TINY-PHYSICS: A COMPACT LARGE LANGUAGE MODEL FOR PHYSICS WORD PROBLEMS ON MOBILE DEVICES

by

Akhil vallala

A Thesis Submitted to the Faculty of

The College of Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

Florida Atlantic University

Boca Raton, FL

May 2025

TINY-PHYSICS: A COMPACT LARGE LANGUAGE MODEL FOR PHYSICS WORD PROBLEMS ON MOBILE DEVICES

by

Akhil vallala

This thesis was prepared under the direction of the candidate's thesis advisor, Dr. Dr. Hari Kalva, Department of Electrical Engineering and Computer Science, and has been approved by the members of his supervisory committee. It was submitted to the faculty of the College of Engineering and Computer Science and was accepted in partial fulfillment of the requirements for the degree of Master of Science.

SUPERVISORY COMMITTEE:

Hari Kalva

Hari Kalva, Ph.D.
Thesis Advisor

Hanqi Zhuang

Hanqi Zhuang, Ph.D.

[illegible]

Xingquan (Hill) Zhu, Ph.D.

Raquel Assis

Raquel Assis, Ph.D.

Hari Kalva

Hari Kalva, Ph.D.
Chair, Department of Electrical Engineering and Computer Science

Stella Batalama

Stella Batalama, Ph.D.
Dean, The College of Engineering and Computer Science

[illegible]

Robert W. Stackman Jr. , Ph.D.
Dean, Graduate College

Apr 25, 2025

Date

VITA

Akhil Vallala is a data science graduate student at Florida Atlantic University, where he is expected to receive his Master of Science in Data Science in May 2025. He holds a strong academic record with a GPA of 3.95/4.0 and a deep interest in natural language processing, machine learning, and educational technology.

Akhil currently serves as a Graduate Research Assistant at Florida Atlantic University, focusing on sentiment analysis and trend modeling in the context of generative AI's influence on education. He previously worked at Accenture and Infosys as a Data Engineer, where he designed and optimized large-scale data pipelines using Azure Data Factory and Databricks, and developed analytics solutions for enterprise clients.

His thesis, titled "Tiny-Physics: A Compact Large Language Model for Enhancing Physics Education on Mobile Devices," explores the development and fine-tuning of small language models tailored for educational use on resource-constrained devices. He designed a custom dataset from the textbook 1000 Solved Problems in Classical Physics and implemented fine-tuning on SmolLM2 variants.

He is the founding President of the PACE Club at FAU, through which he has organized industry speaker sessions, mentorship initiatives, and career guidance events. He was also recognized with the Prestige Graduate Leadership Award for his contributions to student leadership and academic excellence.

His research interests include small language models, instruction fine-tuning, mobile deployment of AI systems, and the intersection of AI with education.

ACKNOWLEDGEMENTS

Deep appreciation is given to Dr. Hari Kalva for his steadfast direction, tenacious encouragement, and invaluable support during the research process. His guidance greatly influenced both the direction and execution of this thesis.

Gratitude is also extended to Dr. Velibor Adzic and Dr. Sven Thijssen for their insightful feedback and ongoing support, which provided consistent inspiration during their academic tenure at Florida Atlantic University.

Sincere thanks are due to family and friends whose unwavering belief and encouragement provided motivation and resilience throughout every challenge encountered.

Appreciation is also given to the FAU PACE Club, whose enthusiastic collaboration significantly contributed to academic and professional development.

ABSTRACT

Author:	Akhil vallala
Title:	Tiny-Physics: A Compact Large Language Model for Physics Word Problems on Mobile Devices
Institution:	Florida Atlantic University
Thesis Advisor:	Dr. Hari Kalva
Degree:	Master of Science
Year:	2025

This thesis offers **Tiny-Physics**, a compact large language model optimized for solving physics word problems on mobile devices. While Small Language Models (SLMs) offer potential for on-device learning, there is a lack of domain-specialized models for physics. This research fine-tunes `smolLM2-360M`, a state-of-the-art SLM, using a combination of publicly available datasets (e.g., `camel-ai/physics`) and a novel synthetic dataset generated through a multi-agent system based on GPT-4o and *1000 Solved Problems in Classical Physics*.

The fine-tuning process incorporates both supervised and instruction-tuning methods, including LoRA for parameter-efficient training. The resulting model is converted to GGUF format for deployment on resource-constrained mobile devices. Evaluation on the MMLU College Physics benchmark shows that instruction-tuning improves performance, with the best model achieving an accuracy of **0.2451**, surpassing the base model.

The proposed study presents a comprehensive, end-to-end framework for synthetic datasets, fine-tuning, evaluation using Lighteval, and deployment on quantized mobile systems. Results underscore the value of lightweight, domain-adapted models in delivering scalable and personalized physics education.

To the graduate students of Florida Atlantic University.

TINY-PHYSICS: A COMPACT LARGE LANGUAGE MODEL FOR PHYSICS WORD PROBLEMS ON MOBILE DEVICES

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 BACKGROUND

Students often require immediate support to understand physics concepts, yet traditional resources such as textbooks or instructors may not always be accessible. This project addresses the need by developing smolLM2, a small and specialized large language model fine-tuned for physics, designed to run efficiently on mobile devices. smolLM2 is available in multiple parameter configurations (135M, 360M, and 1.7B), making it adaptable for mobile deployment scenarios [3].

SmolLM2 learns from curated datasets such as FineWeb-Edu [19], which includes educational content from the web, and Cosmopedia v2 [18], which draws from textbook material. This training enables the model to deliver quick and accurate answers to students' conceptual queries and promotes a more dynamic, self-directed learning experience. This smolLM2 model encourages students to engage with concepts and seek instant clarification. It also serves as a guide apart from traditional learning in the classroom.

Also, such models can assist teachers as an extra learning resource for students outside the classroom, freeing instructors' time to focus on more complex topics and individualized instruction. This strategy leverages technological advancement and available online tools to build more productive and interactive learning spaces, as explored in research on large language models in education. Through its real-time guidance, smolLM2 can make it possible for students and instructors to comprehend and manage physics' complexity more efficiently and meaningfully. Exploring smaller LLMs like smolLM2 is also relevant to the ongoing research in efficient model scaling and performance, as discussed in works exploring smaller language models' behavior and capabilities.

1.2 PROBLEM STATEMENT

Interpreting the physics principles of the real world from textual descriptions and translating them into mathematical formulations is a dual challenge that poses significant difficulties for many students. Educational technology research has continuously examined this obstacle. Contemporary approaches to solving physics word problems rely on rule-based systems and symbolic AI. However, they cannot handle the complicatedness and abnormality of natural language prevalent in such tasks. Failures of such methods often lie in the inability to identify linguistic nuances, varied forms of presentation, and conceptual reasoning requirements beyond pattern identification. Given their ability to comprehend natural language, Small Language Models offer a novel and compelling solution to address this issue.

Unlike traditional symbolic AI or rule-based systems, Language models can comprehend and translate natural language descriptions into appropriate mathematics representations. Such a capability can directly solve students' difficulties comprehending and solving complex physics word problems.

1.3 RESEARCH QUESTION/OBJECTIVES

This thesis explores applying and optimizing Small Language Models (SLMs) to address physics word problems on mobile devices. The investigation focuses on the `smolLM2` model, with parameter sizes of 135M, 360M, and 1.7B. These models are trained on the `camel-ai/physics` dataset [17, 30], as well as synthetic data derived from seed content in *"1000 Solved Problems in Classical Physics"*, along with their respective instruction-tuned variants. The following research questions guide the study:

1. **Fine-tuning for Physics Problem Solving:** What methods effectively fine-tune `smolLM2` using both publicly available datasets (e.g., `camel-ai/physics`) and synthetic datasets to improve accuracy and reasoning in physics word problems?

2. **Comparative Performance:** How does the accuracy and response time of fine-tuned `smolLM2` compare to:

   - its base version (prior to physics-specific fine-tuning),
   - human-level benchmarks in terms of correctness and time efficiency?

3. **Mobile Deployment Considerations:** What key constraints and strategies influence the deployment of `smolLM2` on mobile platforms, including latency, coherence of responses, and memory consumption?

Answering these questions offers practical insights into leveraging SLMs for physics instruction while balancing the benefits of mobile accessibility with the limitations of computational resources. The performance of `smolLM2` is benchmarked against larger LLMs to evaluate its capability in solving and interpreting physics word problems.

1.4 SIGNIFICANCE OF THE STUDY

The study supports the advancement of educational technology by improving personalized physics learning through lightweight AI solutions. Fine-tuned SLMs offer real-time support, enable adaptive learning pathways, and assist instructors in identifying conceptual gaps students encounter. The results provide practical methods for integrating compact language models into a variety of academic domains and problem-solving environments. These insights contribute to designing AI tools specifically adapted to educational contexts that operate under limited computational capacity.

1.5 THESIS STRUCTURE

The thesis comprises five chapters, each progressively contributing to developing, training, evaluating, and deploying a compact language model designed for physics word problem-solving on mobile platforms. The structure maintains a coherent flow from the initial motivation and background to the presentation of experimental results.

Chapter II: Literature Review

This chapter comprehensively surveys existing research on solving physics word problems using rule-based systems and large and small language models (SLMs). It discusses **architectural and data-centric optimization strategies**, provides

benchmarks for evaluating small models and highlights the **need for domain-specific datasets** tailored to physics education.

Chapter III: Methodology

This chapter outlines the **data collection, processing, and model training workflow**. It includes the creation of a **seed dataset from *1000 Solved Problems in Classical Physics***, the development of a **synthetic dataset using a multi-agent GPT-4o pipeline**, and the **instruction tuning process** applied to smolLM2 variants. Additionally, it presents the **LoRA-based training strategy**, **estimates carbon emissions during training**, and describes the **conversion of trained models to GGUF format** for quantized deployment on mobile devices.

Chapter IV: Results

This chapter presents the evaluation results on the **MMLU College Physics benchmark**. It offers **comparative performance analysis** between baseline and instruction-tuned models, provides a **topic-wise accuracy breakdown** and discusses the **implications of fine-tuning strategies**, including cases of model overfitting and loss of generalization.

Chapter V: Conclusion

This chapter summarizes the **contributions** of the study, addresses **limitations such as insufficient MCQ formatting and topic imbalance**, and proposes directions for **future work**, especially enhancing **mobile deployment** and expanding **physics topic coverage** for small language models.

Chapter VI: Future Work

This chapter outlines **future directions** to improve the model's capabilities, such as incorporating **multiple-choice question formatting during fine-tuning**, expanding **topic coverage through targeted synthetic generation**, exploring **multimodal reasoning**, and evaluating **real-world classroom deployment** scenarios. These directions aim to improve further the usability and implementation of lightweight LLMs in physics education.

CHAPTER 2
LITERATURE REVIEW

The study builds upon an extensive foundation in educational technology, natural language processing, and physics education. Prior work has examined how artificial intelligence and huge language models (LLMs) address educational challenges by facilitating personalized learning and improving explanation generation [39]. In physics education, continued efforts aim to identify improved instructional strategies that engage learners and promote conceptual understanding through technological support. This dissertation contributes to this new field by examining the viable application of small-scale language models to physics word problem solving with special emphasis on mobile-friendly deployment. The goal is to create solutions that combine pragmatic usefulness and accessibility, bringing AI-assisted learning to students across a range of learning contexts.

2.1 EXISTING APPROACHES TO PHYSICS WORD PROBLEM SOLVING

Researchers have identified a few existing approaches to solving physics word problems. Researchers initially used rule-based systems to solve physics word problems. In fields like physics, knowledge was often implemented as rules in the form "if ⟨condition is true⟩ then ⟨do action A⟩" [our second turn]. However, these systems often struggled with the flexibility and diversity of real-world physics word problems that go beyond "fixed physical examples or techniques that involve low inference" and rely on "predetermined problem formation and expression patterns" [our second turn, our third turn]. A general drawback of expert systems, including rule-based ones, was their potential for "exponential complexity time, which impairs the desirable real-time response" [our second turn]. The history of automatic solvers for maths word problems experienced stages "from ruled-based system to deep learning models [?]."

Following rule-based systems, statistical learning and deep learning models emerged as approaches to designing automatic solvers for maths word problems. These methods involve training general-purpose computational architectures on a corpus of relevant data to find underlying patterns for the task. However, the paper [?] says "models based on statistical learning and deep learning models failed to work in a large and diversified dataset."

Large language models (LLMs) have brought new approaches to solving maths. These approaches often involve prompting techniques to leverage the reasoning abilities of LLMs. From paper [?], "The significant advantage of these prompt-based methods is that they are free of additional training or gradient updates for the LLM, with researchers focusing on optimizing prompts." Techniques like few-shot and zero-shot prompting over large language models have become promising. Chain-of-thought prompting, which elicits explicit step-by-step reasoning, has also been utilized.

