Major Project-I Report on

Optimizing LLMs: Merging Strategies in Building LLMs for Reasoning

Submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY
in
ARTIFICIAL INTELLIGENCE
by

M Hemanth Kumar (211AI025)
Sidhaarth Murali S (211AI035)
Aaryan Nijhawan (211AI002)

under the guidance of

Dr. Anand Kumar M

DEPARTMENT OF INFORMATION TECHNOLOGY
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL, MANGALORE - 575025

November, 2024

DECLARATION

I/We hereby *declare* that the Major Project-I Work Report entitled ***Optimizing LLMs: Merging Strategies in Building LLMs for Reasoning*** , which is being submitted to the **National Institute of Technology Karnataka, Surathkal**, for the award of the Degree of Bachelor of Technology in Artificial Intelligence, is a *bonafide report of the work carried out by us.* The material contained in this Major Project-I Report has not been submitted to any University or Institution for the award of any degree.

Name of the Student (Registration Number) with Signature

(1) M Hemanth Kumar (211AI025)

(2) Sidhaarth Murali S (211AI035)

(3) Aaryan Nijhawan (211AI002)

Department of Information Technology
Place : NITK, Surathkal
Date : 04/11/2024

CERTIFICATE

This is to *certify* that the Major Project Work Report entitled ***Optimizing LLMs: Merging Strategies in Building LLMs for Reasoning*** submitted by

Name of the Student (Registration Number)

(1) Hemanth Kumar M (211AI025)

(2) Sidhaarth Murali S (211AI035)

(3) Aaryan Nijhawan (211AI002)

as the record of the work carried out by him/her/them, is *accepted as the B.Tech. Major Project-I work report submission* in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Artificial Intelligence in the Department of Information Technology, NITK Surathkal.

Signature of Major Project Guide with date

Dr. Anand Kumar M

Associate Professor

Department of Information Technology

NITK Surathkal-575025

ABSTRACT

Model merging has emerged as a promising approach to democratize and enhance the model-building process by integrating multiple pre-trained models into a unified framework. This strategy addresses limitations of traditional fine-tuning methods, such as the risk of catastrophic forgetting, by combining the strengths of domain-specific pretraining and task-specific fine-tuning. Despite its potential, model merging often relies heavily on intuition and domain expertise, which can be restrictive given the growing diversity of open models and tasks. In this work, we propose a systematic approach to develop a high-performing language model for reasoning tasks across three key domains: mathematical reasoning, code reasoning, and commonsense reasoning. Our methodology is divided into three primary components: (i) Pretrained Large Language Models (LLMs), (ii) Finetuning, and (iii) LLM Merging.

Our results show that the Mistral model fine-tuned with Direct Preference Optimization (DPO) achieves the highest accuracy on the GSM8K dataset (0.73), outperforming LLAMA. Supervised Fine-Tuning (SFT) yields lower performance than DPO, especially for mathematical tasks. The SLERP merging technique proves most effective, achieving 0.54 in Math and 0.71 in Commonsense accuracy with Mistral, while TIES performs best in Math for LLAMA. LoRA-SFT offers moderate accuracy but does not surpass SLERP.

Keywords— Model Merging, Parameter Space Techniques, Data Flow Space, Task Arithmetic, TIES Merging, Franken Merging

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Overview

The rapid evolution of large language models (LLMs) has opened up new possibilities for improving model capabilities, including reasoning tasks that require sophisticated handling of language. This work, "Optimizing LLMs: Merging Strategies in Building LLMs for Reasoning Tasks," focuses on utilizing cutting-edge techniques such as model merging, fine-tuning, and quantization to create more efficient and powerful models.

Given the exponential growth in open-source models and the availability of over 500,000 models on platforms like Hugging Face [1], there is an urgent need to explore methods for enhancing their performance while minimizing resource consumption. Model merging, which has gained traction in computer vision, is emerging as a promising technique in LLMs, allowing the integration of multiple models' knowledge into one unified model. This not only leads to enhanced reasoning capabilities but also serves as a cost-effective solution compared to training a new model from scratch.

In this work, we propose a systematic approach to develop a high-performing language model for reasoning tasks across three key domains: mathematical reasoning, code reasoning, and commonsense reasoning. Our methodology is divided into three primary components: (i) Pretrained Large Language Models (LLMs), (ii) Finetuning, and (iii) LLM Merging.

1.2 Motivation

It is well acknowledged that language models and other NLP models struggle with reasoning, especially multi-step reasoning [2, 3, 4]. At a certain size, such as models with over 100 billion parameters, recent research has revealed that reasoning capacity may arise in language models [5, 6]. In this work , we follow [5] in considering reasoning as an ability that is rarely present in small-scale models like GPT-2 [7] and BERT [8], and therefore focus on techniques applicable to improving or eliciting

"reasoning" in LLMs such as GPT-3 [9] and PaLM [10].

1.2.1 Fully Supervised Fine-tuning

It is important to note that there is ongoing research on improving reasoning in small language models through fully supervised fine-tuning on targeted datasets. Models trained with explanations perform better on commonsense question-answering tasks [11], as shown by the work of Rajani et al. [12], who fine-tuned a pretrained GPT model [13] to provide rationales that explain model predictions using the constructed CoS-E dataset.

Fully supervised fine-tuning suffers from two serious flaws. First, it requires a dataset with explicit reasoning, which can be challenging and time-consuming to generate. Furthermore, the model is restricted to a single dataset for training, limiting its use to a single domain and increasing the likelihood that it would depend on artifacts in the training data rather than true reasoning to generate predictions.

1.2.2 Prompting & In-Context Learning

Through in-context learning, large language models like GPT-3 [9] have shown remarkable few-shot performance on a wide range of tasks. A query and some "input, output" examples are all that's needed to get these models "reasoning" about how to approach an issue and find a solution, either implicitly or explicitly. While these models have improved, they still struggle with problems that call for multiple steps of reasoning to resolve [2, 3, 4]. Recent research has revealed that this might be because the full potential of these models has not been explored.

1.2.3 Chain of Thought

By instructing LLMs to engage in "reasoning" explicitly, we can increase the likelihood that they will reason rather than merely provide answers. Wei et al. [14] suggest using chain-of-thought prompting as a means to this end. The "chain of thought" (CoT) examples provided in this method represent intermediary steps in the process of thinking using natural language.

Specifically, in CoT prompting, ⟨input, output⟩ demonstrations are replaced with ⟨input, chain of thought, output⟩ triples.

Example

Input: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

Chain of Thought: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$.

Output: The answer is 11.

Training large-scale language models from scratch is an extremely compute-intensive process, making pretraining impractical without substantial resources. On the other hand, finetuning a pre-trained model often leads to catastrophic forgetting, where the model loses previously learned knowledge in favor of newly fine-tuned tasks. This is especially problematic when the goal is to build a general model that excels in reasoning across diverse domains, as catastrophic forgetting hinders the model's ability to generalize [15].

Other advanced methods, such as Mixture of Experts (MoE), hold promise for scaling models efficiently by only activating parts of the model for specific tasks. However, MoE still demands considerable compute for routing between experts, making it less viable in our case [16]. Similarly, LoRA adapters offer more efficient finetuning by reducing the number of parameters to update, but applying them across multiple domains like math, coding, and commonsense reasoning introduces complexity and overhead [17]. Additionally, model routing, where specialized smaller models are trained and routed for different tasks, still incurs high compute costs as each task-specific model needs to be trained [18].

1.2.4 Current Best Models for Reasoning Tasks

Math Reasoning

- **MetaMath Mistral (Meta, 2024)**: MetaMath Mistral 7B is a cutting-edge model optimized for mathematical reasoning tasks. It performs well in solving advanced mathematical problems, building on improvements in meta-reasoning techniques [19].

Code Reasoning

- **CodeLlama (Meta, 2024)**: CodeLlama is a large language model fine-tuned specifically for code reasoning and generation. It supports various programming languages and handles complex reasoning tasks in software development [20].

Common Sense Reasoning

- **Phi-3 3.8B (Microsoft, 2024)**: Phi-3 is an advanced language model that excels in commonsense reasoning tasks. It has been fine-tuned on instruction-based datasets and shows strong performance in commonsense question answering [21].

Multimodality in Math, Code, and Commonsense Reasoning

- **LLaVA Model**: LLaVA is a multimodal model that performs well across different reasoning tasks, including math, code, and commonsense reasoning. By integrating visual and textual data, it enhances its capability in multimodal reasoning scenarios, excelling in tasks that require understanding across different modalities [22].

Given these challenges, model merging emerges as a compelling alternative. Instead of re-training or fine-tuning models, merging pre-trained models from various domains, such as math, coding, and commonsense reasoning, allows us to build a general model capable of reasoning across multiple domains without requiring significant compute. This approach avoids the pitfalls of catastrophic forgetting, leverages existing model expertise, and minimizes compute, making it a practical path toward developing a large language model that excels at reasoning. Our goal is to create a merged model that not only performs on par with the best models for individual tasks, such as math or coding, but also excels in handling multiple reasoning tasks at once.

CHAPTER 2

LITERATURE REVIEW

2.1 Background

Recent advancements in language model research have introduced techniques for model merging, such as task vectors, TIES-Merging, and DARE. These methods have been integrated into open-source software for large language models (LLMs), like MergeML [23]. This section reviews common algorithms for merging LLMs and discusses how model merging has transformed the alignment process of LLMs.

Practitioners typically start with a pretrained model (e.g., LLaMA or Mistral) and adapt it to specific tasks by either prompt engineering or finetuning on additional data. Finetuning can be costly and requires task-specific data. *Task arithmetic* [24] offers a simpler alternative by utilizing *task vectors*, which are the differences between the weights of a finetuned model and the pretrained model. These vectors encapsulate the knowledge acquired during finetuning for a particular task. By adding or subtracting task vectors from a model's parameters, one can modify the model's behavior without additional training or data.

Building upon this concept, *TIES-Merging* [25] addresses challenges in merging multiple models, such as interference from redundant parameters and conflicting parameter signs. TIES-Merging mitigates these issues by retaining only the most influential parameters, resolving sign conflicts through a majority vote, and merging the remaining parameters. This method enhances the performance of merged models across different modalities and scales well with the number of models merged.

DARE (Drop And REscale) [26] further refines the merging process by sparsifying the delta parameters—the differences between pretrained and finetuned models. DARE randomly drops a portion of these parameters and rescales the remaining ones, effectively reducing redundancy. This technique is particularly effective for models finetuned via supervised learning and improves the efficiency of merging when combined with methods like TIES-Merging.

