficient multi-GPU training or inference, if needed.
- Unsloth [20] library utilities were considered for potential memory optimization during RL exploration, although SFT was the main focus.

B. Dataset

Our work revolved around the **ChartQA** dataset [5]. This benchmark is designed for visual reasoning on charts and graphs. It contains a variety of chart types (bar, line, pie, etc.) sourced from real-world contexts, paired with natural language questions that often require multiple reasoning steps (e.g., finding values, comparing, calculating differences or sums).

Data Generation: As described in Section III-B, we used questions and images from ChartQA as input to the Gemini API ('gemini-2.0-flash-thinking') to generate CoT traces.

- **Training Sets:** We created two main training datasets using the simplified CoT format: 'smolcot-3k' (approx. 3,000 samples) and 'smolcot-7k' (approx. 7,000-8,000 samples). These contained the image, question, the simplified Gemini-generated CoT, and the final answer.
- **Test Set:** We used a fixed held-out set of 200 samples from ChartQA (that were not used for generating training CoT) for evaluating all model variants. This test set included the image, question, and the ground-truth final answer.

C. Models Evaluated

We evaluated the following model configurations:

- **Baseline Models:**
- '256M-base': Original SmolVLM-256M-Instruct model.
- '500M-base': Original SmolVLM-500M-Instruct model.
- **Baseline + Prompt Engineering:**
- '256M-base+step': Baseline 256M model with "Let's think step by step" added to the test prompts.
- '500M-base+step': Baseline 500M model with the "+step" prompt.
- **SFT Models:**
- '256M-3k': '256M-base' fine-tuned on 'smolcot-3k'.
- '256M-7k': '256M-base' fine-tuned on 'smolcot-7k'.
- '500M-3k': '500M-base' fine-tuned on 'smolcot-3k'.
- '500M-7k': '500M-base' fine-tuned on 'smolcot-7k'.
- **SFT + Prompt Engineering:**
- '256M-3k+step': '256M-3k' with the "+step" prompt.
- '256M-7k+step': '256M-7k' with the "+step" prompt.
- '500M-3k+step': '500M-3k' with the "+step" prompt.
- '500M-7k+step': '500M-7k' with the "+step" prompt.

D. Training Details

For the SFT phase, we used the following typical hyperparameters (referencing the provided script):

- **Optimizer:** Paged AdamW (8-bit) [26].
- **Learning Rate:** 3e-5 (found to work better than the initial 3e-4 during experimentation).
- **Batch Size:** Effective batch size of 32 (per-device batch size 1, gradient accumulation 32).
- **Epochs:** 1 epoch over the respective training dataset ('smolcot-3k' or 'smolcot-7k').

- **Precision:** Mixed precision with ‘bfloat16‘ (when supported).
- **Scheduler:** Linear learning rate decay.
- **Max Sequence Length:** 2048 tokens.
- **Gradient Checkpointing:** Enabled to save memory.

We used the ‘SFTTrainer‘ from the ‘trl‘ library with a custom data collator (‘UnslothVisionDataCollator‘ or similar standard vision collator) to handle the image and text inputs.

V. Results

We evaluated our models using two methods: automated accuracy checking and human preference evaluation.

A. Evaluation Metrics

Automated Accuracy: We developed a Python script to compare the model’s generated answer (extracted from the ‘¡answer¿‘ tag if present, or the full output otherwise) with the ground-truth answer from our test set. The script performed several normalization steps:

- Extracted numbers using regular expressions.
- Converted text numbers to numeric form (e.g., ”seven” to 7) using the ‘word2num‘ library.
- Ignored punctuation and converted text to lowercase.
- Checked for the presence of the ground-truth answer within the model’s output.

This gave us a quantitative measure of correctness. However, this metric has limitations: it can’t handle synonyms well (e.g., ”correct” vs. ”true”), struggles with different numerical formats (e.g., fractions vs. decimals unless explicitly converted), and doesn’t capture the quality of reasoning if the final answer is correct by chance or incorrect despite good steps.

Human Evaluation (ELO Rating): To get a better sense of overall quality and reasoning coherence, we used pairwise comparison inspired by Chatbot Arena [27]. We created a simple tool that showed evaluators (our team members and friends) the chart image, the question, and the outputs from two different model configurations, presented anonymously and in random order. The evaluator chose which response was better (Model A, Model B) or if they were tied/equally bad. We collected hundreds of these comparisons across all model pairs. These win/loss/tie results were then used to calculate an ELO rating [28] for each model configuration, providing a relative ranking based on human judgment. Higher ELO scores indicate better performance according to human preference.