More recent work has focused on conversational frameworks using LLM agents with access to tools. MathChat [45] is an example of a framework that simulates a conversation between an LLM agent and a user proxy agent responsible for tool execution (like Python) and additional guidance to solve challenging maths problems. MathChat integrates various prompting techniques and allows for multi-turn dialogues for iterative refining and debugging.

The above prompting techniques or frameworks can also be applied to physics word problems. However, fine-tuning smolLM2 on physics datasets is crucial because traditional rule-based and statistical approaches often fail to handle the linguistic variability and complex reasoning required for real-world physics word problems. Unlike general-purpose LLMs that rely solely on prompting, a fine-tuned smolLM2 learns domain-specific concepts, vocabulary, and reasoning chains, improving accuracy and contextual understanding.

2.2 CAPABILITIES AND LIMITATIONS OF SMALL LARGE LANGUAGE MODEL

2.2.1 Capabilities of Small Large Language Models (SLMs)

Small Language Models (SLMs) are cost-effective and environmentally sustainable alternatives to larger models [1]. Their reduced parameter count lowers the computational demands for training and inference, which minimizes the carbon footprint [3]. These efficiency gains make SLMs ideal for use on constrained platforms, including embedded and mobile systems, where memory and power constraints remain critical [34]. Despite their compact architecture, these models achieve strong performance in reasoning and problem-solving by leveraging data-centric training, architectural optimization, and fine-tuning strategies.

Key capabilities of SLMs include:

1. **Efficient Computation:** Unlike large language models, which use enormous GPU clusters and carbon footprint for training, Small large language models like MobileLLM and SmolLM2, which have only millions of parameters with less model size, keep computational overhead low.

2. **Targeted Domain Knowledge:** SLMs are trainable with domain-specific datasets to provide them with knowledge in the domain of knowledge, like physics-based concepts, equations, and problem-solving techniques.

3. **Interpretability:** Since they are small, SLMs give greater insight into their workings, allowing for the development of interpreted AI systems in physics education.

4. **Personalization:** SLMs' smaller footprint allows deployment and personalization inside the device, tailoring their behavior to the needs and preferences of individual students.

5. **Domain-Specific Adaptability:** Adapting on domain-specific datasets enhances their performance on specialized tasks, such as physics problem-solving

6. **On-Device Deployment:** Optimized for mobile and edge devices, these models enable low-latency inference without requiring cloud-based computation.

7. **Reduced Memory Footprint:** Techniques such as layer-sharing (MobileLLM) and multi-stage dataset curation (SmolLM2) allow these models to retain high reasoning accuracy while minimizing memory usage.

2.2.2 Limitations of Small Large Language Models

Though efficient, SLMs encounter several challenges that must be addressed:

1. **Limited Knowledge Capacity:** Smaller parameter sizes translate to less general knowledge storage, with more stringent dataset selection necessary to maintain knowledge.

2. **Weaker Complex Reasoning Ability:** SLMs lag behind larger LLMs in multistep problem-solving and deductive reasoning, necessitating instruction tuning and data augmentation.

3. **Weaker Complex Reasoning Abilities:** Compared to larger LLMs, SLMs can struggle with multi-step problem-solving and logical deduction, necessitating instruction tuning and dataset augmentation.

4. **Susceptibility to Overfitting:** Because SLMs rely on smaller datasets, they are more prone to overfitting on domain-specific knowledge, requiring careful data mixing and augmentation strategies[41].

5. **Performance Trade-Offs:** While optimization techniques such as grouped-query attention (GQA) and embedding sharing improve efficiency, they may introduce slight trade-offs in performance on certain benchmarks[2].

2.3 SMOLLM2 AND MOBILELLM: OPTIMIZED SMALL MODELS

2.3.1 MobileLLM: Architecture-Centric Efficiency

MobileLLM is a family of lightweight LLMs designed for mobile deployment. The creation of these models responds to the growing demand for on-device LLMs because of cloud cost and latency concerns. Contrary to the notion that data and parameter numbers are the primary determinants of model quality, this work highlights the importance of model architecture for LLMs with less than a billion parameters [34].

Key characteristics and achievements of MobileLLM include:

1. It utilizes an exceptional architecture to create a strong baseline network. MobileLLM achieves a remarkable accuracy boost over previous SOTA models with 125 million and 350 million parameters. It attains a 2.7% and 4.3% accuracy increase on zero-shot common sense tasks.

2. The paper also proposes an immediate block-wise weight-sharing approach (MobileLLM-LS), further enhancing accuracy with marginal latency overhead and no increase in model size.

3. The MobileLLM family significantly improves chat benchmarks compared to other sub-billion parameter models.

4. The design philosophy of MobileLLM scales effectively to larger models up to 1.5 billion parameters, consistently outperforming previous models of similar scale.

5. In essence, MobileLLM represents an effort to engineer high-quality LLMs within the constraints of mobile devices by focusing on architectural innovations and weight utilization techniques.

MobileLLM Architecture:

Researchers structured the MobileLLM Architecture to improve the efficiency of LLMs with fewer than a billion parameters for on-device applications. They applied deep and thin architectures, embedding sharing, grouped query attention, and layer sharing to the design. Evaluating the performance of these smaller LLMs requires a close examination of the model's architecture [34].

Architectural characteristics:

1. Deep and Thin Architectures: MobileLLM models use deep and narrow network architectures. Such architectures are better than wide and shallow ones for small models learning abstract concepts and generalizing well to many tasks. In experiments, deeper networks performed better than wider ones when the model is small. The optimal depth was around 30 layers.

2. Embedding Sharing: To maximize weight utilization in sub-billion parameter models, which have a significant portion of parameters in the embedding layers, MobileLLM revisits input-output embedding sharing. The model achieves a more compact architecture by sharing the input embedding weights with the output fully connected layer.

3. Grouped-Query Attention (GQA): MobileLLM utilizes grouped query atten-

tion. Originally designed to reduce key-value cache size in larger LLMs, GQA has also effectively reduced redundancy in key-value heads in smaller models. It can be seen as a form of weight-sharing where the number of key-value heads is a fraction of the query heads, and these kv-heads are repeated during attention computation. Experiments indicated that using 16 query heads and reducing kv-heads resulted in comparable or slightly reduced accuracy with a notable decrease in model size, highlighting GQA as a method to optimize small models further. The optimal head dimension was found to be close to 64.

4. SwiGLU in Feed-Forward Network (FFN): MobileLLM incorporates the SwiGLU activation function within its feed-forward networks. Research showed that using SwiGLU instead of the traditional ReLU-based FFN improved average performance on zero-shot reasoning tasks for the 125M model, indicating its benefit for smaller models[40].

5. Layer Sharing (in MobileLLM-LS): Besides the baseline MobileLLM architecture, the researchers proposed immediate block-wise weight sharing in a variant denoted as MobileLLM-LS. This technique involves replicating transformer blocks and sharing their weights. The immediate block-wise repeat strategy was chosen to utilize best the hardware memory hierarchy, where shared weights can stay in the SRAM cache and be computed twice immediately, avoiding the need for transfer between SRAM and DRAM and improving the execution speed. Ablation studies showed that repeating transformer blocks with weight sharing leads to accuracy enhancements without increasing model storage costs.

The development process began with a baseline sub-billion parameter model and progressively integrated architectural techniques to establish a robust MobileLLM foundation. Researchers experimentally evaluated the impact of each design decision, demonstrating the individual contributions of deep and thin architectures, embedding sharing, GQA, and SwiGLU to model effectiveness. The application of layer-sharing techniques advanced the development of MobileLLM-LS, resulting in improved performance. Empirical results validated these architectural enhancements, as evidenced by MobileLLM's strong performance across zero-shot commonsense reasoning, question answering, and reading comprehension benchmarks, surpassing prior sub-billion parameter models. These design principles also scaled effectively to larger models with up to 1.5 billion parameters.

MobileLLM training configuration:

The training configurations for MobileLLM involved several key aspects.

1. Hardware and Batch Size: The researchers executed the experiments on 32 A100 GPUs, assigning a batch size 32 to each GPU.

2. Training Data and Iterations: The researchers conducted initial exploratory experiments using 120k iterations over 0.25T tokens. The highest-performing models underwent 480k iterations using a dataset of 1T tokens. These findings confirm that the top-performing MobileLLM and MobileLLM-LS models achieved their results after being trained on significantly larger datasets and for extended durations.

3. Optimiser and Learning Rate: Training began from scratch by applying the Adam optimizer with a weight decay of 0.1 [28]. The initial learning rate was configured to 2e^{-3} and followed a cosine learning-rate decay schedule.

4. Training Objectives: The training process optimized the models using the next token prediction task with hard labels.

5. Knowledge Distillation (Exploration): The paper explored knowledge distillation (KD), employing LLaMA-v2 7B as the teacher model. The KD loss calculation used cross-entropy between the logits of the teacher and student models. However, results showed that KD significantly prolonged training time (2.6–3.2x slowdown) and delivered accuracy comparable to or lower than models trained solely with hard labels [15]. As a result, the final MobileLLM models adopted training exclusively with the next token prediction objective.

In summary, the MobileLLM models were trained on a substantial amount of data (1T tokens for the best models) using a distributed setup with A100 GPUs and a specific optimiser configuration. While knowledge distillation was explored, the final models relied on standard next token prediction training.

MobileLLM Performance:

As reported in [**?**], MobileLLM demonstrates strong performance across LLM benchmark datasets despite having fewer than one billion parameters.

Analysis of MobileLLM's performance:

1. Zero-shot Common Sense Reasoning:

   (a) MobileLLM achieves a remarkable accuracy boost over previous state-of-the-art models with 125 million and 350 million parameters. It attains a 2.7% and 4.3% accuracy increase on zero-shot common sense tasks. These tasks include ARC-easy, ARC-challenge, BoolQ, PIQA, SIQA, HellaSwag, OBQA, and WinoGrande.

   (b) The layer-sharing variant, MobileLLM-LS, further enhances accuracy by 0.7% and 0.8% over MobileLLM 125M and 350M, respectively, on these tasks.

   (c) Compared to other models, MobileLLM-125M outperforms previous models like OPT-125M, GPT-Neo-125M, and Galactica-125M by a significant margin. It also achieves higher accuracy than Pythia-160M and RWKV-169M while being smaller. MobileLLM-LS-125M shows comparable or even higher results than most previous 350M models.

   (d) In the 350M model size category, MobileLLM-350M surpasses previous state-of-the-art models by more than four percentage points.

   (e) When scaled to larger models (600M, 1B, and 1.5B parameters), MobileLLM consistently outperforms previous models of similar scale. MobileLLM-1.5B achieves an average accuracy of 59.4%, outperforming Qwen1.5-1.8B despite having fewer parameters.

2. Question Answering and Reading Comprehension: On the TQA benchmark, MobileLLM-125M improves over 4.3 F1 score points compared to its predecessors. The MobileLLM-350M model showcases a remarkable performance boost of about 10 F1 score points over other 350M-sized models on TQA.

The MobileLLM family achieved considerable accuracy on the RACE reading comprehension exam.

3. Downstream Tasks:

   (a) Chat: Fine-tuned MobileLLM models significantly outperform previous state-of-the-art sub-billion scale models on chat benchmarks like AlpacaEval and MT-Bench, surpassing models with 1 billion parameters. MobileLLM-LS-350M achieves a remarkable win rate of 48.2% compared to the baseline GPT-3 model (text-davinci-001), which has a self-win rate of 50%.

   (b) API Calling: MobileLLM-350M demonstrates comparable intent and structure exact match scores to the much larger LLaMA-v2 7B model on a synthetic API calling dataset. Smaller models demonstrated strong performance by matching structured exact match scores of the significantly larger LLaMA-v2 7B model on a synthetic API-calling dataset. These results illustrate the effectiveness of lightweight models for typical on-device tasks.

4. Compatibility with Quantization: Employing W8A8 post-training quantization (PTQ) on MobileLLM and MobileLLM-LS models results in a modest accuracy reduction of less than 0.5 percentage points and remains compatible with layer sharing.

5. On-device Latency: On an iPhone 13, MobileLLM-LS-125M incurs only a 2.2% increase in loading and initialization time and a mere 2.6% overhead in execution time compared to MobileLLM-125M despite having double the number

of layers due to weight sharing. Improved data locality explains this efficiency. In contrast, increasing the number of layers without employing weight sharing leads to noticeably higher loading and execution times.

In conclusion, MobileLLM establishes a new SOTA for less than a billion parameter language models, demonstrating significant improvements across various tasks and promising efficient on-device deployment.

2.3.2 SmolLM2: Data-Centric Model Optimization

smolLM2, a compact language model developed by Hugging Face with 1.7 billion parameters, achieves strong performance while maintaining low computational demands. Built on the Llama2 architecture, it leverages a data-centric training approach, pre-trained on 11 trillion tokens across various datasets like Stack-Edu, FineWeb-Edu, and FineMath, followed by fine-tuning on SmolTalk for enhanced instruction-following. The design team optimized the model for reasoning tasks; smolLM2 demonstrates strong performance in mathematics and code generation, outperforming similar-sized models such as LLaMA3.2–1B and Qwen2.5–1.5B on benchmarks including ARC and GSM8K. Its lightweight design and extended 8k-token context length make it an ideal candidate for mobile deployment. It supports applications like physics education by providing efficient, accurate problem-solving capabilities on resource-constrained devices.