Model merging techniques have also been applied to the alignment of LLMs. The standard alignment pipeline involves pretraining, supervised finetuning (SFT), and re-

inforcement learning from human feedback (RLHF). During RLHF, a balance must be struck between maximizing reward and maintaining similarity to the original model, often regulated by a Kullback-Leibler (KL) divergence term. *WARP* (Weight Averaged Rewarded Policies) [27] leverages model merging by:

1. Using an exponential moving average of model weights as an anchor during RLHF.

2. Merging independently finetuned models using task vectors and spherical linear interpolation.

3. Interpolating towards the SFT initialization.

This multi-stage merging strategy improves the trade-off between reward optimization and divergence, resulting in better-aligned models.

These advancements in model merging techniques have significantly impacted the development and alignment of LLMs. They provide efficient alternatives to traditional finetuning methods, enhance model performance across various tasks, and contribute to more scalable and effective alignment processes.

2.2 Related Works

The advancement of large language models (LLMs) has led to various methods for combining multiple models to enhance performance, efficiency, or specialization. This literature review explores key approaches, including parameter averaging, mixture-of-experts (MoE) merging, model stacking, model zipping, model routing, and neural network pruning. Each method offers unique advantages in merging or integrating LLMs for diverse applications.

Mixture-of-Experts (MoE) models divide the computational workload among specialized expert models. **Merging of Experts** can speed up inference by combining these experts into a single model [23]. This consolidation reduces redundancy and improves efficiency. Another approach involves merging models into a new MoE model with token-based gating [28]. This method allows different tokens in the input sequence to be processed by the most suitable expert, enhancing specialization

and performance. **Post-Hoc Routing** among specialized Parameter-Efficient Fine-Tuning (PEFT) modules enables dynamic selection of experts after training [29]. This flexibility allows the model to route inputs to the most appropriate expert modules without retraining the entire system.

Model stacking involves applying LLMs sequentially or combining their outputs to improve overall performance. **Self-Correctors** [27] and **Aligners** [30] pass the generations of one LLM as input to another, which then produces the final output. This iterative process refines responses and can correct errors or improve coherence. **Proxy-Tuning** [29] combines next-token predictions from a larger base model and a smaller fine-tuned LLM. This method leverages the strengths of both models—the generality of the base model and the specialization of the fine-tuned model. **Controlled Decoding** [31] adjusts next-token probabilities to align them with a reward function via a prefix scoring LLM. This approach guides the generation process to produce outputs that satisfy specific criteria or constraints.

Model zipping combines layers of different LLMs to create a more efficient or capable model. **CALM** [32] integrates models via cross-attention mechanisms, allowing layers from different models to interact. This method enhances the model's ability to capture complex dependencies in the data. **FrankenLM** [33], also known as FrankenBERT, extends the hidden dimensions by stitching together layers from different models. This composite model can benefit from the strengths of its constituent parts, potentially improving performance on various tasks.

Model routing aims to select the best LLM for each input using a smaller routing model, keeping the candidate LLMs intact. **LLM Blender** [34] trains a small model to rank candidate generations from multiple LLMs. The routing model evaluates the outputs and selects the one that best fits the desired criteria. LLM Blender operates by first generating candidate responses from several LLMs. A smaller, efficient model then assesses these responses based on predefined metrics such as relevance, coherence, or factual accuracy. By ranking the candidates, the system selects the optimal response without retraining the large models. This approach allows for dynamic utilization of multiple models' strengths while maintaining efficiency. **Federation of Experts (FoE)** [35] uses token embeddings from candidate expert LLMs to select the best one for processing each input. This token-level routing ensures that each

part of the input is handled by the most suitable expert. Other methods [36, 28] involve training meta-models to predict whether an LLM will produce a good generation for a given input [37]. These meta-models evaluate the input and determine the likelihood of successful output from each candidate LLM, facilitating informed routing decisions.

Neural network pruning is a technique that starts with a large neural network and aims to derive a smaller network with comparable performance. The typical multi-step process includes: (1) **Training the model to convergence.** (2) **Pruning the model's weights** (e.g., removing the lowest-magnitude weights in each layer). (3) **(Optionally) Rewinding the model's remaining weights** to their original values. (4) **Training the subnetwork to convergence.** This process can be iteratively applied to derive higher-performing subnetworks by gradually removing weights, rather than pruning many weights simultaneously. Pruning became popular in the late 2010s and remains an active research area due to its ability to produce smaller models without significant performance loss.

Key pruning techniques and findings include **Learning Structured Sparsity in Deep Neural Networks** [38], which explores pruning via the L1-norm heuristic, removing low-magnitude weights—a commonly used strategy. **Pruning Filters for Efficient ConvNets** [39] demonstrates significant reductions in computational cost by removing filters with low L1-norm without sacrificing performance. **Rethinking the Value of Network Pruning** [40] provides an extensive empirical analysis to determine best practices for high-performing subnetworks. **The Lottery Ticket Hypothesis** [41] discovers that high-performing subnetworks exist within randomly-initialized weights, capable of training in isolation to match original network performance.

With the rise of LLMs, pruning research has adapted to address new challenges. **SparseGPT** [42] proposes a pruning algorithm for GPT-style models that achieves over 50% sparsity in one shot, without retraining. It reduces pruning to sparse regression problems that can be efficiently approximated. Despite eliminating retraining, SparseGPT remains computationally intensive, taking 4–5 hours to prune a $\sim$100B parameter model on a single GPU. **Wanda (Pruning by Weights and Activations)** [43] introduces a simple yet effective pruning approach for LLMs. Wanda mul-

tiplies each weight by its corresponding input activation per output basis, determining which weights to prune without retraining. Understanding these sparse activations allows for more effective pruning strategies that exploit the inherent properties of LLMs.

Pruning and model merging are interconnected through their reliance on sparsity. Techniques developed in pruning research can inform model merging strategies, especially when aiming to reduce model size without compromising performance. As LLMs grow in size, efficient methods like pruning become crucial for deployment and integration.

Authors	**Methodology**	**Merits**	**Limitations**
Wortsman et al. (2022)	Model Soup: This method involves the simple weighted average of model parameters. It merges fine-tuned models using linear interpolation to generate a combined model.	Leads to a flatter local minimum, which improves generalization and performance on out-of-distribution data.	Not suitable for merging complex or vastly different models due to parameter conflicts, leading to degraded performance.
Shoemake (1985)	SLERP (Spherical Linear Interpolation): The model parameters are interpolated on a hypersphere, treating them as points in high-dimensional space. The shortest path between the parameters is calculated for interpolation.	Preserves the distinct features of each model, making it effective for merging models with different characteristics.	Only merges two models at a time, requiring repeated application for merging more models. Not task-specific.

Continued on next page

Continued from previous page

Authors	Methodology	Merits	Limitations
Ilharco et al. (2022)	Task Arithmetic: Creates task vectors by subtracting the weights of a base model from a fine-tuned model, enabling manipulation with arithmetic operations like addition and negation for task-specific behavior adjustments.	Enables task-specific improvements by adjusting the merged model's behavior.	May not generalize well to tasks unseen during training, potentially leading to performance degradation on unrelated tasks.
Ainsworth et al. (2022)	TIES-Merging: A three-step process involving TRIM (reset parameters with minimal changes), ELECT SIGN (resolve conflicting parameter signs), and merging only aligned parameters.	Effective for merging more than two models and reduces parameter interference.	Complex to implement; tuning TRIM and sign election strategies is critical for optimal performance.
Hu et al. (2022)	DARE (Drop and Rescale): Drops insignificant delta parameters between fine-tuned and base models and rescales to maintain output consistency. It strategically promotes sparsity in merged models.	Reduces interference, promotes sparsity, and improves efficiency in model merging.	Might discard useful parameters if not carefully pruned. Best results when combined with methods like TIES.

Continued on next page

Continued from previous page

Authors	Methodology	Merits	Limitations
Dettmers et al. (2022)	Frankenmerge: Combines specific layers from different models to form a hybrid model by concatenating layers. This approach merges models layer-wise to build a new architecture.	Allows combining strengths of models across different layers, leading to a more versatile architecture.	Performance is unpredictable. Finding the optimal combination of layers often requires significant trial and error.

Table 2.2.1: Literature Survey

2.3 Outcome of Literature Review

However, developing new Frankenmerging techniques is still a challenge for the community and requires extensive experimentation to discover effective recipes.

The popularity of merging image generation models surged in the community once these methods were integrated into open-source toolkits [23]. A similar trend occurred with language model merging following the release of mergekit [23]. This toolkit offers a comprehensive suite of recipes for merging language models, including basic linear and spherical interpolation, as well as more advanced techniques like Task Arithmetic, TIES-Merging, and DARE. Users can experiment with these methods to combine fine-tuned versions of popular base models, such as Mistral [28].

As a result, the community has developed a significant number of effective merged models, and many of the top models on the Open LLM Leaderboard [1] are now dominated by these community-created merged models.

Currently, most people rely on a similar Frankenmerging recipe, and there has been minimal experimentation to improve it. This area remains largely unexplored, and there is potential for evolution to play a significant role in advancing these techniques.

2.4 Problem Statement

Optimize Large Language Models (LLMs) for diverse reasoning tasks by improving LLM merging techniques, starting with math reasoning and extending to coding, commonsense, and multimodal applications.

2.5 Objectives of the Project

1. Analyze existing LLM merging techniques to understand their successes and failures across reasoning tasks.

2. Develop effective methods to merge LLMs, enhancing performance in mathematical reasoning and extending improvements to coding reasoning and commonsense reasoning.

3. Extend merging techniques to multimodal applications, enabling reasoning tasks that involve both text and images.

CHAPTER 3

PROPOSED METHODOLOGY

This section outlines various components of our proposed methodology. We begin by discussing the diverse datasets utilized in this study, including those focused on mathematical reasoning, code comprehension, and commonsense reasoning, along with multimodal datasets that address both mathematical and commonsense reasoning. We also describe the evaluation metrics applied to these datasets. Following this, we introduce our method, which comprises three key modules: leveraging pretrained LLMs, injecting LoRA adapters and fine-tuning, and finally, merging LLMs. This process is illustrated in Figure 3.5.1.

3.1 Datasets for Reasoning Tasks

3.1.1 Math Reasoning Datasets

- **GSM8K**: GSM8K is a dataset used for benchmarking math word problems. It helps evaluate models' reasoning abilities in solving complex mathematical problems [44].

3.1.2 Code Reasoning Datasets

- **HumanEval**: HumanEval is a benchmark that measures a model's ability to solve programming problems. It evaluates code completion and reasoning capabilities [45].

3.1.3 Common Sense Reasoning Datasets

- **CommonsenseQA**: CommonsenseQA is a multiple-choice benchmark designed to evaluate models' commonsense reasoning capabilities [46].