B. Automated Accuracy Results

Figures 5 and 6 show the automated accuracy scores on our test set for the 256M and 500M models, respectively.

Key Observations from Automated Accuracy:

- **SFT Improvement:** For both 256M and 500M models, SFT on simplified CoT data (‘-3k‘ and ‘-7k‘) generally improved accuracy over the respective baselines (‘-base‘).
- **Scaling Data Helps:** Fine-tuning on the larger ‘smolcot-7k‘ dataset mostly resulted in higher accuracy than fine-tuning on ‘smolcot-3k‘.

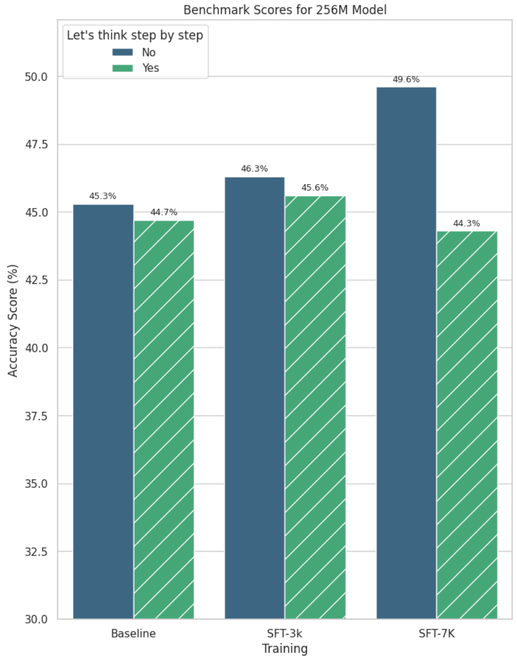


Fig. 5. Automated Accuracy Scores for SmolVLM-256M Variants on ChartQA Test Set. ”Yes” indicates the ”+step” prompt was used.

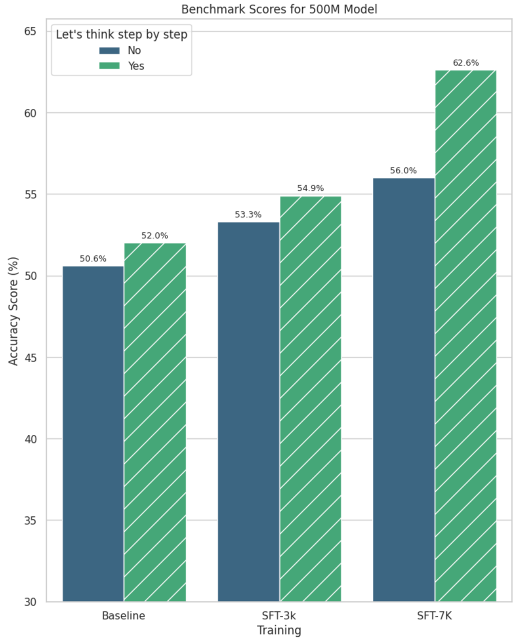


Fig. 6. Automated Accuracy Scores for SmolVLM-500M Variants on ChartQA Test Set. ”Yes” indicates the ”+step” prompt was used.

- **Model Size Matters:** The 500M models consistently achieved higher accuracy than their 256M counterparts across all conditions.
- **Prompt Engineering Boost:** Adding "Let's think step by step" ('+step', the green bars) significantly increased accuracy for all models, including the baselines. This simple prompt modification often provided a substantial performance lift, sometimes rivaling or exceeding the gains from SFT alone. For instance, '500M-base+step' (52.0%) outperformed '500M-base' (50.6%), and '500M-7k+step' (62.6%) showed the highest accuracy overall.
- **Interaction Anomaly (256M):** Interestingly, for the 256M model, the '+step' prompt seemed less effective on the SFT models ('256M-3k+step', '256M-7k+step') compared to the baseline ('256M-base+step'), suggesting the SFT might have already incorporated some step-by-step behavior implicitly, or perhaps there's noise in the evaluation.

C. Human Evaluation Results (ELO Scores)

Figure 7 presents the ELO ratings calculated from our pairwise human evaluations.