SmolLM2 Architecture

SmolLM2 is a state-of-the-art model designed to balance computational efficiency and performance, making it highly suitable for deployment on mobile and edge devices. This model contains approximately 1.7 billion parameters; SmolLM2 is built upon the LLaMA 2 architecture, a transformer-based framework recognized for its scalability and effectiveness in natural language understanding tasks.

Architecturally, SmolLM2 comprises 24 transformer layers, a hidden size of 2048, 32 attention heads, and a feed-forward network (FFN) dimension of 8192. These components enable parallel attention, which is critical for domain-specific tasks like solving physics word problems, deep multi-step reasoning, and effective contextual representation. The 4x FFN multiplier further enhances learning capacity and representational power.

In terms of optimization, SmolLM2 employs tied embeddings, which reduce memory usage and improve inference latency—an advantage for mobile deployment. Rotary Position Embeddings (RoPE) with a base of =10,000 are used to encode positional information efficiently. Although initially trained for a 2048-token context length, the model was extended post-training to support up to 8k tokens, enabling it to handle long educational inputs effectively [42].

The model uses SwiGLU as its activation function—a combination of Swish and Gated Linear Units—chosen for its improved non-linearity and training stability over standard functions like ReLU. This choice contributes to more robust performance in reasoning-intensive tasks such as physics education [40].

smolLM2 Training Configuration

Beyond its efficient architecture, SmolLM2 benefits from a well-structured multi-stage training regimen designed for scalability and educational utility, as shown in Table 2.1. The model was trained over 11 trillion tokens in multiple phases, leveraging a global batch size of 2 million tokens to ensure efficient GPU utilization and stable convergence. The entire pre-training process was conducted using the Nanotron framework across 256 H100 GPUs, allowing for high-throughput training while maintaining fine-grained control over model scaling and resource allocation.

Table 2.1: SmolLM2's Multi-Stage Training Process

Stage	**Token (T)**	**Key Datasets**	**Computational Resources**	**Findings**
Stage 1	0 to 6	FineWeb-Edu (60%, 1.3T), DCLM (40%, 3.8T), StarCoder-Data (10%, 250B)	256 H100 GPUs, stable LR	Poor coding and math performance
Stage 2	6–8	English web (75%), StarCoder-Data (20%), Open-WebMath (5%, 12B)	256 H100 GPUs, stable LR	Improved code performance; minimal math impact

Continued on next page

Table 2.1 – continued from previous page

Stage	**Token (T)**	**Key Datasets**	**Computational Resources**	**Findings**
Stage 3	8–10	English web (75%, adjusted), Stack-Edu (24%), Math (10%, InfiMM-WebMath + OWM)	256 H100 GPUs, stable LR	Improved benchmarks; loss spike
Stage 4	10–11	English web (58%), Stack-Edu (24%), Math (14%, FineMath4+), Cosmopedia v2 (4%, 30B)	256 H100 GPUs, decay LR	Significant coding and math improvements

A key component of SmolLM2's performance is its tokenizer, which has a vocabulary size of 49,152 tokens. This tokenizer was trained on a diverse variety of datasets at different stages, including FineWeb-Edu [19], Cosmopedia v2 [18], OpenWebMath [37], StarCoderData [31], and StackOverflow, providing extensive language coverage across formal academic text, code, and mathematical reasoning. This diversity is significant for adapting the model to educational domains, ensuring it can handle various terminologies and syntax patterns in physics word problems and other learning content.

Base Model vs Instruction-Tuned Model

Language models can be broadly categorized based on training objectives and intended usage. This section highlights the key differences between a base language model and an instruction-tuned model, using SmolLM2 and its instruction variant (SmolLM2-Instruct), as shown in table 2.2 [3].

Base Language Model

A base language model, such as the base SmolLM2, is trained via unsupervised pretraining on a massive corpus of unstructured text (e.g., web pages, code, math problems). The pretraining strategy equips the model to identify language regularities, retain factual information, and adapt to diverse linguistic structures.

1. The training objective is to predict the next token in a sequence, making it capable of language modeling and continuation tasks.

2. The model is not explicitly taught to follow natural language instructions or respond helpfully to user queries.

3. SmolLM2's base version was trained on 11 trillion tokens in a multi-stage training process involving diverse sources such as English web data, FineWeb-Edu, DCLM, math datasets, and coding datasets.

4. Despite its general capabilities, a base model may not reliably complete instruction-following tasks without further tuning.

Instruction-Tuned Model

An instruction-tuned model, such as SmolLM2-Instruct, is derived from a base model and fine-tuned using datasets of instruction-response pairs. This tuning phase explicitly teaches the model to understand and follow human instructions in natural language.

1. The instruction tuning process involves Supervised fine tuning (SFT) on curated datasets, such as SmolTalk, explicitly developed for SmolLM2.

2. After SFT, models are often further optimized using Direct Preference Optimization (DPO) or similar techniques [38]. This aligns the model's behavior with human preferences, enabling it to provide helpful, safe, and contextually relevant responses.

3. SmolLM2-Instruct was created by fine-tuning the base SmolLM2 using SFT on SmolTalk and then applying DPO with UltraFeedback.

In summary, the base model provides the foundational linguistic capabilities. In contrast, the instruction-tuned model adds task-following intelligence, making it more suitable for deployment in educational and assistant-style applications.

Difference between 1.7B and, 360M and 135M

SmolLM2 comes in three variants—1.7B, 360M, and 135M—each designed to serve different deployment needs while maintaining efficient language understanding. Though all share the same transformer-based architecture, their training strategy, data usage, and capabilities differ significantly, as shown in table 2.3 [3].

Table 2.2: Comparison Between Base Model and Instruction-Tuned Model

Feature	**Base Model (SmolLM2)**	**Instruction Model (SmolLM2-Instruct)**
Training Objective	Next-token prediction	Instruction-following
Input Format	Prompt continuation	Instruction/response format
Supervision	Unsupervised (text only)	Supervised with instruction datasets
Alignment	Not aligned to preferences	Aligned via DPO or RLHF
Use Cases	General language modeling	Task-oriented interaction
Example Dataset	FineWeb-Edu, DCLM, OpenWebMath, etc.	SmolTalk (instruction tuning)

Table 2.3: Detailed Comparison of SmolLM2 Model Variants

Aspect	**SmolLM2-135M**	**SmolLM2-360M**	**SmolLM2-1.7B**
Parameters	135 million	360 million	1.7 billion
Training Tokens	2 trillion	4 trillion	11 trillion
Training Strategy	Single-stage training	Single-stage training	Multi-stage training
Data Handling	DCLM filtered using FineWeb-Edu classifier, Stack-Edu, InfiMM-WebMath, FineMath, Cosmopedia	DCLM filtered using FineWeb-Edu classifier, Stack-Edu, InfiMM-WebMath, FineMath, Cosmopedia	Progressive dataset scheduling across stages
Architecture	Same as 1.7B, uses Grouped Query Attention (GQA)	Same as 1.7B, uses GQA	LLaMA-style transformer with standard attention

Continued on next page

Table 2.3 – continued from previous page

Aspect	SmolLM2-135M	SmolLM2-360M	SmolLM2-1.7B
Scheduler	WSD scheduler with 20% decay, learning rate 3e-3	Same	Multi-stage LR decay
Post-Training Tuning	SFT using filtered SmolTalk, DPO with UltraFeedback	Same	Not explicitly mentioned

smolLM2 Performance

SmolLM2-1.7B: Base Model Performance The base SmolLM2 model with 1.7B parameters was pre-trained on a huge 11 trillion tokens using various datasets, including DCLM, FineWeb-Edu, and The Stack, along with newly curated mathematics and coding datasets. The model utilizes a Transformer decoder architecture and was trained with bfloat16 precision. The evaluation results for the base SmolLM2-1.7B, reported as zero-shot unless otherwise stated, show significant improvements over its predecessor, SmolLM1-1.7B, and competitive performance against other models like Llama-1B and Qwen2.5-1.5B is shown in Table 2.4 [23].

These results highlight significant knowledge, reasoning, and mathematics advances compared to the first-generation SmolLM1-1.7B.

Table 2.4: Benchmark Evaluation: SmolLM2-1.7B vs Llama-1B, Qwen2.5-1.5B, and SmolLM1-1.7B

Metric	**SmolLM2-1.7B**	**Llama-1B**	**Qwen2.5-1.5B**	**SmolLM1-1.7B**
HellaSwag	**68.7**	61.2	66.4	62.9
ARC (Average)	**60.5**	49.2	58.5	59.9
PIQA	**77.6**	74.8	76.1	76.0
MMLU-Pro (MCF)	**19.4**	11.7	13.7	10.8
CommonsenseQA	**43.6**	41.2	34.1	38.0
TriviaQA	**36.7**	28.1	20.9	22.5
Winogrande	**59.4**	57.8	59.3	54.7
OpenBookQA	42.2	38.4	40.0	**42.4**
GSM8K (5-shot)	31.0	7.2	**61.3**	5.5

SmolLM2-1.7B-Instruct: Instruction-Following Performance The instruction-tuned version of the 1.7B parameter model was trained with supervised fine-tuning (SFT) utilizing a combination of online and the researchers' curated datasets Smalltalk. Subsequently, Direct Preference Optimization (DPO) using UltraFeedback was applied to align the model with user preferences. The evaluation results for SmolLM2-1.7B-Instruct, reported as zero-shot unless stated otherwise, demonstrate strong instruction-following capabilities compared to Llama-1B-Instruct, Qwen2.5-1.5B-Instruct, and SmolLM1-1.7B-Instruct is shown in Table 2.5 [24].

The instruct model supports a variety of tasks, which includes **summarization**, **text rewriting**, and **function calling**. It achieves a score of **27% on the BFCL Leaderboard** for function calling, demonstrating its practical ability in structured output generation [24].

SmolLM2-360M: Base Model Performance

SmolLM2-360M is a 360-million parameter model trained on a broad and carefully filtered corpus of 4 trillion tokens. Built on a Transformer decoder-only architecture with bfloat16 precision, it is designed to deliver high performance within the small model class. Evaluation across a range of zero-shot benchmarks demonstrates that SmolLM2-360M achieves SOTA results for models of its size, outperforming notable competitors like Qwen2.5-0.5B and the original SmolLM-360M baseline [20].

Specifically, the model assesses common sense reasoning, language understanding, and factual knowledge, as shown in Table 2.6.

SmolLM2-360M-Instruct: Instruction-Following Performance

SmolLM2-360M-Instruct is an instruction-tuned variant of the base model, fine-

Table 2.5: Benchmark Evaluation: SmolLM2-1.7B-Instruct vs Llama-1B-Instruct, Qwen2.5-1.5B-Instruct, and SmolLM1-1.7B-Instruct

Metric	**SmolLM2-1.7B-Instruct**	**Llama-1B-Instruct**	**Qwen2.5-1.5B-Instruct**	**SmolLM1-1.7B-Instruct**
IFEval (Average prompt/inst)	**56.7**	53.5	47.4	23.1
MT-Bench	6.13	5.48	**6.52**	4.33
OpenRewrite-Eval (micro_avg RougeL)	44.9	39.2	**46.9**	NaN
HellaSwag	**66.1**	56.1	60.9	55.5
ARC (Average)	**51.7**	41.6	46.2	43.7
PIQA	**74.4**	72.3	73.2	71.6
MMLU-Pro (MCF)	19.3	12.7	**24.2**	11.7
BBH (3-shot)	32.2	27.6	**35.3**	25.7
GSM8K (5-shot)	**48.2**	26.8	42.8	4.62

Table 2.6: Base Model Benchmark: SmolLM2-360M vs Qwen2.5-0.5B and SmolLM-360M

Metrics	**SmolLM2-360M**	**Qwen2.5-0.5B**	**SmolLM-360M**
HellaSwag	**54.5**	51.2	51.8
ARC (Average)	**53.0**	45.4	50.1
PIQA	**71.7**	69.9	71.6
MMLU (cloze)	**35.8**	33.7	34.4
CommonsenseQA	**38.0**	31.6	35.3
TriviaQA	**16.9**	4.3	9.1
Winogrande	52.5	**54.1**	52.8
OpenBookQA	**37.4**	**37.4**	37.2
GSM8K (5-shot)	3.2	**33.4**	1.6

tuned via Supervised Fine-Tuning (SFT) using a combination of curated public and proprietary datasets. Further enhanced with Direct Preference Optimization (DPO) using UltraFeedback, this model is tailored to follow natural language instructions across various use cases, including summarization, text rewriting, and question answering[25].