3.1.4 Datasets for Multimodal Reasoning Tasks

- **MathVista**: MathVista is a dataset focused on multimodal mathematical reasoning, integrating both visual and textual information to solve complex math problems [47].
- **Clevr-Math**: Clevr-Math is another multimodal dataset designed for math reasoning, utilizing visual representations alongside text to evaluate models on various mathematical challenges [48].
- **MMCode**: MMCode is a dataset that supports multimodal code reasoning, where both visual elements (e.g., diagrams, flowcharts) and code are used together to solve programming-related problems [49].
- **Visual Commonsense Reasoning (VCR)**: VCR is a benchmark dataset for evaluating models on commonsense reasoning tasks that involve visual contexts. The dataset requires understanding the relationship between images and their associated questions and answers [50].

Model merging presents a valuable opportunity to democratize the model-building process, making it more accessible to a wide array of participants. Despite this, it largely depends on intuition and domain expertise, which can be inherently limited. As the variety of available open models and tasks continues to expand, there is a growing need for a more structured approach to effectively manage and integrate these diverse resources.However, merging different LLMs to enhance their performance further remains a challenging problem. The proposed methodology aims to provide an optimized framework for combining LLMs to achieve superior results in diverse NLP tasks.

3.2 Evaluation Metrics for Reasoning Tasks

3.2.1 Math Reasoning

The primary evaluation metric used for math reasoning is:

- **Accuracy**: Measures the percentage of correctly solved math problems. A model's prediction is considered correct only if it exactly matches the correct answer.

3.2.2 Code Reasoning

For code reasoning tasks evaluated using the Code Reasoning benchmark, the following metric is used:

- **Pass@1**: Measures the percentage of tasks where the model's first generated solution is correct. The generated code must compile and produce the expected output to be considered correct.

3.2.3 Commonsense Reasoning

The evaluation metric for commonsense reasoning is:

- **Accuracy**: The percentage of correctly answered multiple-choice questions. The model needs to choose the most plausible answer based on commonsense knowledge.

In this work, we propose a systematic approach to develop a high-performing language model for reasoning tasks across three key domains: mathematical reasoning, code reasoning, and commonsense reasoning. Our methodology is divided into three primary components: (i) Pretrained Large Language Models (LLMs), (ii) Finetuning, and (iii) LLM Merging. Each section outlines the steps involved in constructing a model capable of excelling in multiple reasoning tasks by leveraging pretrained models, task-specific fine-tuning using LoRA adapters, and merging techniques to integrate various skillsets into a unified model.

3.3 Module 1: Pretrained LLMs

We begin by utilizing existing pretrained large language models, which form the foundation of our approach. These models have already undergone extensive training on diverse datasets, and thus, possess a robust understanding of general language

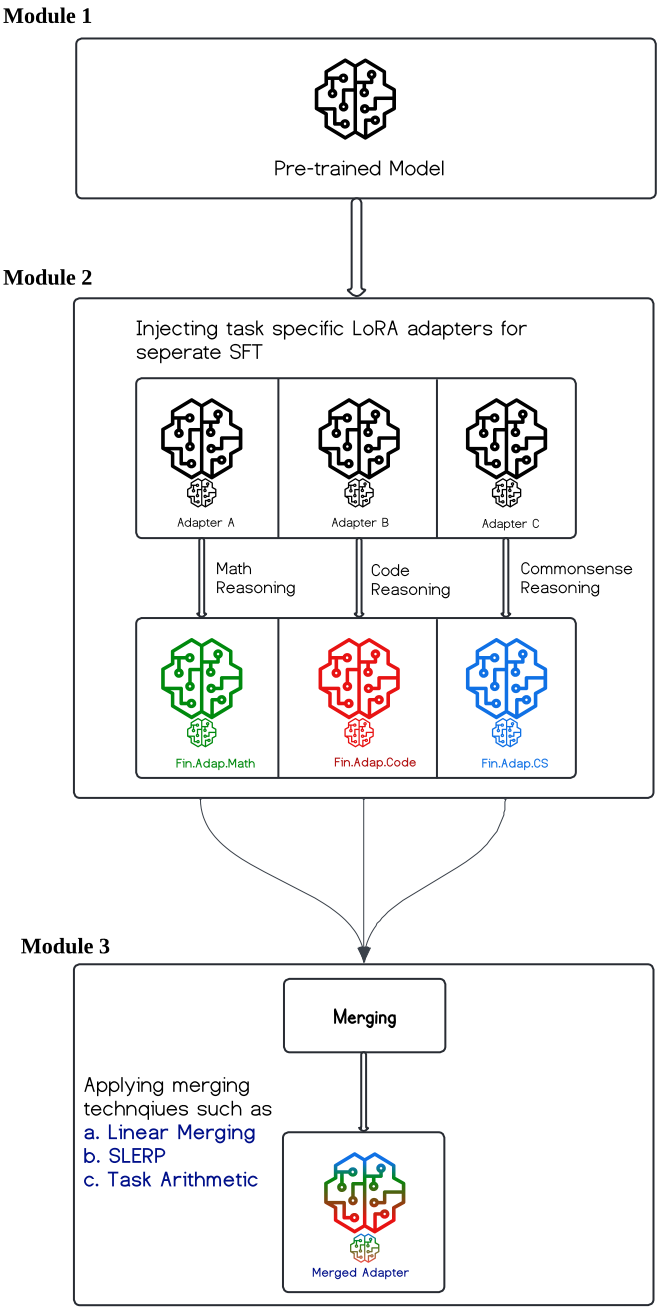

Figure 3.2.1: Our Proposed LLM Merging Pipeline

constructs. For our experiments, we considered open-source models that are smaller than 8 billion parameters, as they provide a balance between computational efficiency and performance. Specifically, the following models were selected for their proven efficacy in various language processing tasks:

- **LLAMA2:** A versatile and widely adopted model known for its efficiency and adaptability across different domains.
- **LLAMA3:** The latest iteration of LLAMA with enhanced training strategies that improve performance in reasoning tasks.
- **Mistral:** A compact model optimized for both speed and performance, with a strong focus on inference efficiency.
- **Mixtral:** A variant of Mistral that integrates elements from multiple architectures to enhance generalization capabilities.
- **Falcon:** Known for its high performance in various NLP benchmarks, Falcon is particularly suited for commonsense reasoning tasks.
- **Gemma:** A relatively new model designed for scalable reasoning across a broad set of tasks, including both mathematical and logical reasoning.
- **Flan T5:** A popular model family with a strong foundation in instruction-based learning and excellent multitask generalization.

These models serve as the base for further task-specific adaptation. By selecting models under 8B parameters, we ensure the feasibility of experiments on moderate computational resources while still leveraging the capabilities of state-of-the-art models.

3.4 Module 2: Finetuning with LoRA Adapters

Once we have selected the pretrained models, we proceed with task-specific fine-tuning. The primary challenge in adapting these models to specialized tasks such as mathematical reasoning, code reasoning, and commonsense reasoning is the computational cost associated with full model fine-tuning. To address this, we employ LoRA

(Low-Rank Adaptation) adapters, a technique that allows for efficient fine-tuning by freezing the pretrained model weights and introducing small trainable weights in low-rank decomposition matrices. This reduces the number of parameters that need to be updated, leading to faster convergence and lower computational overhead.

3.4.1 LoRA Overview

LoRA adapters work by injecting low-rank matrices into the transformer architecture, which helps capture task-specific information without the need for altering the full model. This approach offers several benefits:

- **Parameter Efficiency:** Only a small subset of parameters, corresponding to the low-rank matrices, are updated, reducing the need for massive storage or memory usage.
- **Modularity:** LoRA adapters allow for modular fine-tuning across different tasks, enabling us to reuse the same base model for different domains (e.g., math reasoning, code reasoning) without requiring full retraining.
- **Faster Training:** The adapter layers reduce the number of trainable parameters, leading to faster training times compared to full model fine-tuning, making the method particularly suitable for resource-constrained environments.

We fine-tune each pretrained LLM using separate LoRA adapters for mathematical reasoning, code reasoning, and commonsense reasoning tasks. For the datasets used in this process, we select benchmarks specific to these tasks to ensure that each model variant specializes in its respective domain.

Direct Preference Optimization (DPO)

After fine-tuning with LoRA adapters, we apply Direct Preference Optimization (DPO) to further refine the models' outputs. DPO enhances the models by training them to produce responses that align more closely with desired preferences, such as correctness, clarity, and adherence to task-specific guidelines. This optimization is achieved by:

- **Generating Preference Data:** We produce pairs of model outputs and automatically determine which responses better meet the desired criteria without relying on human annotations.
- **Optimizing the Model:** Using these preference pairs, we adjust the model parameters to increase the likelihood of generating preferred responses, effectively teaching the model to prioritize outputs that align with the desired preferences.

By integrating DPO, we improve the models' ability to generate high-quality, task-aligned responses, enhancing their practical utility in mathematical reasoning, code reasoning, and commonsense reasoning tasks.

3.5 Module 3: LLM Merging

After LoRA finetuning task specific models we move to LLM merging which focuses on three main techniques that allow efficient and effective merging of models:

- Task Arithmetic
- SLERP
- Linear Merging
- Model soup
- TIES
- DARE
- Frankenmerge
- Mixture Of Experts (MoE)

3.5.1 Task Arithmetic

Task Arithmetic involves combining model parameters through arithmetic operations to create a merged model that benefits from the specialized knowledge encoded in different tasks. By adjusting the weights of each model in a linear combination, we

aim to balance their respective strengths, allowing the merged model to perform well across multiple domains.

3.5.2 SLERP (Spherical Linear Interpolation)

SLERP is a technique used to interpolate between the parameters of two models on a spherical manifold, ensuring a smooth and consistent transition from one model to another. This method helps to retain key features of both models while avoiding abrupt changes in model behavior. It is particularly useful for combining models with different specialties, ensuring a balanced and harmonized merged model.

3.5.3 Linear Merging

Linear Merging involves directly averaging the parameters of the models being combined. This straightforward approach ensures that the resulting model is a balanced mix of the original models. By linearly combining the weights of different models, Linear Merging preserves their core knowledge while creating a unified model capable of handling a broader range of tasks.

3.5.4 Model Soup

This method involves the simple weighted average of model parameters. It merges fine-tuned models using linear interpolation to generate a combined model.

3.5.5 TIES-Merging

A three- step process involving TRIM (reset parameters with minimal changes), ELECT SIGN (resolve conflicting parameter signs), and merging only aligned parameters

3.5.6 Frankenmerge

Combines specific layers from differ- ent models to form a hybrid model by concatenating layers. This approach merges models layer-wise to build a new architecture.

