

LLM Causal Reasoning: Initial Report

0 Executive Summary

Objective

The Claire project explores whether small language models (LLMs) can be fine-tuned to reason causally using *supervised fine-tuning (SFT)* with synthetic data. While not tied to specific commercial use cases yet, the work aims to develop novel intellectual property (IP) that could inspire future product development, whitepapers, or VC interest.

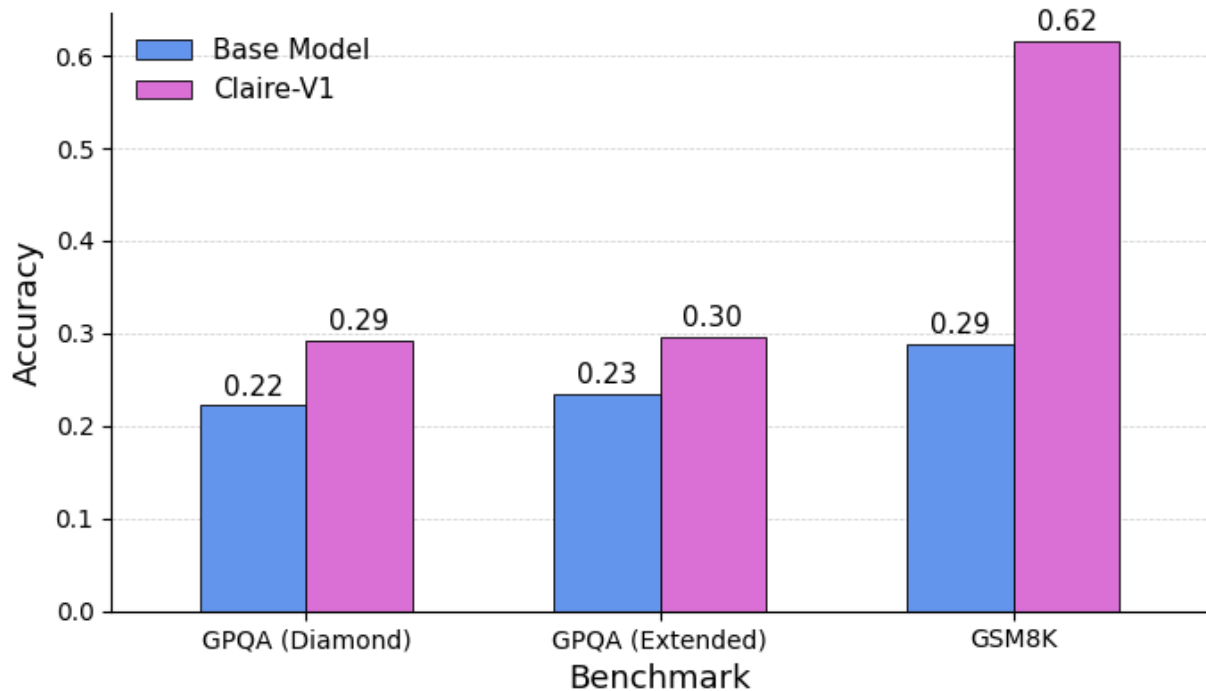
Methodology

- **Data Generation:** Synthetic training data (~4,300 training, 400 evaluation samples) was generated using GPT-4.1, guided by carefully crafted prompts embedding *thinking* and *causality* traces.
- **Models:** Three base LLMs were fine-tuned (0.5B, 1.5B, and 7B parameters), using *Low-Rank Adaptation (LoRA)* to significantly reduce training cost (~95% parameter reduction).
- **Infrastructure:** Training was done via Google Cloud (Vertex AI), HuggingFace Hub for model/data versioning, and trl, peft, wandb, and lm-eval for training and evaluation workflows.

Key Results

- **Performance:** Claire-1.5B and Claire-7B models consistently outperformed their base counterparts in causal reasoning tasks, especially on GSM8K and GPQA benchmarks.
- **Reasoning Quality:** Outputs from the fine-tuned models followed structured causal reasoning formats and demonstrated meaningful world modeling.
- **Efficiency:** Training time was modest (10–30 minutes per model) due to PEFT techniques, although model performance saturated after one epoch due to batch size limitations.

Accuracy Comparison of 1.5B Base vs SFT Model



Limitations

- **Generalization:** Performance on unseen benchmarks was limited—likely due to small model sizes and relatively narrow training data coverage.
- **Data Diversity:** The current dataset is drawn from a limited set of benchmarks and task types.

Next Steps

- **Training Enhancements:**
 - Expand the dataset to ~12,000 training and 1,200 evaluation samples.
 - Improve prompt design for richer and more diverse causality traces.
 - Experiment with reinforcement learning for causal reasoning refinement.
 - Evaluate other open-source models (e.g., LLaMA) and larger batch sizes.
- **Evaluation:**
 - Explore custom evaluation framework using an LLM-as-a-Judge
- **Deployment & Consumption:**
 - Explore inference-time scaling strategies (e.g., test-time augmentation).

- Containerize models using Docker/VLLM and expose via API for demo apps.

Strategic Relevance

Claire represents a promising step toward small, cost-efficient, causality-aware LLMs. The foundation built here may lead to compelling IP, open research artifacts, and eventually customer-facing tools that require structured reasoning—positioning us well in a competitive, rapidly evolving space.