Evaluations against instruction-tuned peers such as Qwen2.5-0.5B-Instruct and SmolLM-360M-Instruct reveal its superior performance in most instruction-based benchmarks as shown in Table 2.7:

Table 2.7: Instruction Model Benchmark: SmolLM2-360M-Instruct vs Qwen2.5-0.5B-Instruct and SmolLM-360M-Instruct

Metric	**SmolLM2-360M-Instruct**	**Qwen2.5-0.5B-Instruct**	**SmolLM-360M-Instruct**
IFEval (Average prompt/inst)	**41.0**	31.6	19.8
MT-Bench	3.66	**4.16**	3.37
HellaSwag	**52.1**	48.0	47.9
ARC (Average)	**43.7**	37.3	38.8
PIQA	**70.8**	67.2	69.4
MMLU (cloze)	**32.8**	31.7	30.6
BBH (3-shot)	27.3	**30.7**	24.4
GSM8K (5-shot)	7.43	**26.8**	1.36

Overall, SmolLM2-360M-Instruct shows clear improvements over its base model, particularly its ability to interpret and respond to structured natural language prompts. The gains in instruction-following capabilities demonstrate the effectiveness of instruction tuning and preference alignment even in smaller-scale models [25].

SmolLM2-135M:Base Pre-trained Model

The base SmolLM2 model with 135 million parameters was pre-trained on 2 trillion tokens using a diverse dataset including FineWeb-Edu, DCLM, The Stack, and newly curated filtered datasets1. The model architecture is a Transformer decoder, and it was trained with bfloat16. The evaluation results for the base SmolLM2-135M are reported as zero-shot unless stated otherwise. Here is a comparison of its performance against its predecessor, SmolLM-135M, as shown in Table 2.8 [21]:

Table 2.8: Benchmark Evaluation: SmolLM2-135M-8k vs SmolLM-135M

Metrics	**SmolLM2-135M-8k**	**SmolLM-135M**
HellaSwag	**42.1**	41.2
ARC (Average)	**43.9**	42.4
PIQA	68.4	68.4
MMLU (cloze)	**31.5**	30.2
CommonsenseQA	**33.9**	32.7
TriviaQA	4.1	**4.3**
Winogrande	51.3	51.3
OpenBookQA	**34.6**	34.0
GSM8K (5-shot)	**1.4**	1.0

These results indicate that the base SmolLM2-135M [21] generally shows significant advances over its previous model, SmolLM1, in areas like knowledge and reasoning, as evidenced by the improved scores on benchmarks such as HellaSwag [47], ARC [5], MMLU [14], and CommonsenseQA.

SmolLM2-135M-Instruct: Instruction-Tuned Model

The instruction-tuned version of the 135M parameter model was created through supervised fine-tuning (SFT) using a combination of public datasets and the researchers' curated smol-smoltalk [26]. Following SFT, Direct Preference Optimization (DPO) [38] using UltraFeedback was applied. The optimization procedure enhances the model's capability to understand and execute instructions, as demonstrated in Table 2.9 [22].

Table 2.9: Benchmark Evaluation: SmolLM2-135M-Instruct vs SmolLM-135M-Instruct

Metric	**SmolLM2-135M-Instruct**	**SmolLM-135M-Instruct**
IFEval (Average prompt/inst)	**29.9**	17.2
MT-Bench	**19.8**	16.8
HellaSwag	**40.9**	38.9
ARC (Average)	**37.3**	33.9
PIQA	**66.3**	64.0
MMLU (cloze)	**29.3**	28.3
BBH (3-shot)	**28.2**	25.2
GSM8K (5-shot)	1.4	1.4

These results demonstrate that SmolLM2-135M-Instruct also shows improvements in instruction following capabilities compared to SmolLM-135M-Instruct across various benchmarks. The instruct model supports tasks such as summarization and text rewriting [22].

2.4 MODEL SELECTION:

SmolLM2-360M was selected over MobileLLM for fine-tuning with physics data due to its superior capabilities demonstrated in prior evaluations.

1. Stronger Pre-training and General Capabilities: SmolLM2-360M has been pre-trained on a substantial 4 trillion tokens using a diverse dataset. This extensive pre-training might provide a more affluent general knowledge and language understanding foundation, which could benefit learning the specific concepts and terminology in physics data. SmolLM2 demonstrates meaningful advances over its previous model, SmolLM1, especially in instruction following reasoning knowledge; after fine-tuning, these enhanced general capabilities could translate to better learning and performance in a specialized domain like physics.

2. Competitive Zero-Shot Performance: The base pre-trained SmolLM2-360M performs well on several zero-shot common sense reasoning tasks compared to other models in its size category. This suggests a strong underlying ability to understand and reason, crucial for comprehending complex physics data. For instance, it outperforms SmolLM-360M and Qwen2.5-0.5B on metrics like HellaSwag [47], ARC (Average), PIQA [4], MMLU (cloze), CommonsenseQA, and TriviaQA.

3. Availability of an Instruct Version: SmolLM2 also has an instruct version, SmolLM2-360M-Instruct, which has been developed through Direct Preference Optimization (DPO) and supervised fine-tuning (SFT). This instruction-tuned model supports text rewriting, summarization, and function-calling tasks. While not directly physics-related, its fine-tuning for instruction could make it more adaptable to learning specific physics tasks or answering physics-related questions after further fine.

Regarding whether the MobileLLM architecture is implemented in the SmolLM2-360M model:

The sources indicate that the SmolLM2 family of models, including the 360M parameter version, shares the same architecture as the 1.7B model, though it uses GQA. The 1.7B version of SmolLM2 follows the LLaMA2 architecture.

MobileLLM also leverages grouped-query attention (GQA) as a key architectural mechanism to improve weight utilization. Therefore, SmolLM2-360M implements at least one key architectural component central to the MobileLLM design: Grouped Query Attention (GQA).

However, the MobileLLM architecture is also characterized by its emphasis on deep and thin architectures and embedding sharing. While SmolLM2-360M uses GQA, the sources do not explicitly state whether it adopts the ”deep and thin” structure or embedding sharing strategies specifically described for MobileLLM. SmolLM2-360M is based on the Transformer decoder, and the larger 1.7B version follows the LLaMA2 architecture, suggesting its overall structure might differ from the specific ”deep and thin” design philosophy highlighted in the MobileLLM paper.

2.5 EXISTING DATASETS TO TRAIN OR FINE-TUNE PHYSICS LANGUAGE MODELS:

One can employ numerous good quality datasets for that purpose, but without a doubt, the best out there is camel-ai/physics [17]. The dataset contains 20,000 physics word problems and solutions in many topics, including mechanics, electromagnetism, thermodynamics, etc. The extensive set of problems, often involving multi-step thinking and applying physics principles, makes the camel-ai/physics dataset an excellent choice for training and evaluating language models for solving physics problems. Each topic on this dataset contains 800 questions, as shown in Table 2.10.

Table 2.10: camel-ai/physics Dataset

Topic	**Question Count**
Quantum mechanics	800
Thermodynamics	800
Electromagnetism	800
General relativity	800
Special relativity	800
Atomic physics	800
Nuclear physics	800
Particle physics	800
Optics	800

Continued on next page

Topic	**Question Count**
Fluid mechanics	800
Solid state physics	800
Plasma physics	800
Astrophysics	800
Cosmology	800
Gravitational waves	800
Superconductivity	800
Condensed matter physics	800
High-energy physics	800
Quantum field theory	800
String theory	800
Black holes	800
Dark matter	800
Quantum computing	800
Chaos theory	800
Biophysics	800

On the contrary, some other physics-focused datasets, such as PhysicsQA and ARC Challenge, are comparatively more contained or concept questioning-based rather than quantitative problem-solving. Camel-ai/physics dataset richness and breadth qualify it as a go-to dataset for researchers and instructors who desire to create and use language models for education and help with physics [7].

With the application of this excellent dataset, scientists can train or fine-tune their tiny language models to perform well on a different range of physics problems and provide practical help and guidance to students in their studying and solving processes.

2.6 NEED FOR CREATION OF OWN DATASET:

The evaluation in this thesis uses the MMLU College Physics dataset to validate model performance. Topic prediction for MMLU was conducted using ChatGPT, followed by cross-verification with the topic coverage of the `camel-ai/physics` dataset, as described in the methodology section. The analysis revealed that `camel-ai/physics` does not encompass all topics in the MMLU dataset. Consequently, a new dataset was developed to address the uncovered topics and ensure comprehensive topic coverage.

2.7 SEED DATASET:

A custom dataset was constructed using Python and the OpenAI API by extracting problems from the textbook *1000 Solved Problems in Classical Physics* by Ahmad A. Kamal [11]. The resulting dataset contains 953 physics problems paired with corresponding solutions, excluding formulas and similar questions. This dataset draws inspiration from the methodology described in paper [7], aggregating physics problems from various online sources.

The dataset includes the following columns: `id`, `question`, `solution`, `number of sub-questions`, `type`, and `most similar question`. During model inference, each primary question is accompanied by a similar question provided as a few-shot prompt, allowing the model to infer the problem-solving strategy better. Since the dataset was created specifically for fine-tuning language models, the content emphasizes the question-and-answer pairs without embedding explicit formulas. Including formulas could serve instruction fine-tuning purposes; however, such information would prompt the model to focus on solving equations rather than reasoning through the underlying physical concepts.

Combining the custom dataset with the `camel-ai/physics` dataset expands the volume and diversity of training data, thereby building a more comprehensive foundation for fine-tuning and assessing small language models in the context of physics education and problem-solving.

2.8 SYNTHETIC DATASET:

2.8.1 Advantages:

Creating synthetic datasets offers a practical method for improving the effectiveness of LLMs, exceptionally when high-quality, real-world training data is expensive or challenging to collect [32]. A custom synthetic dataset of physics questions and answers was generated using seed data and the OpenAI API to address these limitations.

1. Augmenting Training Data in Low-Resource Settings: LLMs can generate synthetic data to augment existing training datasets, mainly when real-world data is scarce.

2. Supporting Zero-Shot Learning: Some studies explore the feasibility of generating a synthetic dataset entirely from scratch using LLMs to support zero-shot learning, where models are expected to perform tasks without any direct training examples for those specific tasks.

3. Improving Model Generalisation: By generating diverse synthetic examples, models can be exposed to more data patterns than in a limited real-world dataset.

4. Cost-Effectiveness and Scalability: Developing open-source models to generate large-scale synthetic data offers a promising alternative to scaling up training data with proprietary models [35].

5. Verification and Reliability of Solutions: For domain-specific datasets like physics word problems, generating synthetic questions and their solutions is not good enough. The generated question and answer should also go through the process of validation.

2.8.2 Potential limitations:

1. Task Subjectivity: The effectiveness of LLM-generated synthetic data for text classification can be negatively associated with the subjectivity of the classification task. For tasks with low subjectivity (e.g., news topic classification and spam email detection), models trained on synthetic data can perform comparably to those trained on real-world data. However, the decrease in performance can be significant for highly subjective tasks (e.g., humor or sarcasm detection).

2. Data Diversity: The real-world data may often be more diverse than synthetic data generated by LLMs, especially for highly subjective tasks. Increasing the diversity of synthetic data is identified as a potential way to improve its effectiveness, particularly for subjective tasks.

3. Domain-Specific Knowledge: For tasks requiring specialized knowledge, such as financial sentiment analysis or understanding political sarcasm, LLMs might not be fully equipped to create adequate synthetic data without that specific domain knowledge.

Despite the limitation, generating synthetic datasets, especially with the advancement in LLMs and with techniques like verification, holds a significant potential in improving model performance.

2.9 TYPES OF FINE TUNING:

Several fine-tuning methods help adapt a language model to a specific domain, such as solving physics word problems. These methods enhance the model's capacity to understand and accurately solve physics questions:

1. Supervised Finetuning: In this method, training happens with the help of labeled datasets specific to the particular task or domain, such as physics word problem and their corresponding solutions. This method is beneficial with task-specific datasets and for accurately solving physics word problems [44].

2. Instruction Tuning: In this type of SFT, the model will be trained with a pair of natural language instructions and their corresponding responses. The

goal is to teach the model to follow instructions accurately, making it better understand the physics queries in physics education [43].

3. Domain-Specific Finetuning: In this method, training happens with a domain-specific dataset on a pre-trained language model. The aim is to ensure the model understands the terminology and concept of the domain [6].

4. Parameter-Efficient Finetuning: This category of methods focuses on updating only a tiny subset of the language model's parameters during finetuning to be particularly efficient for smaller language models. However, its specific application to physics education is less discussed in the provided sources [13].

5. Reinforcement Learning from Human Feedback (RLHF): This technique aims to correct the language models with human preferences. Many use it as a post-training step after completing supervised finetuning. The SmolLM2-Instruct model applies Preference Optimization (DPO), a form of Reinforcement Learning from Human Feedback (RLHF). While potentially helpful in aligning the model with educational needs, it is considered less directly relevant for the initial adaptation to physics content [29].