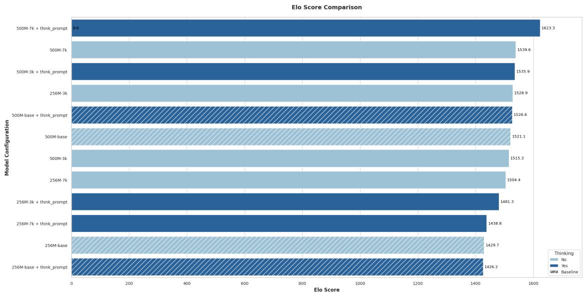


Fig. 7. ELO Score Comparison of SmolVLM Variants based on Human Preference Evaluation. Hatched bars indicate the "+step" prompt was used during inference.

Key Observations from ELO Scores:

- **Clear Ranking:** The ELO scores provide a clear relative ranking of the models based on perceived quality.
- **Top Performers:** The '500M-7k+think_prompt' configuration achieved the highest ELO score (1623.3), aligning with the automated accuracy results showing it as the best model. Generally, models fine-tuned on more data (7k) and using the "+step" prompt ranked higher.
- **SFT and Prompting Benefit:** Both SFT and the prompt "+ step" generally led to higher ELO scores compared to baselines without prompt. For example, '500M-base+think_prompt' (ELO 1526.6) is rated higher than '500M-base' (ELO 1521.1). Similarly, '500M-7k' (ELO 1539.6) is much higher than '500M-base'.
- **The 256M-3k Surprise:** The '256M-3k' model achieved a surprisingly high ELO score (1528.9), outperforming several other configurations, including the '500M-base' and '500M-3k'. This suggests that for its size, the 3k fine-tuning hit a sweet spot that humans perceived favorably, even if its raw accuracy wasn't the absolute highest. It even slightly outperformed the '500M-base+think_prompt'.
- **Baseline Ranking:** The baseline models ('256M-base', '500M-base') without the "+step" prompt received the lowest ELO scores, confirming their weaker reasoning capabilities.

These results collectively show that our approach of fine-tuning small VLMs on simplified CoT, especially when combined with the "Let's think step by step" prompt, leads to noticeable improvements in reasoning performance on the ChartQA task, as measured by both automated metrics and human judgment.

D. Cost Estimation

A key motivation for this project was exploring cost-effective ways to improve small VLM reasoning for potential deployment on resources-constrained devices. Our approach emphasized efficiency. The primary costs incurred were GPU compute time for SFT and API calls for generating the CoT data.

- **GPU Usage:** Fine-tuning the SmolVLM models was performed on NVIDIA A40 GPUs. Based on typical cloud instance pricing (approximately \$0.4 / hour) and the total training time between experiments, the estimated cost of GPU was around \$35.2.
- **API Calls:** Generating the simplified CoT data involved calls to the Google Gemini API. Based on the price of input / output tokens (approximately \$0.1 /\$0.4 per million tokens, respectively) and the volume of data generated (up to 8k samples with prompts and responses), the estimated API cost was approximately \$20.48.

Therefore, the core experimental phase involving data generation and model fine-tuning was achieved with a relatively modest budget (under \$60), demonstrating the cost-effectiveness of using API-based distillation and efficient SFT to improve small VLMs.

VI. Discussion and Future work

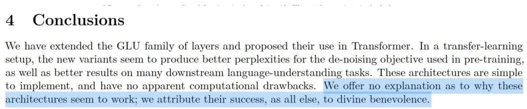
Our experiments highlight several interesting points about enhancing reasoning in small VLMs.

Simplified CoT is Key for Small Models: Our most significant finding is the importance of simplifying the Chain-of-Thought traces for fine-tuning small models like SmolVLM. The initial attempt with detailed, structured CoT failed, likely because the models (256M/500M parameters) lack the capacity to effectively learn complex structures or handle the resulting long context lengths. The simplified 2-3 line reasoning steps provided a digestible format that allowed the models to successfully learn the reasoning patterns via SFT, leading to improved performance on ChartQA. This suggests that when distilling knowledge for small models, adapting the complexity of the distilled information to the student's capacity is crucial.

The Surprising Power of Prompting: The "Let's think step by step" prompt addition proved remarkably effective, boosting performance across the board, sometimes even more than SFT itself (comparing ‘base+step‘ vs. ‘SFT-3k‘ in some cases). This aligns with findings in LLMs [21] and suggests that even small VLMs possess latent reasoning capabilities that can be unlocked with simple prompting cues. Combining SFT with this prompt generally yielded the best results, indicating these two approaches are complementary.