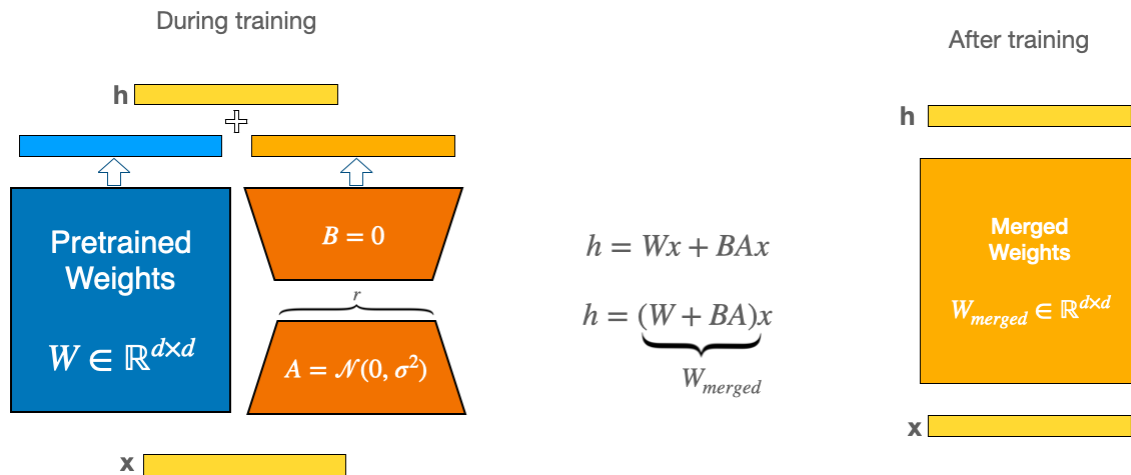
1 Introduction

- The aim of this project is to enable LLMs to reason in a causal manner by means of supervised fine-tuning (SFT)
- The desired outcome of this project is of a strategic nature and has no direct ties with any prospective customers *yet*, although I believe many prospective customers could be interested in fine-tuning a small LLM.
 - Instead, this project is mainly concerned with whether I could create some exciting IP with this that I can use to inspire VCs and prospects.
 - Beyond this, there may be interesting applications in the future, however, or whitepapers / preprints that I could publish if this seems promising
- On a high-level, the main idea is to look at how common reasoning models, such as o1, claude 3.7 sonnet or deepseek-r1, are fine-tuned using “reasoning traces” and then perform the same with “causality traces”.
 - However, I will be focussing on supervised fine-tuning for now, as opposed to reinforcement learning (RL). Most bigger reasoning models, such as o1, have been trained with a combination of both approaches.
- This project will be looking at small LLMs and assess whether we can gain performance gains when fine-tuning them on “causality traces”.
- The fine-tuned models are denoted **Claire**, which stands for **C**ausal **L**ogic in **A**I **R**easoning models.

2 Background

2.1 Low-Rank Adaptation (LoRA)

- LoRA (Low-Rank Adaptation) is a parameter-efficient fine-tuning (PEFT) technique that freezes the pre-trained model weights and injects trainable rank decomposition matrices into the model's layers.
- On a high-level, it works by adding and optimizing smaller matrices to the attention weights, typically reducing trainable parameters by about 90%-95%.
- Instead of training all model parameters during fine-tuning, LoRA decomposes the weight updates into smaller matrices through low-rank decomposition, significantly reducing the number of trainable parameters while maintaining model performance.
- During inference, these adapter weights can be merged with the base model, resulting in no additional latency overhead.



Source: [This](#) Medium article with more explanation on LoRA.

3 Methodology

3.1 Tech Stack

- Google Cloud Platform (GCP) Vertex AI + Colab Notebooks
- Initial testing was done on instances with L4 GPUs (22 GB VRAM)
- Bigger models and later training was done with A100 GPUs (40 GB VRAM)
 - More VRAM and 2-3 times faster than L4, at a slightly higher cost
- Base models were downloaded using the HuggingFace (HF) Hub
- Trained models, tokenizers, adapters, and data were version controlled using HF Hub
- Relied on the `transformers` and `trl` libraries to configure and run the training
 - These are essentially just convenience-wrappers around `torch` with a focus on LLM training and fine-tuning
- Relied on the `peft` library to accelerate training by leveraging Low-Rank Adaptation (LoRA)
 - This allowed me to reduce the number of trainable parameters by ~95% while not losing much performance, which allowed us to fine-tune at low cost and low running times.
- Relied on the Weights & Biases library `wandb` and their GUI to monitor experiments
- Relied on the `lm-eval` package for quantitative evaluation, as well as the `transformers` and Qwen model API for qualitative evaluation of responses

3.2 Generating Synthetic Data

- Utilized the `openai-agents` package and our own OpenAI end points to generate synthetic data with GPT 4.1
- Passed prompts from a specific set of common benchmark datasets to GPT 4.1 and asked it to compute responses in a very specific format that contains thinking and causality traces.
- The exact prompt that was used is as follows: `<INSERT_PROMPT_HERE>`.
- I drew examples from the following well-known benchmarks:
 - `gsm8k`
 - `gpqa` (extended)
 - `numina-math`
 - `olympic-arena`
 - `aime` (1983-2021)
- The V1 dataset ended up containing 4,323 training and 423 evaluation samples.

3.3 Data Preprocessing

- After a few experiments, I found out that the `Qwen` base models respond better to utilizing their native special tokens as opposed to our new special thinking and causality tokens.
- This meant that I had to change all occurrences of `<think>` to `<|im_start|>think`, and all occurrences of `<causality>` to `<|im_start|>causality`, while removing all occurrences of `</think>` and `</causality>`.
- I also found out that SFT improves significantly when the training loss is only computed over the completions and not the prompts / queries.
 - To do this, all I had to do was pre-process the data such that the prompts were contained in the “prompts” column and the completions, i.e. assistant responses, were contained in the “completions” column.
 - The `trl` library then took care of the rest by masking the input when computing the training loss.