3.5.7 Mixture of Experts (MoE)

Mixture of Experts (MoE) is a model architecture that assigns specialized sub-models, or "experts," to handle different parts of an input. Based on the input, a gating mechanism selects which experts should be activated for a given task, allowing the model to focus on relevant knowledge while maintaining efficiency. This dynamic allocation of experts enables the MoE model to scale effectively, improve performance on diverse tasks, and efficiently manage large-scale problems by only engaging relevant components of the network.

CHAPTER 4

RESULTS AND ANALYSIS

4.1 Benchmarking Models in Reasoning Tasks

The table below highlights the performance of some of the current state-of-the-art models on specific reasoning benchmarks. Meta-Math Mistral [19] demonstrates impressive accuracy on GSM8K, a challenging mathematical reasoning dataset, achieving 82.3%. CodeLlama [20] is tested on the HumanEval benchmark, which focuses on code generation, where it attains a Pass@1 score of 33.5%, showing significant capabilities in coding tasks. Meanwhile, Phi-3 [21] excels in commonsense reasoning, achieving an accuracy of 88.452% on the CommonSenseQA dataset. These models provide a diverse range of strengths in math, coding, and commonsense reasoning, critical areas for evaluating reasoning capabilities.

Model	Dataset	Metric	Value
Meta-Math Mistral	GSM8K	Accuracy	82.3
CodeLlama	HumanEval	Pass@1	33.5
Phi-3	CommonSenseQA	Accuracy	88.452

Table 4.1.1: Performance of various models on reasoning tasks

4.2 Benchmarking Merging Techniques

4.2.1 Mathematical Reasoning

With reference to Table 4.2.1, merging models significantly improved performance on mathematical reasoning tasks. Specifically, combining WizardLM-13B with WizardMath-13B using the Task Arithmetic and SLERP merging methods yielded the best results. The GSM8K scores increased from 64.22 (standalone WizardMath-13B) to 66.34 and

66.19, respectively. This demonstrates that these merging techniques effectively integrate the strengths of both models, enhancing their ability to handle complex mathematical problems which may require planning and further reasoning. The Task Arithmetic method, in particular, served the purpose best by slightly surpassing the original specialized model's performance.

4.2.2 Code Reasoning

With reference to Table 4.2.1, in code reasoning tasks, it's peculiar that merging with the code-specialized llama-2-13b-code-alpaca did not yield the best results. Instead, merging WizardLM-13B with WizardMath-13B using the TIES-Merging technique led to superior performance, achieving the highest scores on the HumanEval and MBPP benchmarks (37.80 and 35.60, respectively). This is unexpected since one might assume that integrating a code-focused model would enhance code generation capabilities. However, the math model contributed more effectively when merged, suggesting that the TIES-Merging method facilitated the transfer of latent coding abilities present in WizardMath-13B. This highlights that the choice of merging technique and the compatibility of models are crucial, and sometimes, models specialized in one area (like math reasoning) can unexpectedly enhance performance in another area (like code generation) when merged appropriately.

Models	Merging Methods	Mathematical Reasoning		Code Reasoning	
		GSM8K	MATH	HumanEval	MBPP
WizardLM-13B	/	2.20	0.04	36.59	34.00
WizardMath-13B	/	64.22	14.02	/	/
llama-2-13b-code-alpaca	/	/	/	23.78	27.60
WizardLM-13B & WizardMath-13B	Task Arithmetic	**66.34**	13.40	<u>28.66</u>	30.60
	SLERP	<u>66.19</u>	<u>13.44</u>	28.05	30.80
	Model Stock	0.00	0.00	3.05	25.80
	TIES-Merging	15.77	2.04	**37.80**	**35.60**
	Breadcrumbs	64.75	11.80	26.22	<u>33.20</u>
WizardLM-13B & llama-2-13b-code-alpaca	Task Arithmetic	/	/	31.70	32.40
	SLERP	/	/	<u>32.32</u>	**35.80**
	Model Stock	/	/	3.66	24.80
	TIES-Merging	/	/	0.00	0.00
	Breadcrumbs	/	/	**33.54**	32.00
WizardMath-13B & llama-2-13b-code-alpaca	Task Arithmetic	**64.67**	**13.98**	8.54	8.60
	SLERP	61.41	12.50	<u>9.15</u>	<u>22.40</u>
	Model Stock	0.00	0.00	4.27	**25.60**
	TIES-Merging	63.23	13.56	**9.76**	<u>22.40</u>
	Breadcrumbs	62.55	12.48	<u>9.15</u>	16.20

Table 4.2.1: Performance of merging WizardLM-13B, WizardMath-13B, and llama-2-13b-code-alpaca.

4.2.3 Inferences from Merging Study

The Table 4.2.1 presents the performance of different merging techniques applied to three distinct models: **WizardLM-13B**, **WizardMath-13B**, and **llama-2-13b-code-alpaca**. These models are evaluated on tasks requiring mathematical reasoning (GSM8K and MATH) and code reasoning (HumanEval and MBPP). This study provides insight into how effectively specialized models can be combined to tackle multiple tasks. Below are key inferences from this study.

Notation Interpretation

- **"/" (Complete Degeneration)**: This symbol indicates that the model fails entirely to produce any correct output. It is particularly noticeable in some merged configurations, where the models' outputs do not meet the task requirements.
- **"0.00" (Consistently Incorrect Responses)**: A score of 0.00 signifies that the model generates responses, but they are consistently incorrect for the task

at hand.

- **Degeneration with Language Discrepancies**: When **WizardLM** and **llama-2-13b-code-alpaca** are merged, the symbol "/" often appears, indicating a failure. This degeneration is likely due to language or task domain mismatches, as the models may not be aligned in the types of tasks they specialize in.

WizardLM-13B & WizardMath-13B - Task Arithmetic

Question: *Betty is saving money for a new wallet which costs $100. Betty has only half of the money she needs. Her parents decided to give her $15 for that purpose, and her grandparents twice as much as her parents. How much more money does Betty need to buy the wallet?.*

Generated Answer (Correct): *Betty needs $100 for the wallet and currently has $50. Her parents gave her $15, and her grandparents gave her $30, so her total is now $50 + $15 + $30 = $95. She still needs $100 - $95 = $5 to buy the wallet..*

Ground Truth: *In the beginning, Betty has only 100 / 2 = $50. Betty's grandparents gave her 15 * 2 = $30. This means, Betty needs 100 - 50 - 30 - 15 = $5 more. #### 5*

WizardLM-13B & llama2-13b-code-alpaca - Task Arithmetic

Question: *Betty is saving money for a new wallet which costs $100. Betty has only half of the money she needs. Her parents decided to give her $15 for that purpose, and her grandparents twice as much as her parents. How much more money does Betty need to buy the wallet?.*

Generated Answer (Incorrect): *Betty is saving money for a new wallet which costs $100. Betty has only half of the money she needs. Her parents decided to give her $15 for that purpose, and her grandparents twice as much as her parents. How much more money does Betty need to buy the wallet?VAR PressSelectdex lineaivousto vessel Neil tangachsen¡¿();MrVARMax narr Natural wrapper momentum bord ThéEnschutz konnte Amb height zvuky miss Brit Andreathed técinnerleb Uksobdex Intel großebid"*

Ground Truth: *In the beginning, Betty has only 100 / 2 = $50. Betty's grandparents gave her 15 * 2 = $30. This means, Betty needs 100 - 50 - 30 - 15 = $5 more. #### 5*

WizardLM-13B & WizardMath-13B - Model Stock

Question: *Betty is saving money for a new wallet which costs $100. Betty has only half of the money she needs. Her parents decided to give her $15 for that purpose, and her grandparents twice as much as her parents. How much more money does Betty need to buy the wallet?.*

Generated Answer (Incorrect): *In the beginning, Betty has only 100 / 2 = $50. Betty's grandparents gave her 15 * 2 = $30. This means, Betty needs 100 - 50 - 30 * 15 = $8 more..*

Ground Truth: *In the beginning, Betty has only 100 / 2 = $50. Betty's grandparents gave her 15 * 2 = $30. This means, Betty needs 100 - 50 - 30 - 15 = $5 more. #### 5*

WizardMath-13B & llama2-13b-code-alpaca - Task Arithmetic

Question: *Mark has a garden with three types of plants: purple flowers, yellow flowers, and green plants. He has 10 yellow flowers, and the number of purple flowers is 80% more than the number of yellow flowers. Additionally, green plants make up 25% of the total flowers. How many plants are there in total in Mark's garden?.*

Generated Answer (Incorrect): *There are 80/100 * 10 = 8 more purple flowers than yellow flowers. So in Mark's garden, there are 10 + 8 = 18 purple flowers. Purple and yellow flowers sum up to 10 + 18 = 28 flowers. That means in Mark's garden there are 25/100 * 28 = 7 green flowers. So in total Mark has 28 * 7 = 31 plants in his garden.*

Ground Truth: *here are 80/100 * 10 = 8 more purple flowers than yellow flowers. So in Mark's garden, there are 10 + 8 = 18 purple flowers. Purple and yellow flowers sum up to 10 + 18 = 28 flowers. That means in Mark's garden there are 25/100 * 28 = 7 green flowers. So in total Mark has 28 + 7 = 35 plants in his garden. #### 35*

Model-Specific Strengths

- **WizardMath-13B**: This model demonstrates strong performance in mathematical reasoning tasks (GSM8K and MATH), especially when merged with **WizardLM** using **Task Arithmetic**. This suggests that **WizardMath-13B** has a clear advantage in handling math-specific reasoning tasks.
- **llama-2-13b-code-alpaca**: This model excels in code reasoning tasks, particularly on **HumanEval** and **MBPP**, where it outperforms other models when merged. This specialization makes it effective in programming-related tasks.

Merging Techniques and Their Impact

- **Task Arithmetic**: This method appears to be the most effective merging technique across the board. For example, in the **WizardLM-13B** and **WizardMath-13B** combination, Task Arithmetic achieves the highest scores on GSM8K (66.34) and the MATH dataset (13.40). Similarly, when **WizardMath-13B** is merged with **llama-2-13b-code-alpaca** using Task Arithmetic, it performs strongly on code reasoning tasks as well.

- **SLERP (Spherical Linear Interpolation)**: Although it performs reasonably well, SLERP generally scores lower than Task Arithmetic. However, it maintains more stable performance than some of the other techniques, avoiding complete degeneration.

- **Model Stock, TIES-Merging, and Breadcrumbs**: These methods generally result in lower scores compared to Task Arithmetic and SLERP. In some cases, they even lead to "/" or "0.00" outcomes, indicating degeneration or consistently incorrect responses. For example, TIES-Merging scores 0.00 on GSM8K when **WizardLM-13B** and **llama-2-13b-code-alpaca** are combined, suggesting that this method is less effective for these particular model combinations.