Challenges with RL for Reasoning Enhancement: While RLHF methods like DPO and GRPO are powerful for aligning LLMs and have been explored for VLMs [15], [16], we encountered significant practical hurdles applying them to teach new reasoning skills to our very small VLMs. DPO struggled to make substantial changes beyond the SFT baseline and hit memory limits. GRPO seemed promising theoretically but lacked accessible implementations suited for our vision-based task and resource constraints. This suggests that using RL to instill complex reasoning might require larger model capacity or more specialized RL frameworks than what we could readily employ.

The 256M-3k Anomaly: The high ELO score of the ‘256M-3k‘ model, despite its small size and less training data compared to the 7k variants, is intriguing. It outperformed several larger or more extensively trained models in human evaluations. While automated accuracy didn’t fully reflect this, it suggests this configuration might produce answers that are perceived as more coherent or reliable by humans, even if not always perfectly accurate by script standards. As noted humorously during our work (referencing Noam Shazeer’s comment on GLU variants [29], see Figure 8), sometimes attributing success requires acknowledging a bit of unexpected positive outcome - perhaps the 3k dataset provided just the right amount of focused signal for the 256M model’s capacity.



4 Conclusions

We have extended the GLU family of layers and proposed their use in Transformer. In a transfer-learning setup, the new variants seem to produce better perplexities for the de-noising objective used in pre-training, as well as better results on many downstream language-understanding tasks. These architectures are simple to implement, and have no apparent computational drawbacks. We offer no explanation as to why these architectures seem to work; we attribute their success, as all else, to divine benevolence.

Fig. 8. The surprising performance of 256M-3k sometimes felt hard to explain, echoing sentiments about unexpected success in model architectures.

Limitations: Our study has limitations. The evaluation was primarily on the ChartQA dataset, so generalization to other reasoning tasks is unknown. Our automated accuracy metric has known shortcomings. While human evaluation provides valuable insights, it’s based on a limited number of evaluators and comparisons. Our exploration of RL methods was constrained by time and resources.

Future Work: There are several exciting directions for future research:

- **Broader Tasks:** Apply the simplified CoT SFT approach to other reasoning-intensive VLM tasks like mathematical VQA or visual commonsense reasoning.
- **Efficient RL:** Develop or adapt RL techniques specifically designed for enhancing reasoning in small VLMs, perhaps focusing on sample efficiency or simpler reward structures.
- **Multi-Stage Training:** Explore pipelines where one small model generates the CoT and another generates the final answer based on the CoT (as suggested in slide 40).
- **Hybrid Approaches:** Combine simplified CoT SFT with other model compression techniques like quantization or pruning for further efficiency gains.
- **Data Scaling and Diversity:** Generate even larger simplified CoT datasets covering a wider range of reasoning types.
- **Investigate Prompting Further:** Explore different prompt variations to unlock reasoning capabilities.

VII. Conclusion

Getting powerful AI like VLMs to run effectively on everyday devices remains a significant challenge, especially for tasks requiring complex reasoning. In this work, we tackled the problem of enhancing the reasoning abilities of small, edge-friendly VLMs (SmolVLM-256M and 500M) for the task of visual question answering on charts (ChartQA). We demonstrated that Supervised Fine-Tuning using simplified Chain-of-Thought reasoning traces, distilled from a large teacher model (Gemini), is a practical and effective method. Simplifying the CoT was crucial for these smaller models. Furthermore, we highlighted the surprising effectiveness of a simple inference-time prompt modification (”Let’s think step by step”), which provided substantial gains both independently and in conjunction with SFT. While advanced RL techniques like DPO and GRPO showed theoretical promise, they presented practical challenges in our specific context. Our combined approach of simplified CoT SFT and prompt engineering led to significant improvements in both automated accuracy and human-evaluated quality, making small VLMs considerably ”smarter” without demanding excessive computational resources. This work offers a viable path towards deploying more capable reasoning VLMs on ”toasters”.

References

[1] Radford, A., et al. ”Learning transferable visual models from natural language supervision.” ICML 2021.

[2] Abbas, A., et al. ”SmolLM and SmolVLM: Parameter-Efficient Multi-Modal Knowledge Distillation From Sparse Mixture of Experts,” arXiv preprint arXiv:2406.05578, 2024.