4 Results

4.1 LoRA Configurations

- I tried out a lot of different configurations, but the one that ended up working best was the following:

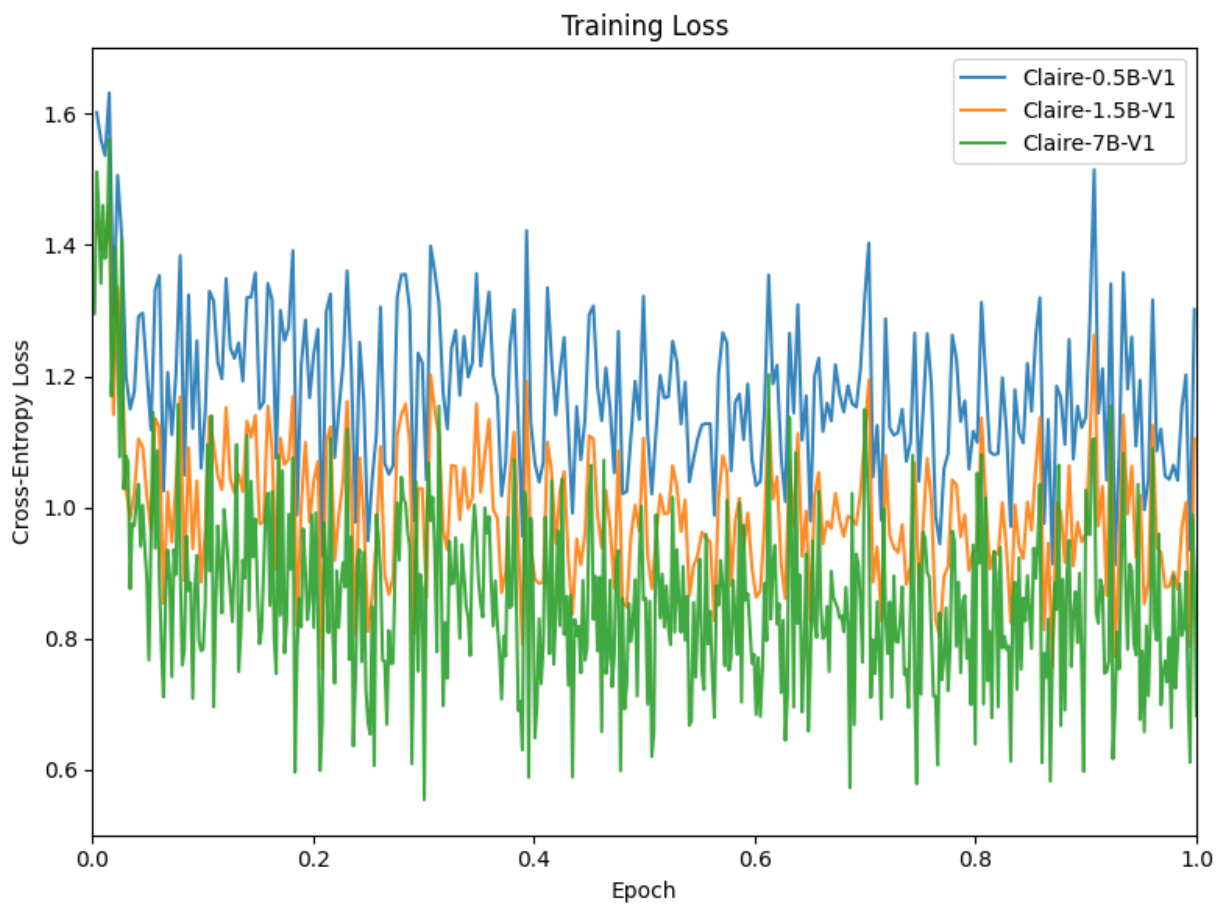
- LoRA rank = 64 (32 for the 7B model)
 - LoRA alpha = 128 (64 for the 7B model)
 - LoRA dropout = 0.05
 - LoRA applied to *all* linear modules
- This typically reduced the number of trainable parameters by ~95%.
- For instance, the reduction in trainable parameters were:
 - Claire-0.5B: From 500M to 35M parameters.
 - Claire-1.5B: From 1.5B to 74M parameters.
 - Claire-7B: From 7.6B to 80M parameters.