In this thesis, the focus is on applying parameter-efficient fine-tuning techniques, such as Low-Rank Adaptation (LoRA) [16] and Instruction Fine-Tuning, to optimize small language models for solving physics word problems with the synthetic dataset and the camel-ai physics dataset. These approaches balance model performance and computational efficiency, enabling real-time deployment on mobile devices.

2.10 LIGHTEVAL

Lighteval is a versatile and efficient toolkit developed by the Hugging Face Leaderboard and Evals Team. It is an all-in-one solution for evaluating Large Language Models (LLMs) across multiple backends, including transformers, tgi, vllm, and others. Its key features include speed, allowing the evaluation of models hosted on the Hugging Face Hub via the accelerated backend; seamless storage of results on platforms like the Hugging Face Hub or locally; and a simple Python API for easy integration. Lighteval offers customization, enabling users to utilize existing evaluation tasks and metrics or to create their own tailored to specific needs. smolLM2 used the Lighteval library as a framework to evaluate the model on benchmark datasets [10].

2.11 GENERIC GPT UNIFIED FILE FORMAT

GGUF (Generic GPT Unified File Format) is a file format specially designed for efficient storage and deployment of large language models, especially the quantized models that are reduced from standard floating-point formats (like 32-bit or 16-bit) to lower-bit formats like 8-bit, 5-bit, 4-bit, or even 2-bit. It is the advanced version of its predecessor, GGML (Generic GPT Model Language), accommodating larger and more complex models while improving efficiency [27].

Before GGUF and GGML, the earlier formats like native Tensorflow (.pb) and PyTorch (.pt or .pth) files were used. These formats store models unquantized with complete precision (32-bit floating point) weights, leading to larger file sizes and high memory usage. ONNX (Open Neural Network Exchange) emerged as a crossframework standard for interoperability but mostly handled full-precision weights with limited quantization capabilities. The shift towards the half-precision floating point (FP16) models helped reduce carbon footprint but was still insufficient for a billion parameter scaled models on consumer hardware. These limitations lead to the development of more compact and optimized formats to run GPT-like models on resource-constrained devices.

GGML introduces the initial step in quantization-based model optimization by storing models using reduced precision formats such as 8-bit integers (INT8) and 4-bit integers (INT4). This approach minimizes model size and facilitates efficient execution on consumer-level hardware, including CPUs. However, as models are growing and require more sophisticated metadata management, GGUF is developed to extend and improve GGML.

Features of GGUF:

GGUF is an advanced version of GGML, with new features to efficiently handle the modern Large Language Models.

1. Quantization Support: It supports various quantization formats like 4-bit and 8-bit to reduce the model size and memory footprint while maintaining adequate precision for inference.

2. Enhanced Metadata Storage: GGUF can store more extensive metadata about the model, such as architecture, tokenization schemes, hyperparameters, and quantization details.

3. Compatibility with Large Models: GGUF is designed to work with models of

size 100GB+, optimizing memory usage to run them on consumer hardware.

4. Efficient Inference: GGUF optimizes the inference process, allowing faster execution even on low-power devices by intelligently using quantized weights.

5. Single-file deployment: GGUF models can be loaded into a single file containing all the necessary information[12].

6. Extensibility: GGUF is an extensible new feature that can be added without breaking the compatibility.

   The development of GGUF has significantly impacted the LLM's model deployment, addressing previous challenges related to model size, compatibility, and efficiency.

The development of GGUF has significantly impacted the LLM's model deployment, addressing previous challenges related to model size, compatibility, and efficiency.

CHAPTER 3
METHODOLOGY

3.1 BECHMARK DATASET

The Massive Multitask Language Understanding (MMLU) benchmark is a comprehensive suite designed to evaluate the performance of language models across 57 diverse tasks, covering a lot of subjects which includes STEM, humanities, and social sciences. It is widely used to assess a language model's reasoning capabilities, factual knowledge, and generalization ability in zero-shot or few-shot learning scenarios [14].

Within this broader benchmark lies the College Physics subset, which is curated to assess a language model's capacity to understand and solve undergraduate-level physics problems. These questions reflect students' problems during their first or second year of a college-level physics course. This subset presents the questions in a multiple-choice format with five options (A–E), where only one option represents the correct answer. These questions actively test the LLM's ability to understand physics concepts, perform mathematical reasoning to solve quantitative problems, and apply fundamental physical laws such as Newton's Laws, kinematics, thermodynamics, and electric circuit principles. Furthermore, some questions require dimensional and unit analysis to evaluate a solution's correctness and apply physics concepts to real-world scenarios in areas like engineering and astronomy. MMLU: College physics can be a strong benchmark for evaluating the model's performance when trained in physics. The training dataset was designed based on the topics presented in the MMLU College Physics benchmark.

3.2 NEED FOR A SYNTHETIC DATASET:

A good training dataset is significant in this thesis; camel-ai/physics [30] is used, which contains 800 questions per topic and is reliable for training models on physics word problems. However, it covers all the topics in the validation dataset MMLU college physics [14]. OpenAI predicted the topic of each question from MMLU college physics and identified 20 topics in this dataset. The predicted topics are Quantum mechanics, Optics, Thermodynamics, Atomic physics, Electrostatics, Special relativity, Solid state physics, Rotational Dynamics, Nuclear physics, Electromagnetism II, Particle physics, Oscillations, Gravitation, Waves, Electric Circuits, Particle Dynamics, Superconductivity, Kinematics and statics, Cosmology, and Heat and Matter. However, not all the topics are covered in camel-ai/physics. The common topics between MMLU and camel-ai/physics are Atomic physics, Cosmology, Nuclear physics, Optics, Particle physics, Quantum mechanics, solid-state physics, Special relativity, Superconductivity, and thermodynamics. The missing topics are Electric-Circuits, Electromagnetism, Gravitation, HeatandMatter, Kinematics and statics, Oscillations, Particle Dynamics, Rotational Dynamics, and Waves. The dataset found online lacks specific topics, questions, and answers. Table 3.1 shows the common and missing topics between MMLU College Physics and camel-ai/physics. This study creates a synthetic dataset to address the missing topics.

3.3 CREATION OF SEED DATASET:

This study creates the seed dataset using the textbook 1000 Solved Problems in Classical Physics, which contains diverse physics problems and their corresponding solutions. As shown in Table 3.2, this dataset contains 15 topics (Kinematics and statics, Rotational Kinematics, Gravitation, Electromagnetism I, Rotational Dynamics, Electrostatics, FluidDynamics, Oscillations, Waves, Particle Dynamics, Lagrangian and Hamiltonian Mechanics, HeatandMatter, ElectricCircuits, Electromagnetism II, and Optics) and 953 records with columns, Question_number, Questions, Sub-category, Category, Solutions, Readable_Questions and Readable_Solutions.

STEPS TAKEN TO EXTRACT THE DATA

1. **Data Extraction Preparation:** The PDF version of the textbook *"1000 Solved Problems in Classical Physics"* presents each chapter's concepts, problems, and solutions. This study organizes each chapter as a PDF, including only the problems and their solutions while excluding the conceptual content.
2. **Data Extraction Process:** To extract physics questions and answers containing mathematical symbols, `PyPDFLoader` was identified as the most suitable option compared to `pdfminer.six` and `PyMuPDF`. `PyPDFLoader` provides a more structured text extraction, essentially retaining the formatting, symbols, and layout of the content, which is highly critical when dealing with mathematical content that must be exact.

Table 3.1: Topic Comparison: MMLU vs Camel-AI Physics

Only in MMLU	**Common Topic**	**Missing Topic**
Atomic physics	Atomic physics	
Special relativity	Special relativity	
Optics	Optics	
Electromagnetism II		Electromagnetism II
Waves		Waves
Quantum mechanics	Quantum mechanics	
Thermodynamics	Thermodynamics	
Solid state physics	Solid state physics	
Particle Dynamics		Particle Dynamics
Gravitation		Gravitation
Rotational Dynamics		Rotational Dynamics
Nuclear physics	Nuclear physics	
ElectricCircuits	ElectricCircuits	
Particle physics	Particle physics	
Oscillations		Oscillations
Kinematics and statics		Kinematics and statics
Superconductivity	Superconductivity	
Cosmology	Cosmology	
Heat and Matter		Heat and Matter

Table 3.2: Number of Questions per Topic

Topic	**Question Count**
Electrostatics	97
Electromagnetism II	91
Electromagnetism I	85
Waves	83
Rotational Dynamics	72
Optics	72
Heat and Matter	70
Oscillations	65
Particle Dynamics	61
Kinematics and Statics	60
Electric Circuits	56
Gravitation	54
Rotational Kinematics	42
Fluid Dynamics	28
Lagrangian and Hamiltonian Mechanics	15

`pdfminer.six` is slower and fails to retain complicated formatting, i.e., equations. `PyMuPDF` is stable and fast but may require more effort in cleaning and organizing the extracted data, especially complicated mathematical symbols. Considering the specific need for exact extraction of physics material and mathematical symbols, `PyPDFLoader` became the best choice. The extracted data was cleaned by removing headings, and the cleaned data was passed through OpenAI's language model to convert it to LaTex format. The questions and solutions were matched based on the chapter number and problem number.

Architecture for data extraction form PDF: The pipeline reads the PDF and returns data in LaTeX format. Figure 3.1 illustrates the flowchart of this pipeline.

(a) Read the PDF and store the data.

(b) Remove unnecessary content, such as headlines, page numbers, etc., from the data.

(c) Use regular expressions to extract and store all the question numbers in a list.

(d) Loop through each question pair, extracting the text between the current question and the next.

(e) Send the extracted data to OpenAI and convert it into LaTeX format.

(f) Store the question number and LaTeX-formatted data in a CSV file.

(g) Using Python, convert the LaTeX formatted data to a readable format and store it as a new column in the CSV.

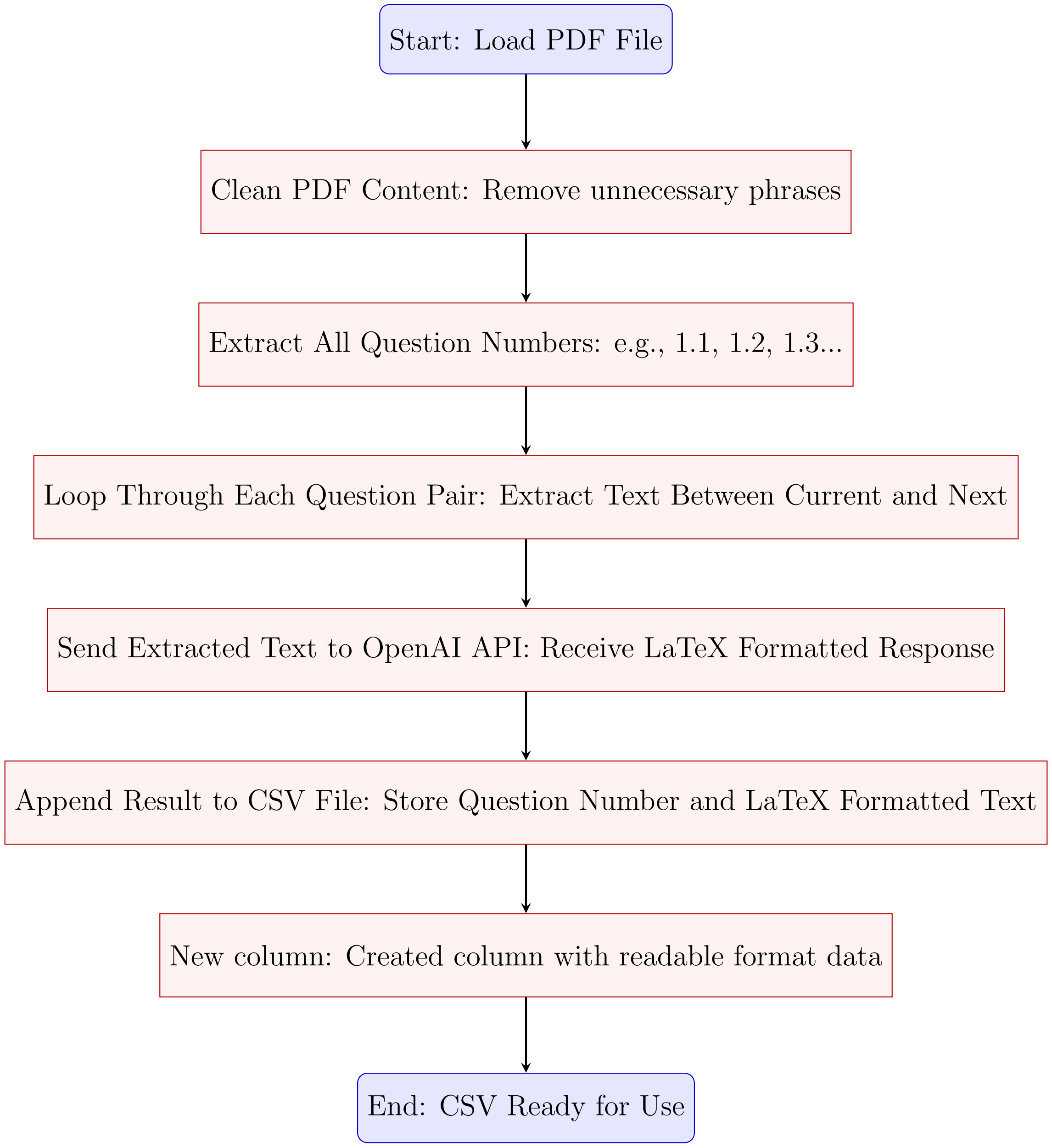

Start: Load PDF File
Clean PDF Content: Remove unnecessary phrases
Extract All Question Numbers: e.g., 1.1, 1.2, 1.3...
Loop Through Each Question Pair: Extract Text Between Current and Next
Send Extracted Text to OpenAI API: Receive LaTeX Formatted Response
Append Result to CSV File: Store Question Number and LaTeX Formatted Text
New column: Created column with readable format data
End: CSV Ready for Use