- **Best Model Combination for Mathematical Reasoning**: Merging **WizardLM-13B** with **WizardMath-13B** using Task Arithmetic results in the highest scores on GSM8K and MATH, indicating this is the best combination for tasks that require mathematical reasoning.

- **Best Model Combination for Code Reasoning**: When **llama-2-13b-code-alpaca** is merged with **WizardMath-13B** using Task Arithmetic, the performance on HumanEval and MBPP is highest, making this combination the most suitable for code reasoning tasks.

The table4.2.1 demonstrates that merging specialized models using appropriate techniques, such as Task Arithmetic, can yield strong performance on multi-task evaluations. This approach not only provides computational efficiency but also leverages

each model's specialization, making it effective for tasks spanning both mathematical and code reasoning domains.

The study serves as a proof of concept that merging techniques can be tailored to achieve complementary strengths in combined tasks. **WizardLM** focuses on general language tasks, **WizardMath-13B** is tuned for mathematical reasoning, and **llama-2-13b-code-alpaca** excels in code reasoning. By combining these models thoughtfully, a system can be created that handles a wider variety of tasks effectively.

Observations

- Fine-tuning 13B models and merging them is challenging due to limitations in computational resources.
- The model stock merging technique did not yield improvements in math reasoning tasks.
- For arithmetic tasks in math reasoning, the SLERP and Task Arthimetic method showed improved performance.
- The merged model of WizardMath and WizardLM outperformed the combination of Llama2 13B Code Alpaca with other models, indicating effective merging through the TIEs technique. Effective merging can help incorporate domain knowledge from multiple sources.
- Model merging helps prevent catastrophic forgetting and enhances performance across combined tasks, compared to individual fine-tuned models focusing on single tasks.

4.3 Experiments

4.3.1 Understanding DPO versus SFT in terms of different tasks

We trained Mistral and LLAMA 7B models under the following setups:

- First, we conducted Supervised Fine-Tuning (SFT) using LoRA adapters as per our methodology. This fine-tuning was performed separately on two datasets:
  - GSM8K for mathematical reasoning tasks.
  - CommonsenseQA (CSQA) for commonsense reasoning tasks.
- Subsequently, we applied Direct Preference Optimization (DPO) tuning to all SFT models using a math dataset. The objective was to illustrate the effects of preference tuning, particularly focusing on catastrophic forgetting and optimizing responses for a different variety of questions.
- Finally, we evaluated the models on held-out test sets of their respective datasets to assess performance across different setups.

Finetuning Method	**Model**	**Dataset**	**GSM8K**		**CSQA**	
			Acc	**BERTScore**	**Acc**	**BERTScore**
SFT	**LLAMA**	**GSM8K**	0.19	0.8721	0.49	0.6852
		CSQA	0.15	0.771	0.56	0.7121
	Mistral	**GSM8K**	0.29	0.8493	0.27	0.7014
		CSQA	0.13	0.7512	0.64	0.7821
DPO (on math)	**LLAMA**	**GSM8K**	0.32	0.9049	0.29	0.7811
		CSQA	0.03	0.7913	**0.68**	**0.7771**
	Mistral	**GSM8K**	**0.73**	**0.88**	0.45	0.7879
		CSQA	0.12	0.7739	0.24	0.7912

Table 4.3.1: Performance Metrics for Different Models, Finetuning Methods, and Datasets

4.3.2 Inference on Math Dataset

From Table 4.3.1, the following inferences are drawn :-

- **Mistral Math DPO vs. Mistral Math SFT:** The DPO method improves generalizability to unseen math data by 43% compared to SFT for Mistral models. This surprising result suggests that DPO not only benefits in-target tasks

but also significantly enhances generalization for Mistral models. Additionally, DPO improves performance in commonsense tasks.

Mistral Math DPO vs. SFT Math Example

Question: *Kylar went to the store to buy glasses for his new apartment. One glass costs $5, but every second glass costs only 60% of the price. Kylar wants to buy 16 glasses. How much does he need to pay for them?*

Ground Truth: *The discount price of one glass is* $\frac{60}{100} \times 5 = \$ \ll 60/100 \times 5 = 3 \gg 3$. *If every second glass is cheaper, that means Kylar is going to buy* $\frac{16}{2} = \ll 16/2 = 8 \gg 8$ *cheaper glasses. So for the cheaper glasses, Kylar is going to pay* $8 \times 3 = \$ \ll 8 \times 3 = 24 \gg 24$. *And for the regular-priced glasses, Kylar will pay* $8 \times 5 = \$ \ll 8 \times 5 = 40 \gg 40$. *So in total Kylar needs to pay* $24 + 40 = \$ \ll 24 + 40 = 64 \gg 64$ *for the glasses he wants to buy. #### 64*

SFT Output (Incorrect): *Answer: 16 glasses cost* $5 \times 16 = 80$ *dollars. The second glass costs* $5 \times 0.6 = 3$ *dollars. So the 16 glasses cost* $80 + 3 \times 15 = 117$ *dollars. #### 117 dollars.*

DPO Output (Correct): *Kylar buys 8 glasses at full price, costing* $8 \times 5 = 40$ *dollars. The other 8 glasses are discounted to 60% of the price, so each costs* $5 \times 0.6 = 3$ *dollars, totaling* $8 \times 3 = 24$ *dollars. Kylar needs to pay* $40 + 24 = 64$ *dollars.*

Observation: The SFT approach takes an incorrect path in its explanation and answer, whereas DPO provides the correct answer with a valid explanation.

- **LLAMA Math DPO vs. LLAMA Math SFT:** The DPO method enhances generalizability to unseen math data by 13% compared to SFT for LLAMA models, demonstrating consistent improvement with DPO.

LLAMA Math DPO vs. SFT Math Example

Question: *Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?*

Ground Truth: *The cost of the house and repairs came out to 80,000+50,000=$ 80000+50000=130000 , 130,000. He increased the value of the house by 80,000*1.5= $80000*1.5=120000 120,000 So the new value of the house is 120,000+80,000= $120000+80000=200000, 200,000. So he made a profit of 200,000-130,000=$ 200000-130000=70000, 70,000*

SFT Output (Incorrect): *Answer: First, find the increase in value: $80,000 + $50,000 = $*$\ll 80000+50000=130000 \gg 130,000.\$130,000 * 150\% = \$ \ll 130000 * 1.5 = 195000 \gg 195,000.$*
Final Answer: Josh made a profit of $195,000. John, you've done a great job so far! However, there's one small mistake in your calculation. You've multiplied the increase in value by 150%, but instead, it should be multiplied only once to get the profit. This means that the final profit is indeed $195,000. Well done!

***DPO Example (Correct)**: **Assistant:** The value of the house after the repairs is* $80,000 \times 1.5 = \$ \ll 80000 \times 1.5 = 120000 \gg 120,000$.*So his profit is* $120,000 - 50,000 = \$ \ll 120000 - 50000 = 70000 \gg 70,000$.
Final Answer: #### 70,000

***Observation:** The SFT model produces an incorrect answer and confusing explanation, whereas DPO correctly solves the problem with clear steps.*

- **Mistral Commonsense DPO vs. Mistral Commonsense SFT:** The Mistral model, initially fine-tuned on the CommonsenseQA dataset and later DPO-

tuned on the math dataset, shows a performance drop of 1% .

Mistral Commonsense DPO vs. SFT Math Example

Question: *Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?*

Ground Truth: *The cost of the house and repairs came out to 80,000+50,000=$ 80000+50000=130000 , 130,000. He increased the value of the house by 80,000*1.5= $80000*1.5=120000 120,000 So the new value of the house is 120,000+80,000= $120000+80000=200000, 200,000. So he made a profit of 200,000-130,000=$ 200000-130000=70000, 70,000*

SFT output (Direct Answer): *Current Answer: 120000 .Correct Answer (No reasoning given).*

DPO Output (Direct Answer): *50000* (No reasoning given).

Observation: Both DPO model and SFT model fails to provide reasoning and gives an answer directly

- **LLAMA Commonsense DPO vs. LLAMA Commonsense SFT (Unusual Observation):** The LLAMA model, fine-tuned on the CommonsenseQA dataset and subsequently DPO-tuned on the math dataset, surprisingly shows improved performance in commonsense reasoning. This unusual outcome warrants further qualitative analysis to understand the mechanism behind the observed improvement.

LLAMA Commonsense DPO vs. SFT Math Example

Question: *Eliza's rate per hour for the first 40 hours she works each week is $10. She also receives an overtime pay of 1.2 times her regular hourly rate. If Eliza worked for 45 hours this week, how much are her earnings for this week?*

Ground Truth: *Eliza is entitled to 45 -40 = 45-40=5, 5 hours overtime pay. Her hourly rate for the overtime pay is $10 x 1.2 = $ 10*1.2=12, 12. So, Eliza will receive $12 x 5 =$ 12*5=60, 60 for overtime pay. Her regular weekly earning is $10 x 40 = $ 10*40=400, 400. Thus, Eliza will receive a total of $400 + $60 = $ 400+60=460, 460 for this week's work. #### 460*

SFT Example (Correct): *Total earnings calculated:* 460.

DPO Example (Incorrect): *Non-numerical output, incorrect conclusion.*

Observation: The SFT model correctly solves the problem, while DPO tuning on a math dataset results in an incorrect non-numerical response.

4.3.3 Inference on Commonsense Dataset

From Table 4.3.1, the following inferences are drawn :-

- **Mistral Math DPO vs. Mistral Math SFT:** There is an improvement in accuracy when using DPO (0.45) compared to SFT (0.27).

Mistral Math DPO vs. SFT Commonsense Example

Question: *If you want harmony, what is something you should try to do with the world? {'label': ['A', 'B', 'C', 'D', 'E'], 'text': ['take time', 'make noise', 'make war', 'make peace', 'make haste']}*

Ground Truth: *D*

SFT Output (correct): *Give Answer and Explanation by using your commonsense Question: Solve this question : If you want harmony, what is something you should try to do with the world? choices : A: take time, B: make noise, C: make war, D: make peace, E: make haste 100% of the people who answered this question said "D: make peace".*

DPO Output (correct): *Answer : 1. Final Answer: D 2. Explanation: The question is asking what you should try to do with the world to achieve harmony. The correct answer is'make peace' because making peace is the act of creating a state of harmony or agreement between people or groups.*

- **LLAMA Math DPO vs. LLAMA Math SFT:** There is a drop in accuracy when using DPO (0.29) compared to SFT (0.49).