4.2 Training Configurations

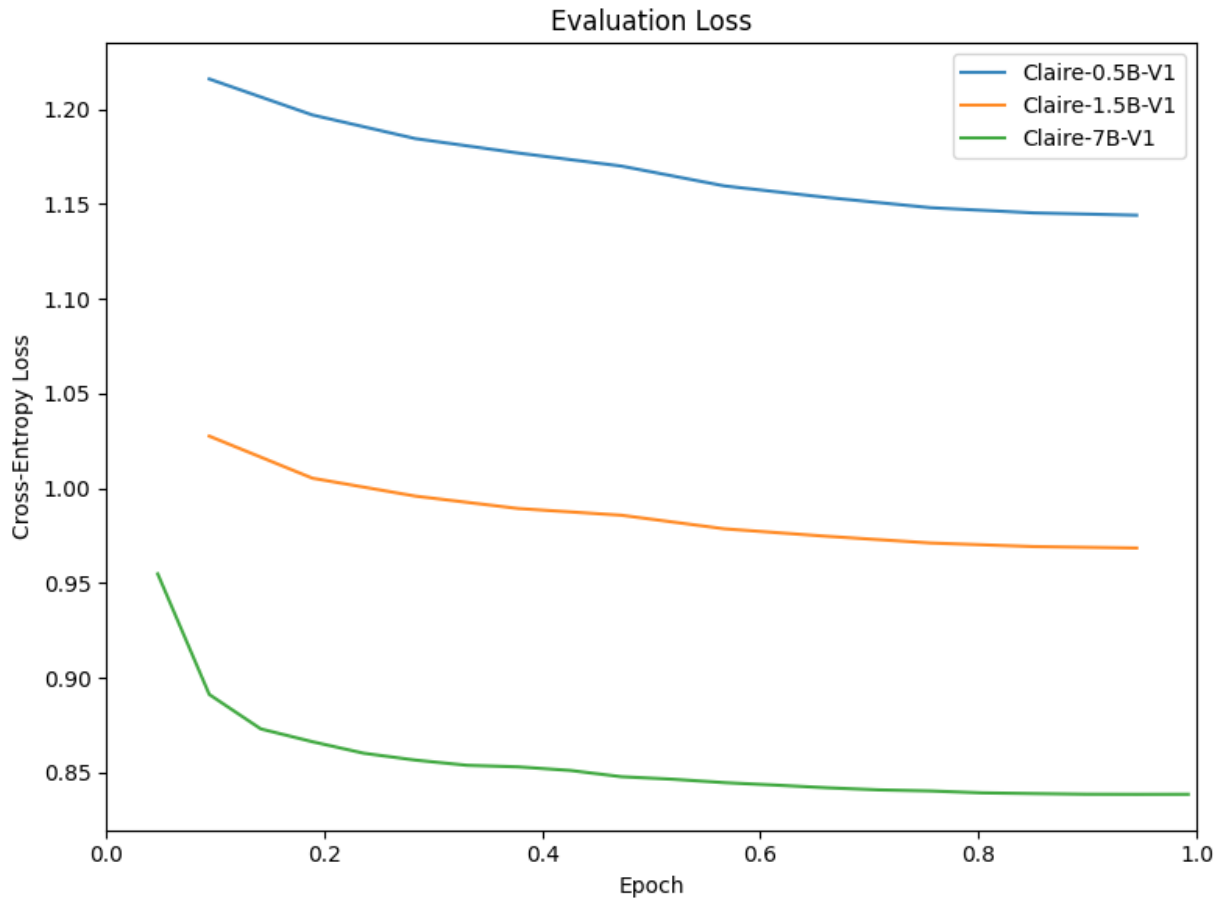
- There were a lot of different configurations that I tested, and every model and version had to be tested again for the learning rate, weight decay, learning rate scheduler, number of epochs, etc.
- However, the ones that ended up working for most models was:
 - Learning rate: 3×10^{-4}
 - Learning rate scheduler: cosine
 - Warm-up ratio: 0.05
 - Weight decay: 1×10^{-4}
 - Batch-size: 4
 - Gradient accumulation: 4 (meaning the effective batch size was 16)
 - Number of epochs: 1
 - Training for more than 1 epochs with very small batch sizes resulted in overfitting. If you want to train for more than 1 epoch, we would have to increase the batch size to something bigger than 128, which was computationally infeasible for now.
- This resulted in training times between 10-30 minutes depending on the base model.

4.3 Training Results

- The loss function is cross-entropy of the generated assistant response, which is also a function of the dictionary size.
 - Generally, however, lower is better, and a value below 1 is acceptable during SFT
- See below for screenshots of the training cross-entropy loss as a function of steps.
- The screenshots showcase the best SFT models from each category.



- See below for screenshots of the evaluation cross-entropy loss as a function of steps



- It's clear that the 0.5B parameter model does not yield great cross-entropy on the evaluation dataset, but the 1.5B and the 7B parameter models both yield good completion.

4.5 Evaluation Results (V1)

- I ran some quantitative evaluation on the [gsm8k](#) and [gpqa](#) datasets using the [lm-eval](#) package with the results shown in the screenshots below:
- Claire-0.5B evaluation results:

LLM Eval Results: Comparison			
Benchmark	Metric Name	Qwen2.5-0.5B	Claire-0.5B
gpqa_diamond	acc	0.23	0.27
gpqa_extended	acc	0.26	0.27
gsm8k	exact_match,strict-match	0.22	0.31

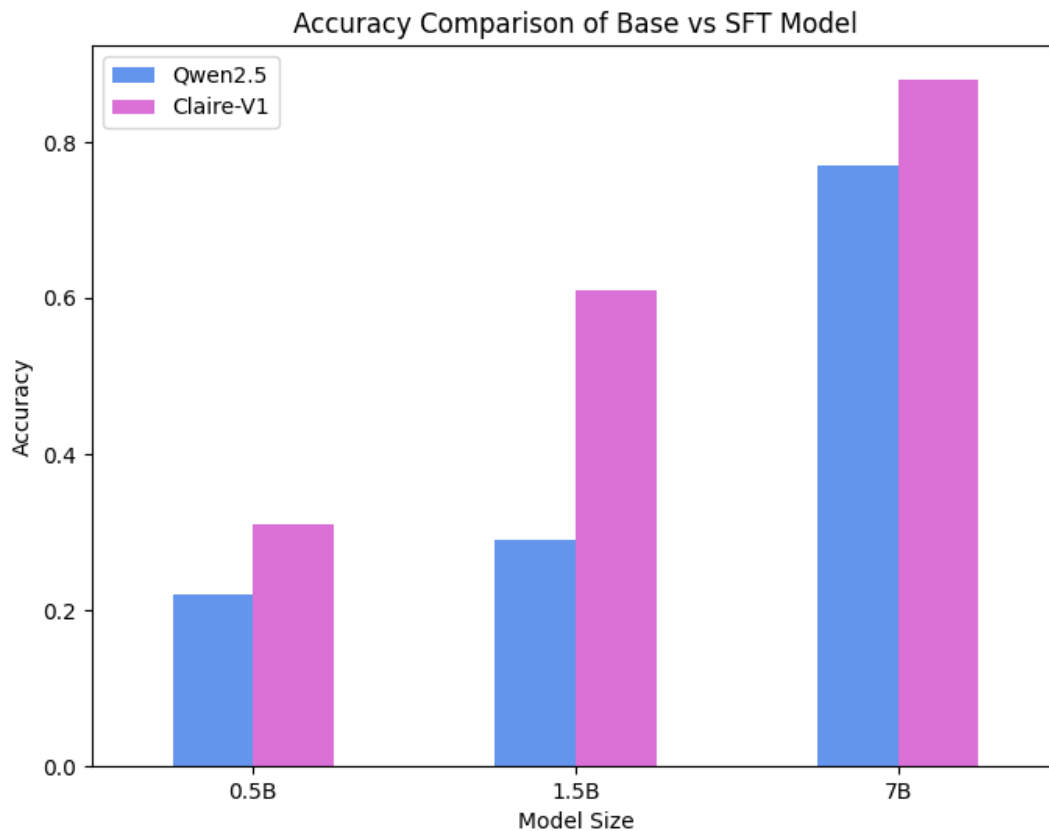
- Claire-1.5B evaluation results:

LLM Eval Results: Comparison			
Benchmark	Metric Name	Qwen2.5-1.5B	Claire-1.5B
gpqa_diamond	acc	0.29	0.29
gpqa_extended	acc	0.31	0.29
gsm8k	exact_match,strict-match	0.29	0.61

- Claire-7B evaluation results:

LLM Eval Results: Comparison			
Benchmark	Metric Name	Qwen2.5-7B	Claire-7B
gpqa_diamond	acc	0.32	0.38
gpqa_extended	acc	0.33	0.35
gsm8k	exact_match,strict-match	0.77	0.88

- The above tables show that our Claire models almost always outperform the base model on all benchmarks, in particular the gsm8k benchmark.
- See below for a plot showcasing these numbers in more detail:



4.6 Qualitative Analysis

Here is an excerpt of a response from the **Claire-1.5B-V1** model, which follows the required format extremely well and produces a reasonable answer. Note that this question is not at all in the training data.

<|im_start|>system

You are Qwen, created by Alibaba Cloud. You are a helpful assistant.<|im_end|>

<|im_start|>user

Let A cause C, B cause C, and W cause both A and B. What is the root cause of C?

<|im_end|>

<|im_start|>assistant

<im_start>think

Let's break down the causal relationships:

- A causes C ($A \rightarrow C$)

- B causes C ($B \rightarrow C$)

```
# - W causes both A and B ( $W \rightarrow A$ ) and ( $W \rightarrow B$ )

# The question asks: "What is the root cause of C?"
<im_start>causality
# World model:
# There are three variables: A, B, and C.
# Each variable has an effect on C through two different paths:
# 1. Path from A to C:  $A \rightarrow C$ 
# 2. Path from B to C:  $B \rightarrow C$ 
# 3. Path from W to both A and B:  $W \rightarrow A$  AND  $W \rightarrow B$ 

# To find the root cause(s), we need to consider which variable(s)
directly cause(s) C without any intervening steps.

# Step-by-step reasoning:
# If there were no other factors involved, then the direct cause
would be whichever variable appears first in the chain leading to C.
# In this case, if A or B caused C, that would be the root cause; but
since W also affects both A and B, it could potentially act as a
common cause for both A and B, thereby indirectly affecting C via
their effects on A and B.

# However, the problem specifies that W causes both A and B, so its
presence means that A and B cannot exist independently of each other
due to W's influence. Therefore, the only way for C to depend on A or
B is if they are causally linked to C directly, not because of W.

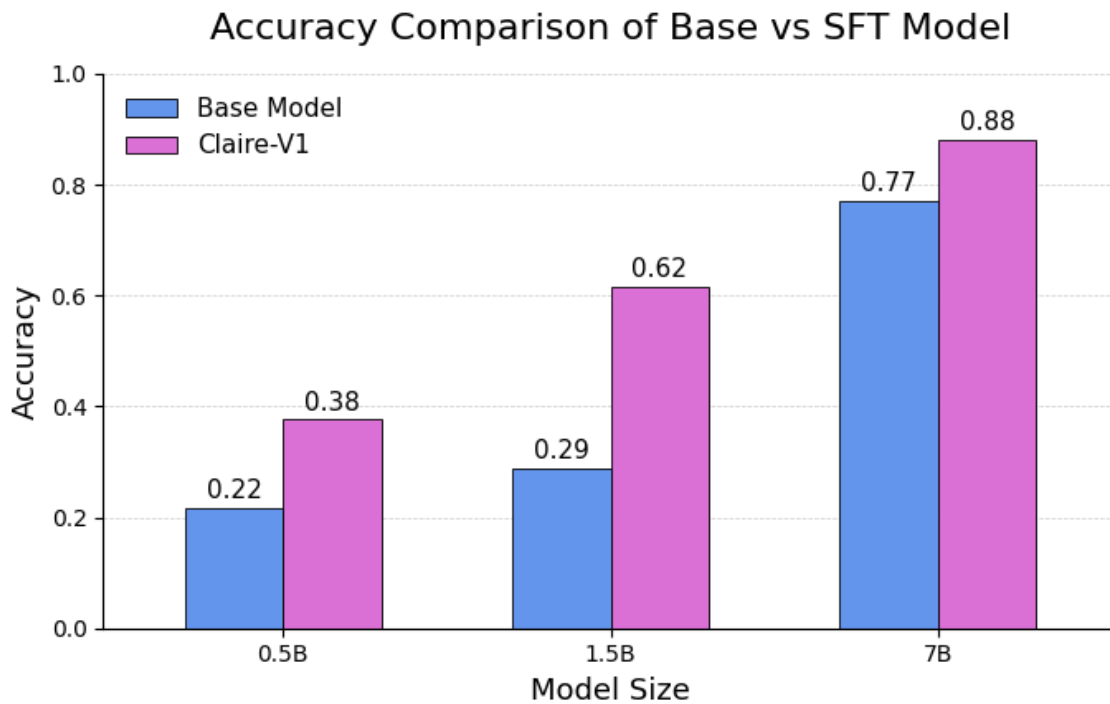
# Thus, the root cause of C must be either A or B, depending on
whether one of them directly causes C, or if both do, then neither
can be considered the sole root cause unless specified otherwise.

# But more specifically, the world model shows that W is necessary
for both A and B, so if A or B exists at all, it must have been
brought about by W. So, the root cause of C is W, because it is the
only factor present in the causal chain that ensures both A and B
exist, thus necessitating their existence for C to occur.
<im_start>think
# Therefore, the answer is: W is the root cause of C.
<im_start>answer
W is the root cause of C.
```