Figure 3.1: Pipeline Flowchart for PDF to LaTeX Dataset Conversion

3. **Other Methods Used for Data Extraction:** Numerous libraries were explored and tested to extract data from the PDF, including evaluating their performance on the equations presented:

   - **pdf2txt (pdfminer.six):** A `pdfminer.high_level` library was utilized to extract data from the PDF. However, as evidenced in the provided figure, the library encountered limitations in accurately recognizing the symbols and order of the equations, resulting in imperfect data extraction.
   - **PyMuPDF (fitz):** Another library, `PyMuPDF`, was tested, which proved more effective in extracting the equations in a structured format. This library is relatively better than `pdf2txt (pdfminer.six)`. However, it still lacks the power to recognize the power and division symbols.
   - **PyMuPDF & SymPy:** Now, we have used a combination of `PyMuPDF` and `SymPy`. However, the results are still the same.
   - **EasyOCR:** A text extraction approach was implemented to obtain content from the PDF images. Specifically, the PDF was first converted to image format using the `pdf2image` library. Then, the `EasyOCR` optical character recognition tool was employed to extract the text data from the images. However, this approach yielded extracted text lacking the structural features needed since it could not effectively extract the symbolic elements in the source PDF content.
   - **Visual Model smolVLM LLaVA:** This study evaluates the `smolVLM` [36] and `LLaVA` [33] visual models for their ability to extract informa-

tion from PDF images generated from the source text. While the models promise to extract some content, they exhibit weaknesses in capturing essential structural elements, particularly mathematical symbols and equations from the original material.

This method extracts 953 questions from the PDF into LaTeX format. The table 3.2 shows the distribution of questions across each topic.

3.4 SYNTHETIC DATASET GENERATION

The data generation pipeline generates more diverse and substantial training data for synthesis. It leverages a multi-agent system built on a robust large language model (LLM), specifically GPT-4o [9], accessible via the Trussed AI API. The system generates new, high-quality physics questions and answers that resemble real academic content.

3.4.1 Motivation

The seed data contains only **953 records**, which is less than enough to fine-tune a model. We can boost the number of records in the data by employing synthetic data generation. The synthetic data generation pipeline mimics the seed dataset's style, format, and mathematically rigorous nature, specifically the physics problems and solutions.

3.4.2 Architecture: Multi-Agent System

The synthetic pipeline is composed of three agents as shown in Figure 3.2, each powered by GPT-4o:

1. **Question Generator Agent**: The Question Generator Agent retrieves a question from the seed data along with its category and sub-category, then feeds this information into the prompt of ChatGPT-4o. Using the prompt, the model generates three similar questions for each difficulty mode—easy, medium, and hard—based on the seed question as a reference.

   Prompt:

   Question Generator Agent Prompt

   difficulty_prompt = "easy": "Generate a simple high school physics question that is easy to solve.", "medium": "Generate a standard high school physics question with moderate difficulty.", "hard": "Generate a challenging high school physics question that requires deep understanding."
   conversation_history = ["role": "system", "content": "You are a physics expert generating high school physics questions.", "role": "user", "content": f" difficulty_prompt[mode] The question should belong to the Category: Category and sub-category: sub_category. Here is an example question: original_question Generate a new question based on this example."]

2. **Answer Generator Agent**: The answer generator agent takes questions,

which are generated synthetically by the question generating agent, mode of difficulty, Category, and sub-category, and generates the answer to the synthetic question using a chain of thought prompting.

Answer Generator Agent Prompt

conversation_history = [“role”: “system,”

“content”: “You are a physics tutor providing detailed step-by-step solutions for high school physics questions.”, “role”: “user,” “content”: f“ Provide a detailed and correct answer to the following question: question

- The question belongs to the Category: category and Sub-category: sub_category.

- The difficulty mode is: mode.

- Think step by step before arriving at the final answer.

- Clearly state any physics concepts or formulas used.

- Show the calculations if applicable.

- Ensure the answer is structured and easy to understand. ”]

3. **Validation Agent**: The validation agent takes a synthetic question and its answer, which the answering agent generates, mode of difficulty, Category, and sub-category, validates the question, and is checked. If the answer is correct to the question, the validation agent will return correct; if the answer is incorrect, it generates it as incorrect. If there is no clear answer, the agent will generate it as unclear. Moreover, if the validation result is not correct, which is unclear or Incorrect, the model re-validates again once.

Validation Agent Prompt

conversation_history = [“role”: “system,”

“content”: “You are an AI tutor that validates physics answers.”, “role”: “user,”

“content”: f“ The question belongs to Category: Category and Sub-category: sub_category. The difficulty mode is mode.

Does the following answer wholly and correctly address the question? Answer only with ’Correct,’ ’Incorrect,’ or ’Unclear.’

Question: question

Answer: answer ”]

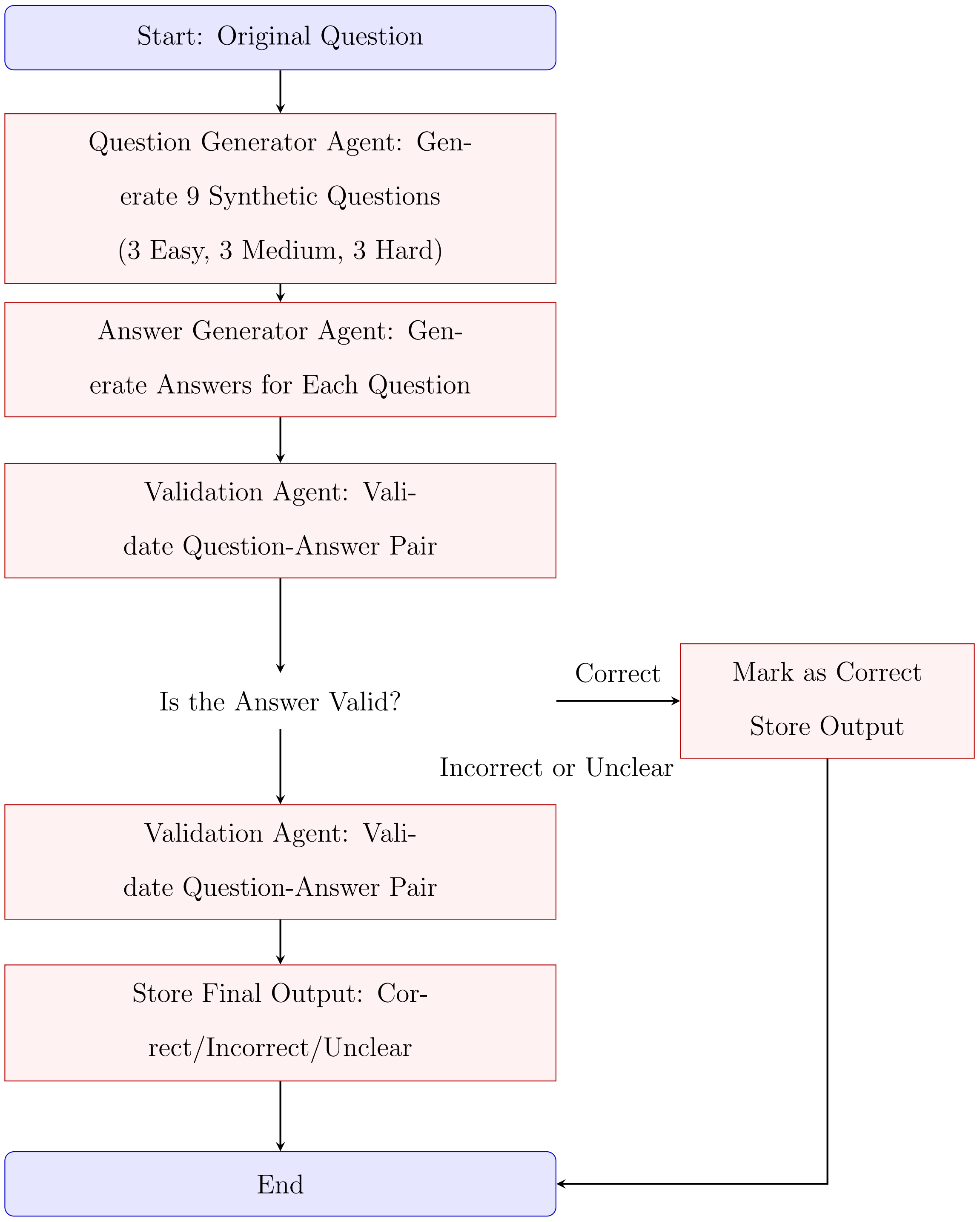

Start: Original Question
Question Generator Agent: Generate 9 Synthetic Questions (3 Easy, 3 Medium, 3 Hard)
Answer Generator Agent: Generate Answers for Each Question
Validation Agent: Validate Question-Answer Pair
Is the Answer Valid?
Correct
Mark as Correct Store Output
Incorrect or Unclear
Validation Agent: Validate Question-Answer Pair
Store Final Output: Correct/Incorrect/Unclear
End

Figure 3.2: Synthetic Dataset Generation Pipeline with Multi-Agent Validation

3.4.3 Data Collection Pipeline

The dataset generation follows a batch-oriented approach that enables controlled API utilization and effective management of rate limits. This process uses the OpenAI API provided by Trusted AI. The system generates three artificial questions at varying complexity levels for each unique original question: easy, medium, and hard. As a result, the pipeline produces nine questions per original question—three at each difficulty level: easy, medium, and hard. After generating each group, the system incrementally stores the data in CSV format to ensure data integrity and prevent loss.

3.4.4 API Access and Infrastructure

All agents utilize the Trussed AI platform, a proxy for OpenAI GPT-4o and other LLM models like Gemini [8], Ollama, etc. The system uses retry logic, exponential backoff strategies, and batch-based scheduling to effectively manage transient network issues and rate limitations.

3.4.5 Quality Assurance

After creating the dataset, the process cleans the data by removing duplicates and filtering entries based on the 'Validation Status' marked as correct. The AI system generates solutions for several open questions, as illustrated in the figure 3.2. The cleaning step eliminates solutions mistakenly included in the question column. The final cleaned dataset contains 6,279 records. Table 3.3

presents the number of questions per topic. Each entry in the dataset contains attributes such as the Original Question, Category, Sub-category, Difficulty Mode, Synthetic Question, Synthetic Answer, and Validation Status.

Table 3.3: Topic-Wise Distribution of Questions from the Seed Dataset

Topic	**Question Count**
Waves	651
Electromagnetism I	588
Electrostatics	549
Optics	518
Electromagnetism II	498
Heat and Matter	482
Rotational Dynamics	464
Oscillations	445
Kinematics and Statics	408
Electric Circuits	374
Gravitation	351
Particle Dynamics	344
Rotational Kinematics	334
Fluid Dynamics	203
Lagrangian and Hamiltonian Mechanics	70

3.5 CREATION OF INSTRUCTED DATASETS:

The `camel-ai/physics` dataset includes columns such as `topic`, `sub_topic`, `message_1`, and `message_2`, where `message_1` presents the physics question and `message_2` provides its corresponding answer. The process reformats each record into a structured `prompt–response` pair and stores it in a new JSON file, making it suitable for instruction fine-tuning in large language models.

- **Prompt:** This field presents the physics problem to the model in an instructional tone. It incorporates the dataset's topic, sub-topic, and central question.
- **Response:** This field contains the answer or solution to the problem, phrased as continuing the instruction.

The following Python string formatting is used to generate the `prompt` and `response` fields from the dataset:

Instruction for camel-ai/physics dataset

prompt = f"Solve the following physics problem related to row['topic'], (row['sub_topic']): row['message_1']"

response = f"The solution is as follows: row['message_2']"

This method maintains the contextual integrity of each problem and effectively delivers it to the model, enhancing the accuracy of generated responses. The system stores each example as a JSON Lines (`.jsonl`) entry to facilitate efficient and scalable training.