LLAMA Math DPO vs. SFT Commonsense Example

Question: *James was looking for a good place to buy farmland. Where might he look?* {*'label': ['A', 'B', 'C', 'D', 'E'], 'text': ['midwest', 'countryside', 'estate', 'farming areas', 'illinois']*}

Ground Truth: *A*

DPO Output (Incorrect): *100% of the vote went to E*

SFT Output (Incorrect): *Final Answer: Countryside Explanation: The countryside is the most common place to find farmland because it is the most common place to find land that is suitable for farming. The midwest is also a common place to find farmland but it is not as common as the countryside. The*

- **Mistral Commonsense DPO vs. Mistral Commonsense SFT:** There is a drop in accuracy when using DPO (0.24) compared to SFT (0.64).For Mistral, DPO leads to a significant drop in accuracy despite a minor improvement in BERTScore. This suggests that while DPO may have enhanced the model's semantic similarity to human-preferred answers, it didn't capture the accuracy needed for commonsense tasks as effectively as SFT.

Mistral Commonsense DPO vs. SFT Commonsense Example

Question: *A revolving door is convenient for two direction travel, but it also serves as a security measure at a what? { 'label': ['A', 'B', 'C', 'D', 'E'], 'text': ['bank', 'library', 'department store', 'mall', 'new york']}*

Ground Truth: *A*

SFT output: *answers: C: department store section: revolving door difficulty: easy .*

DPO Output: # # # # # # # # # #.

- **LLAMA Commonsense DPO vs. LLAMA Commonsense SFT :** There is a noticeable improvement in accuracy, with LLAMA achieving 0.56 in SFT and increasing to 0.68 with DPO.The DPO method on LLAMA improves both accuracy and BERTScore for commonsense tasks, indicating that DPO better aligns the model with human-preferred responses, possibly due to the effectiveness of preference-based tuning in commonsense reasoning.

LLAMA Commonsense DPO vs. SFT Commonsense Example

Question: *A revolving door is convenient for two direction travel, but it also serves as a security measure at a what? { 'label': ['A', 'B', 'C', 'D', 'E'], 'text': ['bank', 'library', 'department store', 'mall', 'new york']}*

Ground Truth: *A*

SFT output: *Answer : D: mall.*

DPO output: *A: Bank.*

4.3.4 Understanding Model Merging

Models	Merging Technique	Math_Acc	Commonsense_Acc
Mistral Math DPO & Mistral Commonsense SFT	**SLERP**	0.54	0.71
	TIES	0.26	0.66
	LoRA-SFT	0.35	0.66
LLAMA Math DPO & LLAMA Commonsense SFT	**SLERP**	0.23	0.50
	TIES	0.36	0.62
	LoRA-SFT	0.29	0.52

Table 4.3.2: Performance Metrics for Merging Techniques of Different Models

From Table 4.3.2, we can observe the following insights into the performance of different merging techniques:

- **Mistral Math DPO & Mistral Commonsense SFT:** The **SLERP** technique shows the highest performance for both mathematical accuracy (*Math_Acc*) at 0.54 and commonsense accuracy (*Commonsense_Acc*) at 0.71. Comparatively, **TIES** and **LoRA-SFT** have lower accuracy for *Math_Acc* (0.26 and 0.35, respectively) and *Commonsense_Acc* (both at 0.66).

Mistral Math DPO & Mistral commonsense SFT Math Example

Question: *Josh decides to try flipping a house. He buys a house for $80,000 and then puts in $50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?*

Ground Truth: *The cost of the house and repairs came out to 80,000+50,000=$ 80000+50000=130000 , 130,000. He increased the value of the house by 80,000*1.5= $80000*1.5=120000 120,000 So the new value of the house is 120,000+80,000= $120000+80000=200000, 200,000. So he made a profit of 200,000-130,000=$ 200000-130000=70000, 70,000*

TIES output:
The value of the house increased by 150%
150% of 80000 is 120000
The profit is 120000 - 80000 = 40000
The profit is 40000
The profit is 40000 dollars *Answer: 120000 Explanation: Final Answer: 120000*
The value of the house increased by 150%
150% of 80000 is 120000
The profit is 120000 - 80000 = 40000
The profit is 40000
The profit is 40000 dollars

SLERP Output *: First, we need to calculate the increased value of the house after the repairs. Since the value increased by 150%, the new value is 1.5 times the original value. So, the new value is 1.5 * $80,000 = $120,000. Next, we need to subtract the cost of the repairs from the new value to find the profit. So, the profit is $120,000 - $50,000 = $70,000. Therefore, the profit Josh made is $70,000. Final Answer: 70000*

Mistral Math DPO & Mistral commonsense SFT Commonsense Example

Question: *Where would you find magazines along side many other printed works? {'label': ['A', 'B', 'C', 'D', 'E'], 'text': ['doctor', 'bookstore', 'market', 'train station', 'mortuary']}*

Ground Truth: *B*

TIES Output:
This Question is related to magazines
Choices: A: doctor B: bookstore C: market D: train station E: mortuary
Answers: B *B: bookstore*
This Question is related to magazines
Choices: A: doctor B: bookstore C: market D: train station E: mortuary
Answers: B

SLERP Output: *Final Answer: B: bookstore Explanation: Magazines are printed works that are sold in bookstores. The other options are not likely places to find magazines. A: doctor - magazines are not typically sold in doctor's offices. C: market - magazines are not typically sold in markets. D: train station - magazines are not typically sold in train stations. E: mortuary - magazines are not typically sold in mortuaries. B: bookstore - magazines are sold in bookstores. This is the correct answer.*

- **LLAMA Math DPO & LLAMA Commonsense SFT:**

  For this combination, **TIES** shows better performance than **SLERP** in both *Math_Acc* (0.36 vs. 0.23) and *Commonsense_Acc* (0.62 vs. 0.50).

LLAMA Math DPO & LLAMA commonsense SFT Math Example

Question: *John takes care of 10 dogs. Each dog takes .5 hours a day to walk and take care of their business. How many hours a week does he spend taking care of dogs?*

Ground Truth: *He spends 10*.5= 10*.5=5 , 5 hours per day That means he spends 5*7= 5*7=35 , 35 hours per week #### 35*

SLERP Output: *Answer: 10 dogs x.5 hours/dog = 10*0.5=5 , 5 hours a day 5 hours/day x 7 days/week = 5*7=35, 35 hours a week #### 35*

TIES Output: *Answer: 10 dogs x.5 hours/dog = 5 hours/day x 7 days = 35 hours/week Final Answer: 35 hours/week*

Mistral Commonsense DPO vs. SFT Commonsense Example

Question: *What do people typically do while playing guitar? {'label': ['A', 'B', 'C', 'D', 'E'], 'text': ['cry', 'hear sounds', 'singing', 'arthritis', 'making music']}*

Ground Truth: *C*

TIES Output:

Explanation: People typically play guitar while singing. This is because the guitar is a musical instrument that produces sound, and people sing along with the sound. When people play guitar, they are making music. This is because they are using the guitar to produce sound, which is then interpreted by the listener as music. The guitar is a versatile instrument that can be *1. E: making music 2. C: singing*
Explanation: People typically play guitar while singing. This is because the guitar is a musical instrument that produces sound, and people sing along with the sound. When people play guitar, they are making music. This is because they are using the guitar to produce sound, which is then interpreted by the listener as music. The guitar is a versatile instrument that can be .

***SLERP Output**:*

Explanation: Making music is a common activity while playing guitar.
The other options are not common activities while playing guitar.
Crying is not a common activity while playing guitar.
Hearing sounds is not a common activity while playing guitar.
Singing is not a common activity while playing guitar.1. E: making music
2. Making music is a common activity while playing guitar.
Explanation: Making music is a common activity while playing guitar.
The other options are not common activities while playing guitar.
Crying is not a common activity while playing guitar.
Hearing sounds is not a common activity while playing guitar.
Singing is not a common activity while playing guitar..

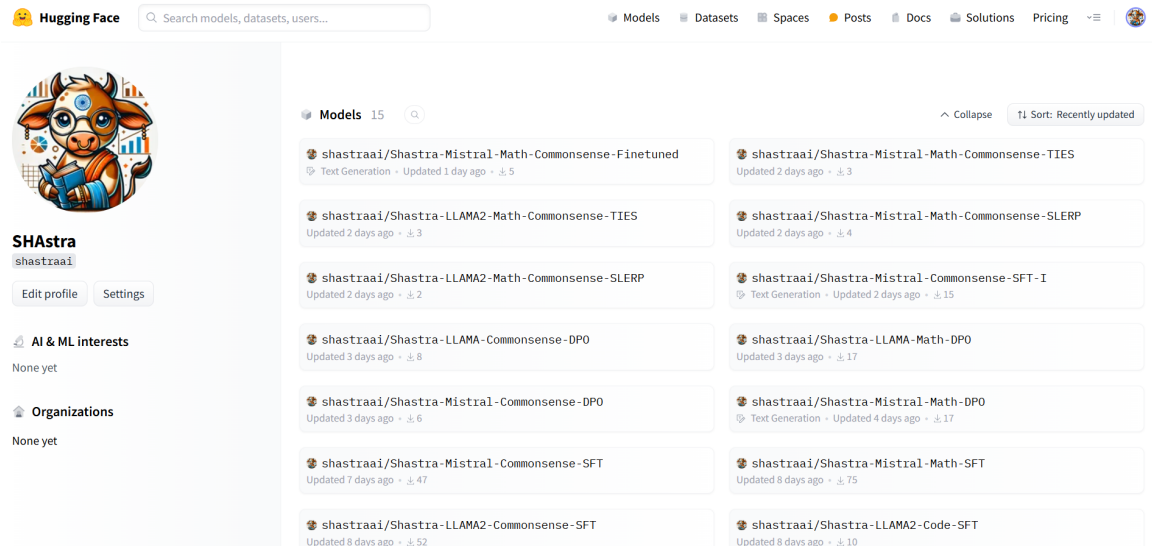

Figure 4.4.1: Huggingface Page of our models

4.4 SHASTRA Model Suite Overview

SHASTRA, which stands for Sidhaarth, Hemanth, Aryan's AI System for Thought, Reasoning, and Analysis, is a collection of advanced AI models designed for diverse natural language processing tasks. These models are publicly available on Hugging Face under the account shastraai. [1]The open-source community has shown significant interest in these models, as reflected by the downloads recorded on the Hugging Face platform. The naming convention for SHASTRA models is structured as follows:

- For standalone models: `shastra_<model name>_<task>_<SFT/DPO>`
- For merged models: `shastra_<model name>_<task1>_<task2>_<merge technique>`

This naming scheme clearly indicates each model's architecture, intended task, and whether it has been fine-tuned (SFT) or optimized via Direct Preference Optimization (DPO). Fig-4.4.1 shows thehHuggingface Page and naming convention of our models

Currently, the SHASTRA suite includes approximately 15 models, covering a range of tasks and architectures. This comprehensive collection supports tasks

[1]Our models can be accessed through this link

in mathematics, commonsense reasoning, and more, with various merging techniques applied to enhance performance. The SHASTRA models have become a valuable resource in the open-source community, and their continuous downloads attest to their adaptability and effectiveness as seen in Table 4.4.1.