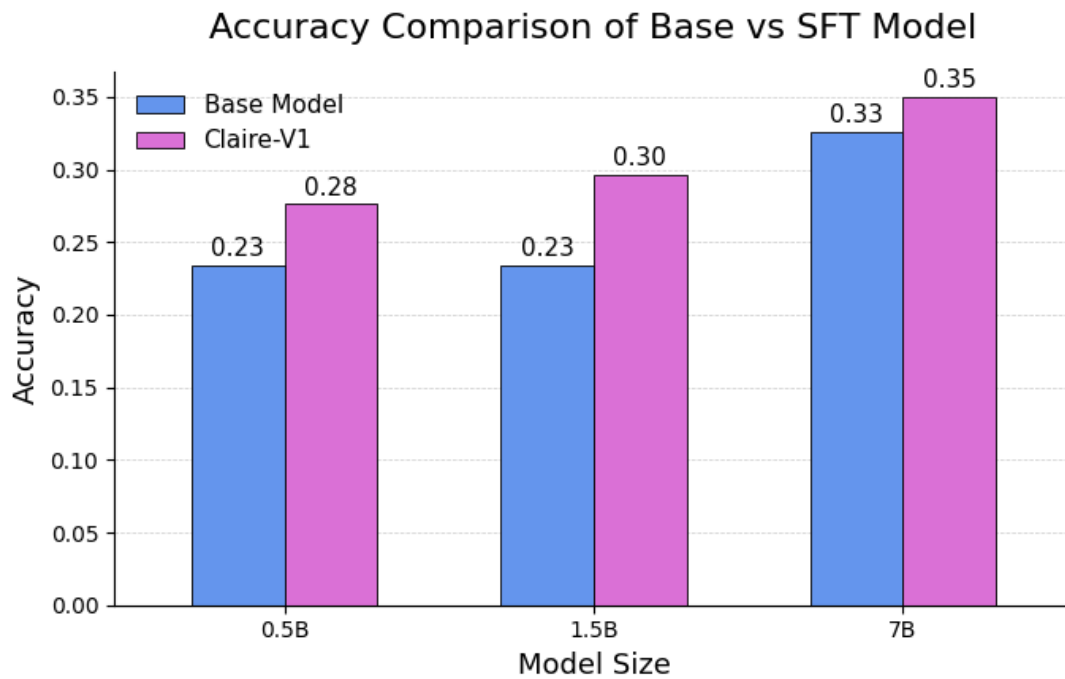
4.7 Evaluation Results (V2)

- Ran training and evaluation again using V2 version of synthetic data with ~12k rows of training data.
- Results of all benchmarks for different model sizes are shown below, followed by results for different benchmarks at the same model size.