The formatting process for the synthetic dataset follows a similar structure. It replaces the original columns with new labels: `Category` in place of `topic`, `Subcategory` instead of `subtopic`, `Questions` in place of `message_1`, and `Solutions` instead of `message_2`. The structure applied to the synthetic dataset is illustrated below.

Instruction for synthetic dataset

prompt = f"Solve the following physics problem related to row['Category'], (row['Subcategory']): row['Questions']"

response = f"The solution is as follows: row['Solutions']"

3.6 MODEL TRAINING:

3.6.1 Training Datasets:

The training process utilizes four primary datasets: (1) `camel-ai/physics`, (2) `Instruction tuned camel-ai/physics`, (3) the synthetic dataset, and (4) `Instruction tuned synthetic dataset`, each prepared through custom preprocessing. A merged dataset, referred to as `combined_dataset`, integrates `camel-ai/physics` with the synthetic dataset. Although both sources contain similar content, their column labels differ. The process unified the structure by applying the following mappings: `Topic` aligns with `Category`, `Sub_Topic` with `Subcategory`, `Message_1` with `Questions`, and `Message_2` with `Solutions`. The final merged dataset adopts the standardized column names: `Topic`, `Sub-Topic`, `Questions`, and `Solutions`. In addition, another dataset, named `Instruction` `tuned synthetic dataset`, combines both instruction-tuned datasets. This preparation results in six total datasets available for training.

3.6.2 Models used:

In this thesis, the training involves smolLM2-360M [20] as the base model and smolLM-360M-Instruct [25] as the instruction model on the previously mentioned physics dataset. This thesis trains a base model on both uninstructed datasets—combined camel-ai/physics [30] and synthetic datasets—as well as on instructed datasets, including the Instruction tuned combined dataset, Instruction tuned camel-ai/physics, and Instruction tuned synthetic dataset.

3.6.3 Experiment

In this thesis, nine fine-tuning experiments are done. Among those six train the base model `smolLM-360M` on uninstructed and instruction-tuned versions of the `combined`, `camel-ai`, and `synthetic` datasets. The other three are with the instruction-tuned model `smolLM-360M-Instruct`, which undergoes training on instruction-tuned versions of the same datasets. Each experiment follows identical training configurations, as outlined in Table 3.4, with an 80:20 train-test split.

The difference between general and instruction training lies in the models used, the datasets, and the integration of the Trainer class and Tokenizer, as shown in Table 3.5. Table 3.6 presents the training output for each fine-tuning.

After training, the evaluation uses **accuracy** as the metric to assess model performance on the MMLU: `college_physics` [14] benchmark using lighteval library. The results section discusses the **topic-wise analysis** in detail.

Additional experiments apply **LoRA** (Low-Rank Adaptation) [16] for fine-tuning, resulting in lower accuracy than full fine-tuning. The lower accuracy occurs because LoRA updates only a subset of the model's parameters, limiting its capacity to adapt fully. The smolLM2-360M model, when fine-tuned using LoRA, achieved an accuracy of 0.1471 on the camel_ai_physics_dataset, 0.1569 on the combined_dataset, and 0.1471 on the synthetic_dataset. The configuration used for LoRA is r=16, lora_alpha=32, lora_dropout=0.1, task_type = "CAUSAL_LM." Increasing the number of training epochs to 10 reduces accuracy due to overfitting. A smaller number of epochs proves sufficient for small models, as increasing the epochs tends to cause overfitting.

After completing training, the system converts the best-performing model into the **GGUF** (Generic GPT Unified Format) [12] and deploys it to **Hugging Face** for streamlined access and integration. The GGUF model can be accessed and used directly in applications like **LM Studio** or **PocketPal** for efficient local inference and deployment.

Table 3.4: Training Configuration for Fine-Tuning

Parameter	**Value**
Batch Size per Device	4

Parameter	**Value**
Gradient Accumulation Steps	4
Learning Rate	2×10^{-5}
Number of Epochs	3
Warmup Steps	100
Max Sequence Length	512
Evaluation Strategy	End of each epoch
Save Strategy	End of each epoch
Early Stopping	Enabled (patience = 2)
Best Model Selection Metric	Validation Loss
Optimizer	AdamW 8-bit
Weight Decay	0.01
LR Scheduler Type	Linear
Logging Steps	100
Save Steps	1000
Save Total Limit	2 checkpoints
Random Seed	3407
Output Directory	`outputs`
Experiment Tracker	Weights & Biases (`report_to="wandb"`)

Table 3.5: Comparison Between Full Training and Instruction Tuning

Aspect	**Supervised Fine Tuning**	**Instruction Tuning**
Model	Base Model (e.g., `SmolLM2-360M`)	Base Model (e.g., `SmolLM2-360M`) or Instruction Model (e.g., `SmolLM2-360M-Instruct`)
Dataset	Uninstructed Dataset (e.g., `camel-ai/physics`, `synthetic`)	Instruction-Tuned Dataset (e.g., `Instruction_tuned_camel-ai`, `Instruction tuned synthetic`)
Trainer Class	`Trainer` from Hugging Face	`SFTTrainer` from TRL (`trl`)
Tokenizer Integration	Standard tokenizer usage	Tokenizer with `setup_chat_format` and `apply_chat_template`

3.7 CARBON FOOTPRINT FOR TRAINING THIS MODELS

Based on Table 3.6 the training outputs provided, the estimated carbon footprint for training the smolLM2 series models using an NVIDIA RTX A5500 24GB GPU is approximately 2.64 kg COe. This estimation considers the Energy consumed by the GPU and the system during the training runtime for each model-dataset combination and converts it to carbon dioxide equivalent emissions using a global average carbon intensity for electricity.

Table 3.6: Training Outputs for Model-Dataset Combinations

Model	**Dataset**	**Steps**	**Loss**	**Runtime (s)**	**Samples/sec**	**Steps/sec**	**FLOPs**	**Epochs**
smolLM2-360M	combined_dataset	3290	0.1519	8825.97	28.563	0.446	2.03e+17	10.00
smolLM2-360M	combined_Instruction_dataset	4926	0.1419	13171.40	23.936	0.374	3.04e+17	11.00
smolLM2-360M-Instruct	combined_Instruction_dataset	4926	0.1412	13153.49	23.968	0.375	3.04e+17	11.00
smolLM2-360M	camel_ai_physics_dataset	2500	0.1607	6737.51	28.497	0.445	1.55e+17	10.00
smolLM2-360M	Instructed_camel_ai_physics_dataset	3750	0.1511	10056.00	23.866	0.373	2.32e+17	11.00
smolLM2-360M-Instruct	Instructed_camel_ai_physics_dataset	3750	0.1496	10070.13	23.833	0.372	2.32e+17	11.00
smolLM2-360M	synthetic_dataset	939	0.1309	2446.46	24.565	0.384	5.77e+16	11.89
smolLM2-360M	synthetic_Instruction_dataset	1176	0.1255	3030.95	24.832	0.388	7.21e+16	11.00
smolLM2-360M-Instruct	Instrusted_synthetic_dataset	1176	0.1232	3030.35	24.837	0.388	7.21e+16	11.00

The calculation involved several steps:

3.7.1 Estimating the Power Consumption

Estimating GPU power draw requires assuming an average utilization of 80% during training. With this assumption, the NVIDIA RTX A5500, which has a rated power consumption of 230W (0.23 kW), operates at approximately 184W (0.184 kW) during training. Adding 100W (0.1 kW) for other system components results in a total estimated power draw of 0.284 kW.

3.7.2 Calculating Energy Consumption

The estimation of energy consumed for each training run uses the formula:

$$Energy(kWh) = Power(kW) \times Runtime(hours) \tag{3.1}$$

The process converts each runtime value from seconds to hours. For instance, training `smolLM2-360M` on the `combined_dataset` involves a runtime of 8825.97 seconds, which equals 2.4517 hours, resulting in an estimated energy consumption of 0.6963 kWh.

3.7.3 Determining the Carbon Footprint

The estimation process calculates the carbon footprint using the formula:

$$CarbonFootprint(kgCO_2e) = Energy(kWh) \times CarbonIntensity(kgCO_2e/kWh) \tag{3.2}$$

The calculation uses a global average carbon intensity of 0.475 kg CO_2e/kWh. For example, training `smolLM2-360M` on the `combined_dataset` consumes 0.6963 kWh of energy, resulting in a carbon footprint of 0.3307 kg CO_2e.

3.7.4 Summing the Individual Footprints

The analysis added the carbon footprints of all nine model-dataset training runs to calculate the estimated carbon footprint of 2.64 kg CO_2e.

3.7.5 Considerations and Assumptions

- The GPU power draw might vary based on the specific workload optimizations.
- The carbon intensity of the electricity grid significantly impacts the carbon footprint. This estimate of 2.64 kg CO_2e uses a global average as its basis.
- The assumed system overhead of 100W may vary depending on the hardware used.
- This calculation only accounts for the operational emissions during the training runtime and does not include the embodied carbon footprint of manufacturing the GPU or other infrastructure.

For context, the estimated total carbon footprint of 2.64 kg CO_2e is roughly equivalent to driving a petrol car for approximately 6–7 miles.

3.8 GGUF FORMAT AND QUANTIZATION STRATEGY

The **GGUF (GPT-Generated Unified Format)** defines a model format that enables the conversion of the `akhilfau/fine-tuned-smolLM2-360M-with-on_combined_Instr` model. The `llama.cpp` framework handles the conversion process. This format facilitates **efficient inference on resource-constrained devices**, particularly mobile platforms.

3.8.1 Purpose of Quantization in Mobile Deployment

The system applies specific quantization techniques to fine-tuned models, such as `akhilfau/fine-tuned-smolLM2-360M-with-on_combined_Instruction _dataset`, to minimize model size and accelerate inference. These optimizations are essential for deploying models on platforms like mobile devices. The source details several quantization types applied to the SmolLM2-360M model in GGUF format, each with its trade-offs.

3.8.2 Quantization Types and Trade-Offs

- **FP16 (Floating-Point 16-bit)**: This format represents a full-precision GGUF type that uses 16-bit floating-point weights. Developers designed it to serve as a **reference model with approximately 100% of the original model's quality** (no quantization loss). While it offers the highest accuracy, it increases the model size (700–800MB) and slows down inference speed (2–4 tokens/second on high-end mobile devices), making it less practical for most mobile applications due to its high memory demands (1.5–2GB RAM). Its primary use case emphasizes **maximum accuracy** or usage in non-mobile setups.
- **Q8_0 (8-bit Integer)**: This quantization method converts model weights to **8-bit integers with a zero-point offset** to preserve high accuracy. It achieves approximately 99% of the original quality with reduced size (500–600MB) and moderate inference speed (3–6 tokens/second on high-end

devices). The format reduces RAM usage (1–1.5GB). **Q8_0 supports mobile devices with sufficient resources** (e.g., iPhone 12+, Android with 6GB+ RAM) and favors **accuracy over maximum speed**.

- **Q4_K_M (4-bit Integer with K-means Clustering, Medium Variant)**: This method uses **4-bit integers along with K-means clustering** to optimize the weight distribution, aiming for a **balance between accuracy and efficiency**. It results in the smallest model size (300–400MB), the fastest inference speed (5–10 tokens/second on mid-range devices), and the lowest RAM usage (600–800MB) among the GGUF quantizations. While it experiences a slight accuracy loss (95–97% of original quality), developers consider it **best for real-time mobile applications** (e.g., chatbots) on mid-range devices (4GB+ RAM) due to its low resource demands and high speed, potentially leveraging mobile NPUs.

3.8.3 Quantization Selection Criteria

The selection of quantization depends on the **specific use case and the target device's capabilities**. FP16 or Q8_0 might offer the best option for tasks requiring the highest possible accuracy, provided the device includes sufficient resources. Q4_K_M delivers a good compromise between speed and accuracy for real-time mobile applications on less powerful devices with significantly reduced resource usage.

3.8.4 Preservation of Original Model Files

The report highlights that developers retain the original model files—typically in FP32 or BF16 format—to maintain high numerical precision when using libraries such as `transformers` in non-mobile environments. However, these formats are too large and inefficient for mobile deployment.

3.8.5 Summary of Quantization Strategy

In summary, the strategic decision to apply these specific GGUF quantizations to the fine-tuned SmolLM2-360M model enabled the creation of versions that support **efficient deployment on mobile devices**. The quantization process balances trade-offs between model size, inference speed, accuracy, and RAM usage. Developers **prioritize Q4_K_M** for speed and lower resource usage on mid-range mobiles, **choose Q8_0** to enhance accuracy on high-end devices, and **rely on FP16** as a high-accuracy reference.

CHAPTER 4
RESULTS:

4.1 PERFORMANCE OF EXISTING MODELS ON MMLU COLLEGE_PHYSICS:

Based on the provided Tables 4.1 and 4.2, here is a discussion of the benchmark results and the impact of fine-tuning on language models, including SmolLM2:

Benchmark Results on MMLU: College Physics:

Table 4.1 presents the evaluation results of several language models on the college_physics subset of the MMLU benchmark, using accuracy as the primary metric.