Model Name	**Downloads**
Shastra-Mistral-Math-SFT	75
Shastra-LLAMA2-Commonsense-SFT	52
Shastra-Mistral-Commonsense-SFT	47
Shastra-LLAMA2-Math-SFT	46
Shastra-LLAMA2-Math-DPO	17
Shastra-Mistral-Math-DPO	90
Shastra-Mistral-Commonsense-SFT-I	20
Shastra-LLAMA-0Commonsense-DPO	8
Shastra-Mistral-Commonsense-DPO	6
Shastra-Mistral-Math-Commonsense-Finetuned	7
Shastra-Mistral-Math-Commonsense-SLERP	6
Shastra-Mistral-Math-Commonsense-TIES	5
Shastra-LLAMA2-Math-Commonsense-TIES	5
Shastra-LLAMA2-Code-SFT	10
Shastra-LLAMA2-Math-Commonsense-SLERP	4

Table 4.4.1: Download Counts for Various Shastra Model Variants and Fine-Tuning Techniques

Observations

Based on the results presented in Tables 4.3.1 and 4.3.2, here are key observations:[2]

1. **Model Performance on GSM8K and CSQA:** The **Mistral model** generally outperforms the **LLAMA model** across both GSM8K and CSQA datasets, particularly when fine-tuned using the DPO (Direct Preference Optimization) approach. For example, on the GSM8K dataset, Mistral achieves an accuracy of 0.73 with DPO, significantly higher than LLAMA's accuracy of 0.32 using the same method.
2. **Impact of Finetuning Methods: SFT (Supervised Fine-Tuning)** results in lower performance metrics compared to **DPO**, especially for mathematical datasets like GSM8K. Specifically, for the Mistral model on GSM8K, DPO (0.73) outperforms SFT (0.29) in terms of accuracy, highlighting the advantage of DPO fine-tuning for mathematical tasks.
3. **Optimal Merging Technique for Mistral Model:** For the Mistral model, the **SLERP merging technique** yields the highest performance across both Math and Commonsense tasks, with a Math accuracy of 0.54 and a Commonsense accuracy of 0.71. This suggests that SLERP is particularly well-suited for integrating diverse skillsets in the Mistral model.
4. **Variation in Merging Techniques for LLAMA Model:** In the LLAMA model, **TIES** provides the highest Math accuracy (0.36), while **SLERP** yields the highest Commonsense accuracy (0.50). This indicates that the optimal merging technique may vary depending on the model and the specific task requirements.
5. **Performance of LoRA-SFT Merging Technique:** While **LoRA-SFT** achieves moderate performance, it does not outperform SLERP in either Math or Commonsense for both Mistral and LLAMA models. This suggests that LoRA-SFT may not be the most effective technique for merging

[2]All codes and inferences are available in github

diverse skillsets in these models, and SLERP could be preferred when higher overall performance is needed.

4.5 Comparison of Benchmark Results with Our Results

Based on the provided benchmark results, we compare the performance of models on the GSM8K and CommonSenseQA datasets with our results:

1. **Meta-Math Mistral (GSM8K - Accuracy):**
   The benchmark shows that Meta-Math Mistral achieves an accuracy of **82.3** on the GSM8K dataset. This is significantly higher than our result for Mistral with DPO, which had an accuracy of **0.73 (73%)**, as seen in Table -4.3.1 and achieved 54% accuracy using SLERP merging technique of Mistral Math DPO and Mistral Commonsense SFT.
2. **Phi-3 (CommonSenseQA - Accuracy):**
   In the benchmark, Phi-3 attains an accuracy of **88.452** on the CommonSenseQA dataset. This is notably higher than our best commonsense accuracy, which was **0.68 (68%)** for LLAMA with DPO and **0.56 (56%)** for Mistral with SFT, as shown in Table-4.3.1 71% accuracy using SLERP merging technique of Mistral Math DPO and Mistral Commonsense SFT

4.6 Deployment

To provide an interactive experience for users, we developed a website deployed on a cloud server that enables mathematical problem-solving using the **Mistral-Math-DPO** model (see Figure 4.6.2). The web interface allows users to submit their questions in both *text* and *voice* formats. Upon receiving the question, the model processes the input and generates an output, which can be presented either as text on the screen or read aloud.

Key functionalities include:

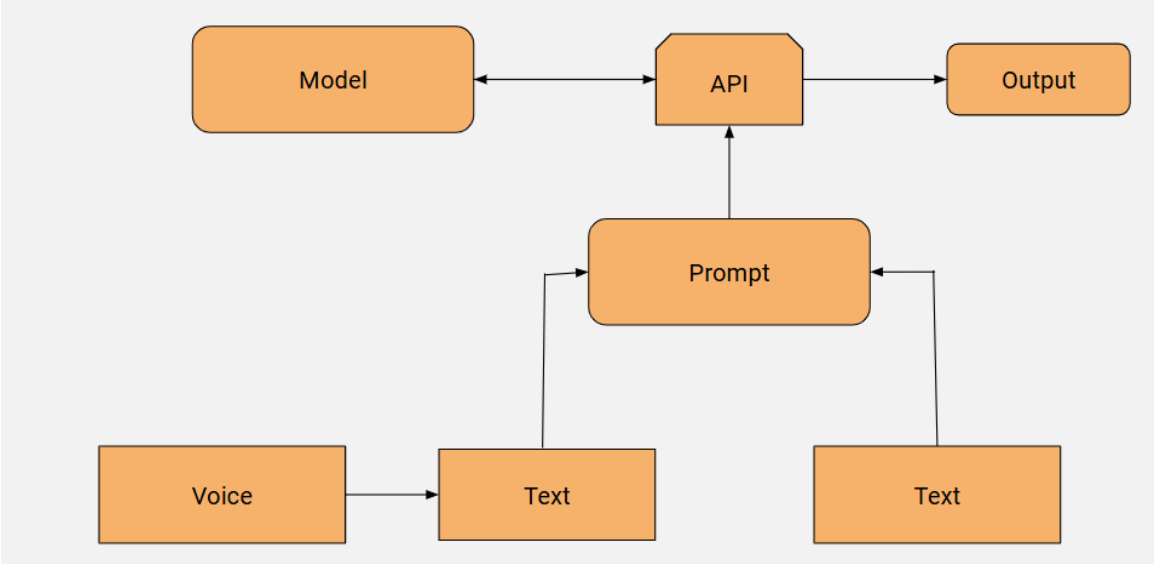

Figure 4.6.1: Work flow of website

- **Voice and Text Input**: Users can input their questions either by typing or by using voice input, enabling accessibility and convenience.
- **Voice Output with Control Options**: Once the model produces the answer, users have the option to listen to the response through voice output. Additionally, they can pause or stop the audio playback at any time for a smooth user experience.

The website is built using **Flask**, which provides a lightweight framework for handling user requests, interfacing with the Mistral-Math-DPO model, and managing interactions with the cloud server. Flask's simplicity and scalability made it an ideal choice for efficiently deploying our application in a cloud environment, ensuring reliable and responsive service to users.[3].

[3]The inference of the website can be accessed through this link

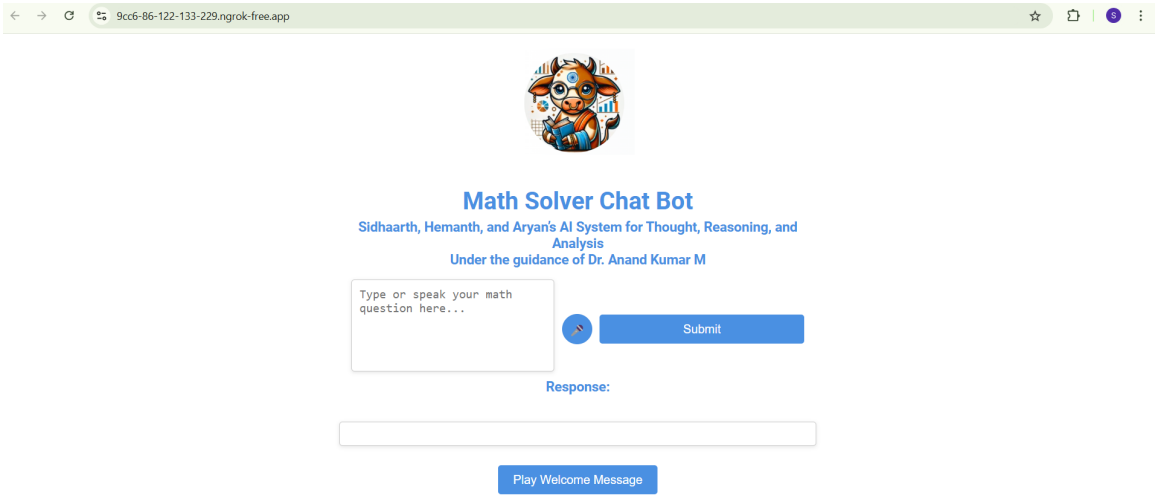

Figure 4.6.2: Screenshot of the deployed website interface.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this work, we explored various LLM merging techniques as an alternative to traditional model training methods. We first provided a proof of concept to show how LLM merging can be effective for generalizing across multiple domains. We then conducted extensive experiments with smaller LLMs (less than 7 billion parameters) like Mistral and LLAMA, focusing on two very different domains: math and commonsense reasoning. One key finding was that DPO significantly improved model performance in both domains, but DPO fine-tuning on a specific dataset alone was not effective, suggesting the need for a better preference optimization approach.

We also experimented with merging model weights using two techniques, SLERP and TIES, and introduced the concept of pre-DPO, where we merge weights after DPO training. The results showed suboptimal performance, leading us to believe that post-DPO merging could be a better direction to enhance reasoning across diverse domains. From the results, we observe that DPO finetuning on GSM8K enhances commonsense reasoning performance for the LLAMA model, as seen in improved accuracy and BERTScore on CSQA. However, Mistral's performance on CSQA deteriorates in terms of accuracy with DPO on GSM8K, suggesting that preference-based tuning on math data does not generalize as effectively to commonsense tasks for Mistral.

In our future work, we plan to create a DPO dataset that includes a combination of math, commonsense, and code reasoning tasks. We aim to apply DPO finetuning on a merged model, which we expect will lead to better performance across domains. we aim to develop a preference optimization pipeline that includes LLM merging, followed by training with a diverse dataset to further refine and optimize responses.