[3] Placeholder for a model compression survey. Example: Cheng, Y., et al. ”A Survey of Model Compression and Acceleration for Deep Neural Networks,” IEEE Signal Processing Magazine, 2018.

[4] Wei, J., et al. ”Chain-of-thought prompting elicits reasoning in large language models,” NeurIPS 2022.

[5] Masry, A., et al. ”ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning,” Findings of ACL 2022.

[6] Rafailov, R., et al. ”Direct Preference Optimization: Your Language Model is Secretly a Reward Model,” NeurIPS 2023.

[7] Placeholder for the main GRPO paper. Example: Liu, Z., et al. ”Group Relative Policy Optimization,” arXiv preprint arXiv:2405.14370, 2024.

[8] Dosovitskiy, A., et al. ”An image is worth 16x16 words: Transformers for image recognition at scale,” ICLR 2021.

[9] Gheini, M. M., et al. ”Analysis of Knowledge Distillation Techniques on Image Captioning Models,” CEUR Workshop Proceedings, Vol-3283, Paper 126, 2022. (Note: As per discussion, use cautiously or find a better reference).

[10] Placeholder for MiniCPM-V paper. Example: Ren, S., et al. ”MiniCPM-V: A GPT-4V Level MLLM on Your Phone,” arXiv preprint arXiv:2404.16917, 2024.

[11] Shao, Z., et al. ”DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models,” arXiv preprint arXiv:2402.03300, 2024.

[12] Yao, S., et al. ”Tree of Thoughts: Deliberate Problem Solving with Large Language Models,” NeurIPS 2023.

[13] Placeholder for Gemini inference-time search/verification reference if available, or cite the main Gemini paper. Example: Pichai, S., et al. ”Introducing Gemini: our largest and most capable AI model,” Google AI Blog, 2023.

[14] Placeholder for DeepSeek-R1 paper. Example: Bi, W., et al. ”DeepSeek-Coder-V1.5 Technical Report,” arXiv preprint arXiv:2406.11917, 2024. (Assuming R1 is related).

[15] Zhu, D., et al. ”Improving Vision-Language Models' Chain-of-Thought Reasoning via Instruction Tuning,” arXiv preprint arXiv:2401.16198, 2024. (Note: Actual citation might be different, this is based on slide reference).

[16] Zhang, Z., et al. ”R1-VL: Learning to Reason with Multimodal Large Language Models via Step-wise Group Relative Policy Optimization,” arXiv preprint arXiv:2403.14597, 2024. (Note: Actual citation might be different, this is based on slide reference).

[17] Zhai, X., et al. ”Sigmoid Loss for Language Image Pre-Training,” ICCV 2023.

[18] Placeholder for Phi-1.5 paper. Example: Gunasekar, S., et al. ”Textbooks Are All You Need,” arXiv preprint arXiv:2306.11644, 2023.

[19] von Werra, L., et al. ”TRL: Transformer Reinforcement Learning,” GitHub Repository, Hugging Face, 2020. https://github.com/huggingface/trl

[20] Placeholder for Unsloth citation if available, or just mention the library. https://github.com/unslothai/unsloth

[21] Kojima, T., et al. ”Large Language Models are Zero-Shot Reasoners,” NeurIPS 2022.

[22] Paszke, A., et al. ”PyTorch: An Imperative Style, High-Performance Deep Learning Library,” NeurIPS 2019.

[23] Wolf, T., et al. ”Transformers: State-of-the-Art Natural Language Processing,” EMNLP 2020 System Demonstrations.

[24] Lhoest, Q., et al. ”Datasets: A Community Library for Natural Language Processing,” EMNLP 2021 System Demonstrations.

[25] Patil, S., et al. ”Accelerate: A simple way to train and use PyTorch models with multi-GPU, TPU, and mixed precision,” GitHub Repository, Hugging Face, 2022. https://github.com/huggingface/accelerate

[26] Loshchilov, I., Hutter, F. ”Decoupled Weight Decay Regularization,” ICLR 2019. (AdamW is based on Adam, but this paper introduced the decoupled weight decay aspect often used).

[27] Zheng, L., et al. ”Judging LLMs is Hard: Introducing Chatbot Arena and LMSys-Chat-1M,” arXiv preprint arXiv:2306.05685, 2023.

[28] Elo, A. E. ”The rating of chessplayers, past and present,” Arco Pub., 1978.

[29] Shazeer, N. ”GLU Variants Improve Transformer,” arXiv preprint arXiv:2002.05202, 2020.