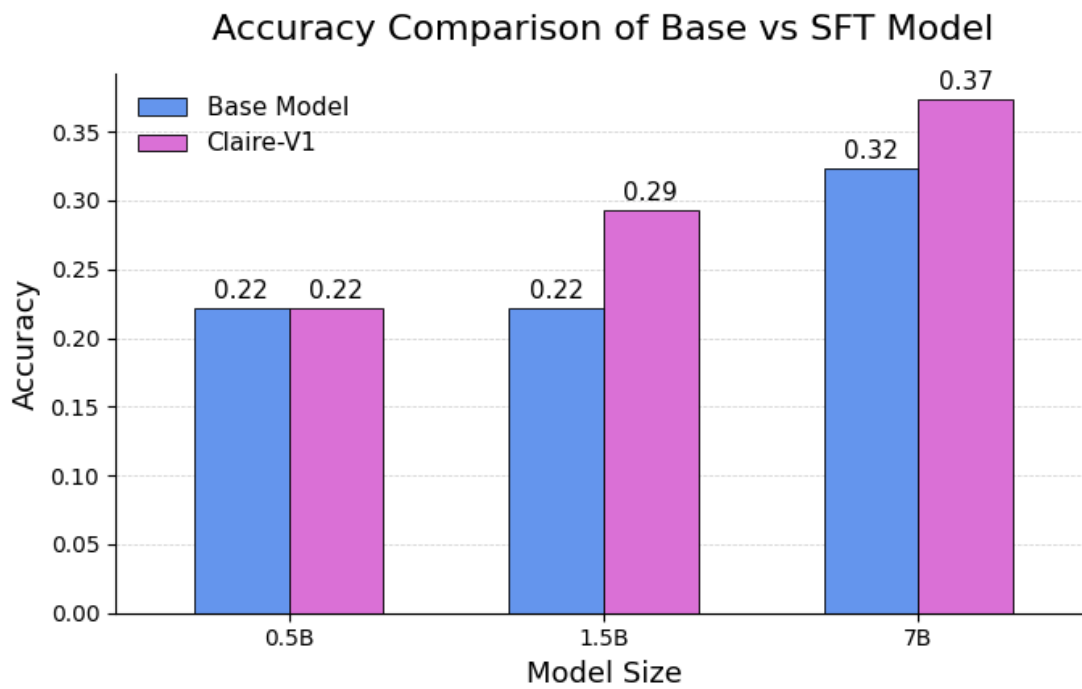
4.7.1 GSM8K



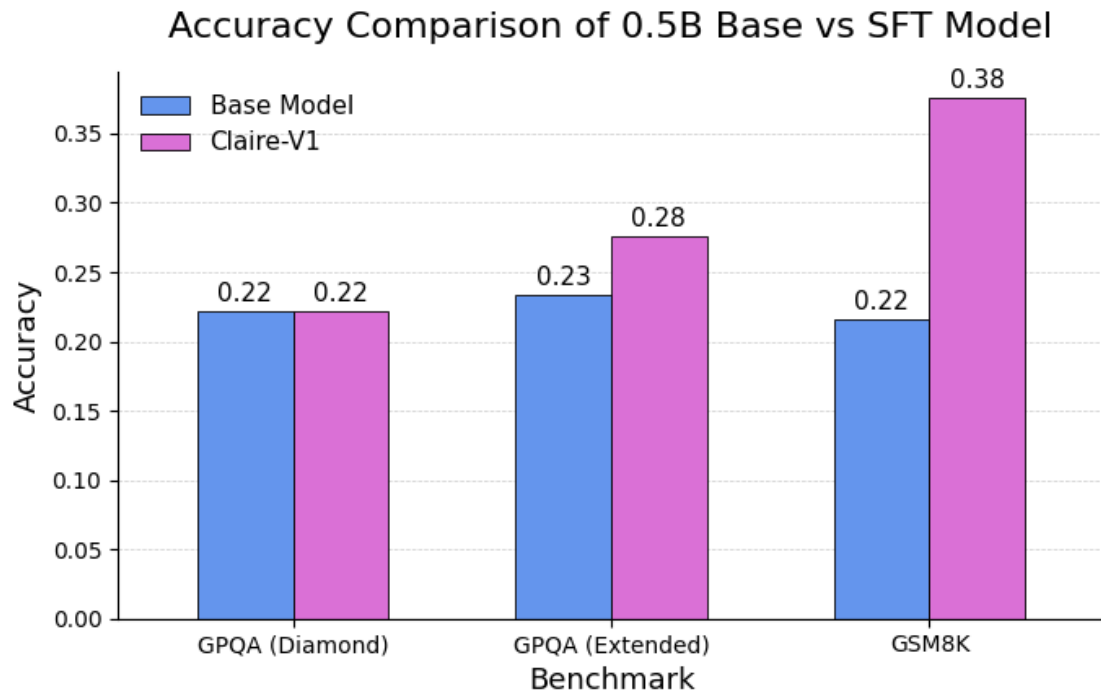
4.7.2 GPQA Extended



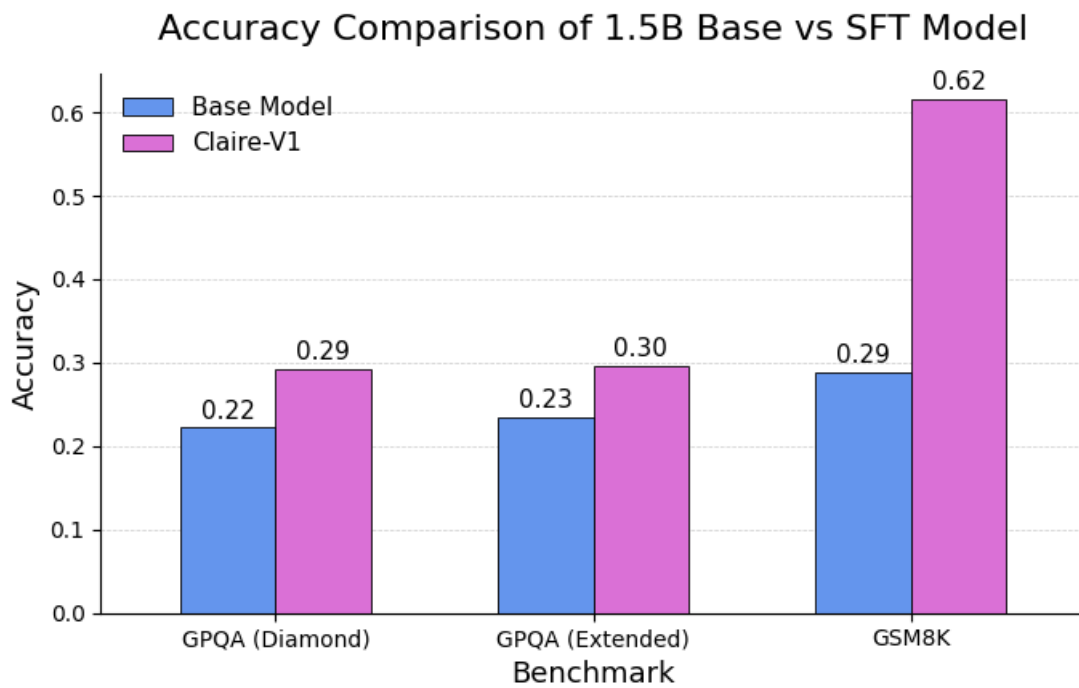
4.7.3 GPQA Diamond



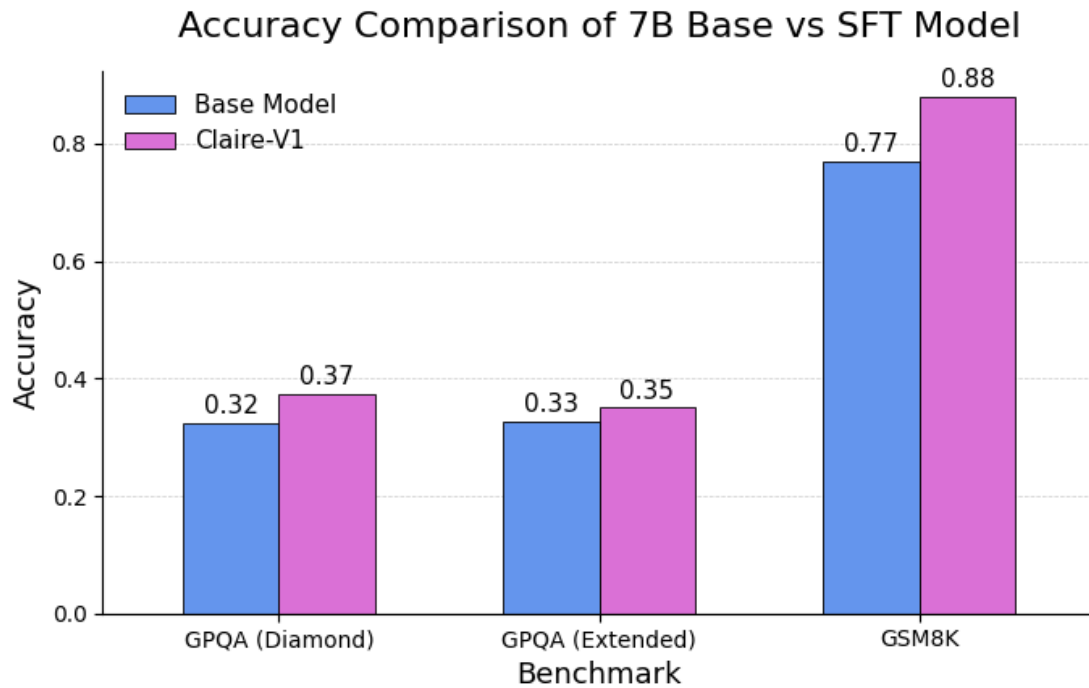
4.7.4 0.5B Comparison



4.7.5 1.5B Comparison



4.7.6 7B Comparison



5 Conclusion

5.1 Summary

- In this project I successfully applied supervised fine-tuning (SFT) to several small LLMs in order to obtain specialized causal reasoning models, utilizing training data that contained thinking and causality traces.
- I call the fine-tuned models **Claire**, which stands for **Causal Logic in AI REasoning** models.
- The fine-tuned models showed significant improvements in performance over the base model on selected, specialized tasks.
- The responses of the fine-tuned models contain both thinking and causality traces, showcasing qualitative improvements and reasoning capabilities.

5.2 Limitations

- We unfortunately do not observe amazing generalization performance on benchmark datasets that were not included in the training set.

- In my opinion, that is to be expected, since the models that we used were very small and we had very little training data.
- I would expect the generalization performance to improve if we:
 - Increased the variety of sources in the training data, and
 - Increased the sample size of the training data.
- The open-source evaluation framework that I used was very lackluster in terms of capabilities.
 - It would often mark responses as wrong when they are correct.

5.3 Future Directions

5.3.1 Training

- Apply to a wider variety of open-source base models such as Llama
- Generating better synthetic data from a wider variety of tasks, which may include
 - Generating more samples,
 - Utilizing more reasoning datasets,
 - Generating longer reasoning traces,
 - Improving the prompt to the synthetic data generator,
 - Utilizing a reasoning model such as DeepSeek-R1 instead of GPT-4.1.
- Try out reinforcement learning fine-tuning as opposed to supervised fine-tuning

I have already started to generate more data using this updated prompt for GPT 4.1:

You are a helpful assistant whose task it is to generate synthetic data for the supervised fine-tuning of a smaller LLM.

You will be given a question and your task is to generate an answer and **causal** reasoning traces.

Your reasoning trace should involve building a causal world model based on your knowledge, the query, and any other user context. You should try very hard to reason causally to the best of your ability. You should think step-by-step and reassess your causal reasoning if necessary.

When outputting your causal reasoning, you should use `<think>...</think>` XML tags to denote the reasoning trace, and `<causality>...</causality>` XML tags to denote the use of causality-aware reasoning. Note that the causality tokens should be used within the thinking tokens.

You MUST return the ENTIRE reasoning trace, i.e. everything in the `<think>` and `<causality>` XML tags as well.

You should follow this format:

<think>

Do some thinking, if necessary (you may exclude this part if not needed)

<causality>

Build a world model of relevant concepts, from the query and your knowledge

Reason about the world model and query in a causal manner

</causality>

Do some more thinking, if necessary (you may exclude this part if not needed)

</think>

Finally, answer the query based on your thinking above.

Instead of generating 4323 training and 423 evaluation samples, I am planning on generating 12105 training and 1214 evaluation samples using the above prompt, drawing from more examples from the discussed benchmarks (i.e. I'm not including more benchmarks at this point in time; I should be able to add them in later if I want to generalize better).

5.3.2 Evaluation

- Utilize an LLM-as-a-Judge framework using, e.g., the [openai-agents](#) package
 - Since the [lm-eval](#) package is severely lackluster and often marks correct answers as incorrect, biasing the evaluation

5.3.1 Consumption

- Test-time scaling during inference, similar to the s1 paper
- Deploying the fine-tuned LLM via docker or vllm and serving it in an app