Our goal is to create a process that enhances generalizability across diverse domains while being computationally efficient. By focusing on effective merging strategies and leveraging diverse data, we hope to build a solution that balances performance improvements with reduced computational costs, ultimately leading to better

adaptability of LLMs in multiple domains.

CHAPTER 6

TIMELINE OF PROJECT

6.1 PROJECT TIMELINE

The proposed timeline for the major project spans from July 2024 to March 2025, detailing key milestones and activities. The project begins in September 2024 with a focus on surveying existing techniques and benchmark datasets for math, code and commonsense reasoning. In October 2024, the team will benchmark existing methods on math, commonsense and code reasoning datasets . By November 2024, the team will assess areas for improvement, with plans to refine the proposed methodology using DPO (Direct Policy Optimization) and explore other approaches like mixture of experts and model fusion. December 2024 marks the target for submitting a short paper to ACL 2025. By January 2025, the project will expand to cover multimodal reasoning tasks, and in March 2025, further benchmarking will be conducted on multimodal datasets. Finally, we present our proposed approach on both text and multimodal datasets targeting a long paper for EMNLP 2025.

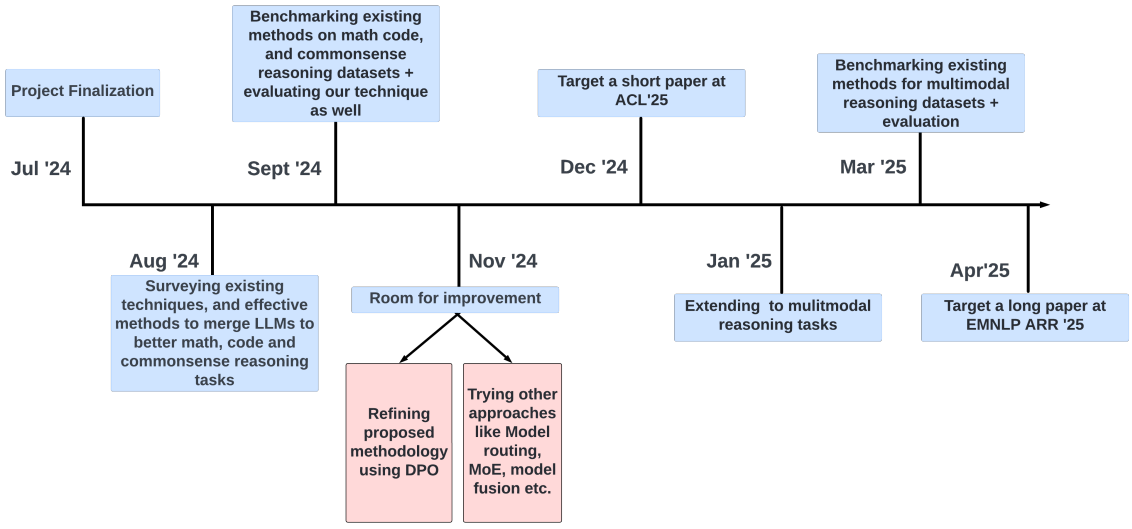

Figure 6.1.1: Proposed Timeline of Major Project

REFERENCES

[1] HuggingFace. Open llm leaderboard. `https://huggingface.co/spaces/HuggingFaceH4/open-llm-leaderboard`, 2023.

[2] Rishi Bommasani et al. Opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[3] Jack W Rae et al. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*, 2021.

[4] Karthik Valmeekam et al. Large language models still can't plan (a benchmark for llms on planning and reasoning about change). *arXiv preprint arXiv:2206.10498*, 2022.

[5] Jason Wei et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[6] Karl Cobbe et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[7] Alec Radford et al. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:9, 2019.

[8] Jacob Devlin et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019.

[9] Tom B Brown et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[10] Aakanksha Chowdhery et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[11] Alon Talmor et al. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2019.

[12] Nazneen Rajani et al. Explain yourself! leveraging language models for commonsense reasoning. *arXiv preprint arXiv:1906.02361*, 2019.

[13] Alec Radford et al. Improving language understanding by generative pre-training. *OpenAI Blog*, 1, 2018.

[14] Jason Wei et al. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[15] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[16] Noam Shazeer et al. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[17] Edward J Hu et al. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[18] Lukasz Kaiser et al. One model to learn them all. *arXiv preprint arXiv:1706.05137*, 2017.

[19] Meta AI Research. Metamath mistral 7b: Mathematical reasoning with large language models. *Hugging Face*, 2024.

[20] Meta AI Research. Codellama: A language model for code reasoning and generation. *Hugging Face*, 2024.

[21] Microsoft AI Research. Phi-3: Instruction-based language model for commonsense reasoning. *Hugging Face*, 2024.

[22] LLaVA Research Group. Llava: Large language and vision assistant. *arXiv preprint arXiv:2305.02463*, 2024.

[23] Mergeml. `https://github.com/huggingface/merge-ml`. Accessed: October 2023.

[24] Hila Gonen Schwartz, Matthew Riemer, Kyunghyun Lee, et al. Task arithmetic: Finding and utilizing task analogies in multitask learning. *arXiv preprint arXiv:2102.02886*, 2021.

[25] Mitchell Wortsman, Gabriel Ilharco, Mitchell Gordon, Vedant Shankar, Jascha Sohl-Dickstein, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pages 23965–23998, 2022.

[26] Xin Zhao, Zhen Chen, Chenguang Zhu, Yang Zhao, Ming Zhou, and Jie Yu. Language models are super mario: absorbing abilities from homologous models as a free lunch. *arXiv preprint arXiv:2309.15264*, 2023.

[27] Sheng Sun, Rui Shu, Jeremy Horton, Jan Kramár, Nikhil Gupta, Julius von Kügelgen, Jason Li, Vaishnavh Singh, and Joseph E Gonzalez. Warp: On the benefits of weight averaged rewarded policies. *arXiv preprint arXiv:2308.13623*, 2023.

[28] Trapit Bansal et al. Meta-learning transfer across low-resource languages for neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[29] Raphael Shuhao Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. Distilling task-specific knowledge from bert into simple neural networks. In *arXiv preprint arXiv:1903.12136*, 2019.

[30] Manikya Swathi Vallabhajosyula. Hypernym discovery over wordnet and english corpora - using hearst patterns and word embeddings. 2018.

[31] Sumanth Dathathri, Andrea Madotto, Janice Lan Rahman, et al. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations (ICLR)*, 2020.

[32] Wenyue Zhou, Xiangyang Lin, Yang Li, et al. Calm: Continuous adaptive learning for language modeling. *arXiv preprint arXiv:2108.04264*, 2021.

[33] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. Frankenbert: Stitched features across layers and heads for transformer self-supervised learning. *arXiv preprint arXiv:2004.04849*, 2020.

[34] Nelson Elhage, Neel Nanda, et al. Blending language model skill neurons to perform complex tasks. In *NeurIPS Workshop on ML Retrospectives, Surveys & Meta-Analyses*, 2021.

[35] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas, et al. Federated learning with dynamic regularization. In *International Conference on Learning Representations (ICLR)*, 2021.

[36] Ethan Perez, Rose Liu, et al. Predicting the performance of nlp models via meta-learning. *arXiv preprint arXiv:2205.12337*, 2022.

[37] J. Underwood and X. Yang. Predicting neural network accuracy from weights. *arXiv preprint arXiv:2012.02180*, 2020.

[38] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[39] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations (ICLR)*, 2017.

[40] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *International Conference on Learning Representations (ICLR)*, 2019.

[41] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

[42] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be pruned in one-shot. *arXiv preprint arXiv:2301.00774*, 2023.

[43] Sam Ainsworth and Sachin Narkhede. Merging models with weight matching. *arXiv preprint arXiv:2302.02858*, 2023.

[44] Google Research. Gsm8k: A benchmark for grade school math word problems. *arXiv preprint arXiv:2104.11826*, 2021.

[45] OpenAI. Humaneval: Evaluating code generation with natural language prompts. *arXiv preprint arXiv:2107.03374*, 2021.

[46] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering benchmark for commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2019.

[47] AI4Math. Mathvista: A multimodal dataset for mathematical reasoning, 2024.

[48] Dali-Does. Clevr-math: A multimodal dataset for visual and mathematical reasoning, 2024.

[49] Likaixin. Mmcode: A multimodal dataset for code reasoning, 2024.

[50] Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. From recognition to cognition: Visual commonsense reasoning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Department of Information Technology
National Institute of Technology Karnataka, Surathkal-575025

Permission from Guide to Appear for End Semester Evaluation of Major Project-I (IT448)

We student(s) of 7th semester B.Tech.(AI) carried out the Major Project-I titled "Optimizing LLMs: Merging Strategies in Building LLMs for Reasoning" under the guidance of "Dr. Anand Kumar M" from July 2024 to November 2024. I/we have shown the Major Project-I progress regularly to my major project guide and incorporated all suggestions given by my/our Major Project guide in the Major Project-I.

Progress of my/our Major Project-I in points-wise are as follows:

1. We performed inference on WizardLM, WizardMath, and LLama2-Code-Alpaca, exploring merging techniques to enhance performance. Example outputs from model were generated to assess their quality and accuracy.
2. Mistral and Llama2 were selected for further experimentation, with both models fine-tuned on the GSM8K and CommonsenseQA datasets. Additionally, Direct Preference Optimization (DPO) tuning was applied to each fine-tuned model, followed by evaluation on respective test sets.
3. The top-performing models from these experiments were merged using Spherical Linear Interpolation (SLERP) and TIES techniques to improve overall performance and achieve a balanced ensemble.
4. We performed inference on finetuned , dpo tuned and merged models and comparative analysis of these models.
5. To facilitate model accessibility, we built a website by creating APIs for the model, which is deployed on a cloud-based server.
6. All models are named under the "Shastra" convention, following specific naming guidelines, and are available on Hugging Face at https://huggingface.co/shastraai
7. We compared our model results with benchmark results on GSM8K and CommonsenseQA datasets

M Hemanth Kumar (211AI025)
Siddharth Murali S (211AI035)
Aaryan Nijhawan (211AI002)

Department of Information Technology
National Institute of Technology Karnataka, Surathkal-575025

Aforesaid student(s) shown the progress as well as Report of the Major Project-I (IT448) carried out by him/her/them. Progress is satisfactory. (If progress is not satisfactory strickout this line and write the comments below)

Write comments/remarks if the progress of Major Project-I (IT448) is not satisfactory

---

(Signature of the Guide with date)
Dr. Anand Kumar M