- The model achieving the highest accuracy is Qwen2.5-1.5B-Instruct [46], with a score of 0.4608. This indicates that, among the evaluated models, this instruction-tuned 1.5 billion parameter model performs best on college-level physics questions.
- In general, larger models exhibit better performance, as seen by the higher accuracy of Qwen2.5-1.5B and SmolLM2-1.7B compared to their smaller counterparts. This aligns with the expectation that increased model capacity allows for a better understanding and processing of complex infor-

mation.

- GPT-2, a historically significant model, shows a lower accuracy of 0.2157, highlighting recent advancements in smaller language models (SLMs).

Table 4.1: Model Evaluation Results on MMLU: College Physics

Model	**Accuracy**	**Accuracy Std. Error**
HuggingFaceTB/SmolLM2-360M	0.1863	0.0387
HuggingFaceTB/SmolLM2-360M-Instruct	0.2451	0.0428
Qwen2.5-0.5B	0.3039	0.0458
Qwen2.5-0.5B-Instruct	0.3137	0.0462
gpt2	0.2157	0.0409
HuggingFaceTB/SmolLM2-1.7B	0.3039	0.0458
HuggingFaceTB/SmolLM2-1.7B-Instruct	0.3235	0.0466
Qwen2.5-1.5B	0.3922	0.0486
Qwen2.5-1.5B-Instruct	0.4608	0.0496

Performance Evaluation of SmolLM2 Models:

This area details the performance evaluation of the SmolLM2 series of language models, focusing on the result of instruction fine-tuning on the smolLM2-360M. The analysis draws upon benchmark results and a comparative study of the base smolLM2-360M and its instruction-tuned variant, smolLM2-360M-Instruct, across various datasets.

From Table 4.1 a significant observation across the evaluation is the consistent benefit of instruction fine-tuning. As the benchmark results highlight, instruction-tuned models outperform their base counterparts across various model sizes. For instance, SmolLM2-360M-Instruct achieved an accuracy of 0.2451 on the MMLU: College Physics benchmark, notably higher than the 0.1863 achieved by the base SmolLM2-360M on the same benchmark.

Table 4.2: Accuracy of smolLM2-360M and smolLM2-360M-Instruct on Different Datasets

Dataset	**Model**	
	smolLM2-360M	**smolLM2-360M-Instruct**
Instructed_camel_ai_physics_dataset	0.2255	**0.2549**
camel_ai_physics_dataset	0.2059	–
combined_Instruction_dataset	**0.2451**	0.2451
combined_dataset	0.2353	–
synthetic_Instruction_dataset	0.2353	0.2353
synthetic_dataset	0.2255	–

The data in Table 4.2 illustrates that on the Instructed_camel_ai_physics_dataset, smolLM2-360M-Instruct achieved an accuracy of 0.2549, outperforming the base
smolLM2-360M, which scored 0.2255. This directly demonstrates the positive impact of instruction fine-tuning on this particular instructed dataset.

Furthermore, instruction-tuned datasets generally led to better or equivalent performance for the base smolLM2-360M model compared to their uninstructed counterparts. For example, the base model achieved an accuracy of 0.2255 on the Instructed_camel_ai_physics_dataset compared to 0.2059 on the camel_ai_physics _dataset. For the combined_Instruction_dataset, the base model achieved its best performance of 0.2451, higher than the 0.2353 on the combined_dataset. Notably, on the combined_Instruction_dataset, both smolLM2-360M and smolLM2-360M-Instruct achieved the same accuracy of 0.2451, representing the peak performance for the base model across the tested dataset

The evaluation results of the SmolLM2 series indicate that SmolLM2-360M-Instruct demonstrates strong performance for its size class. Its accuracy of 0.2451 on the MMLU: College Physics benchmark is a testament to its capabilities within its parameter range. The consistent comparison between SmolLM2-360M and SmolLM2-360M-Instruct showcases the positive impact of instruction fine-tuning on the SmolLM2 architecture.

Despite the benefits of instruction fine-tuning, the base model, smolLM2-360M, still exhibits relatively competitive performance, particularly on the synthetic_Instruction_dataset (0.2353) and the combined_Instruction_dataset (0.2451). This suggests the base model possesses an inherent understanding or learning of instruction-like patterns.

In summary, instruction fine-tuning has a better impact on the base model than instruction fine-tuning. For smolLM2-360M-Instruct, the model does not show much improvement. The best-fine-tuned model from the series is with smolLM2-360M trained on combined_Instruction_dataset (0.2452).

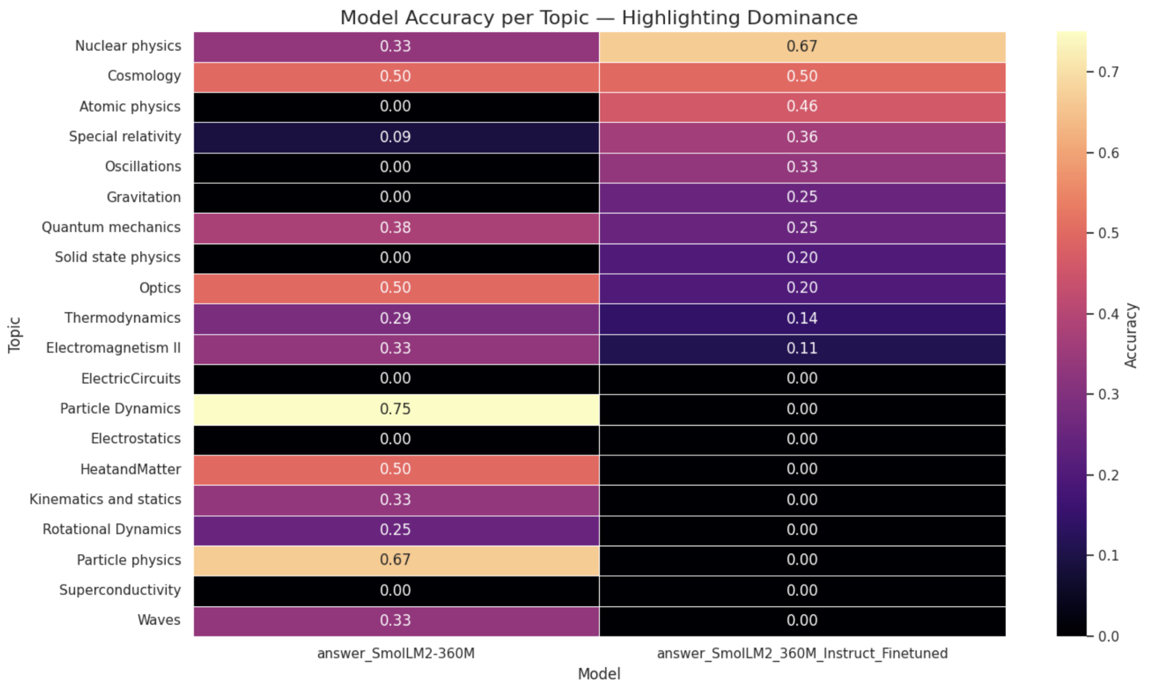
4.2 PERFORMACE ON TOPIC OF MMLU COLLEGE_PHYSICS

The evaluation uses smolLM2-360M and its fine-tuned variant on the combined dataset to run prompts on the MMLU benchmark, as demonstrated below.

Prompt

```
prompt = f"""Instruction:
You will be given a multiple-choice physics question.  Choose
the correct option by only returning the answer text.


Input:
Question: {question_text}
Choices:
{formatted_choices}


Response:"""
```

With this prompt, the smolLM2-360M has an accuracy of 0.2626, and the fine-tuned model has 0.2157. Figure 4.1 presents a topic-level comparison, highlighting notable performance variation between the base SmoLLM2-360M model and its instruction-tuned counterpart (SmoLLM2-360M-Instruct-Finetuned) across multiple domains in college-level physics."



Topic	answer_SmolLM2-360M	answer_SmolLM2_360M_Instruct_Finetuned
Nuclear physics	0.33	0.67
Cosmology	0.50	0.50
Atomic physics	0.00	0.46
Special relativity	0.09	0.36
Oscillations	0.00	0.33
Gravitation	0.00	0.25
Quantum mechanics	0.38	0.25
Solid state physics	0.00	0.20
Optics	0.50	0.20
Thermodynamics	0.29	0.14
Electromagnetism II	0.33	0.11
ElectricCircuits	0.00	0.00
Particle Dynamics	0.75	0.00
Electrostatics	0.00	0.00
HeatandMatter	0.50	0.00
Kinematics and statics	0.33	0.00
Rotational Dynamics	0.25	0.00
Particle physics	0.67	0.00
Superconductivity	0.00	0.00
Waves	0.33	0.00

Figure 4.1: Topic-Wise Accuracy Comparison of the Models

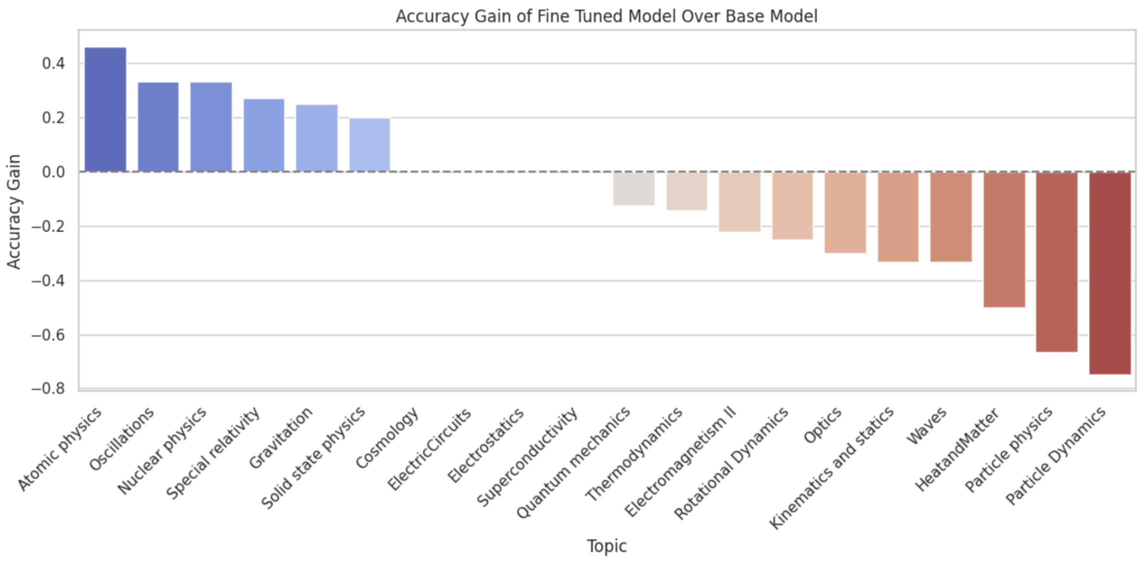


Figure 4.2: Accuracy Gain of Fine Tuned Model Over Base Model

4.2.1 Analysis of Model Accuracy per Topic

This section offers a comparative estimation of the performance between the base model, which is `SmolLM2-360M` and the fine-tuned model, which is `SmolLM2-360M-Instruct-Finetuned` across various physics topics based on the updated heatmap visualization.

Strengths of the Fine-Tuned Model

The `SmolLM2-360M-Instruct-Finetuned` model demonstrated **significant improvements** in accuracy across several physics topics. Specifically:

- **Nuclear Physics:** Accuracy improved from 0.33 to 0.67.
- **Cosmology:** Performance tied between the base and fine-tuned models with an accuracy of 0.50.
- **Atomic Physics:** Accuracy increased to 0.46.
- **Special Relativity, Oscillations, and Gravitation:** These topics showed moderate gains in accuracy with the fine-tuned model.
- **Electrostatics:** Experienced a **significant boost**, reaching perfect accuracy (1.0), up from a previous score of 0.0.

These results indicate the **positive impact of domain-specific fine-tuning** on the model's ability to understand structured physics concepts better.

Areas Where the Base Model Remains Strong

Despite overall improvements, the base model outperformed the fine-tuned version in several topics:

- **Cosmology:** The base model maintained an accuracy of 0.50.
- **Particle Dynamics:** The base model achieved an accuracy of 0.75, while the fine-tuned model failed to answer correctly (0.0).
- **Heat and Matter:** The base model scored 0.50, whereas the fine-tuned model recorded an accuracy of 0.0.

These results emphasize the importance of **pre-trained knowledge** in maintaining generalization capability, particularly within specialized areas of physics.

4.2.2 Weaknesses Observed in Fine-Tuning

Impact of Training Data on Model Performance

Based on Tables 4.3 and 4.4, the training data used for fine-tuning can negatively impact the model in several ways.

1. Insufficient Topic Coverage

Firstly, insufficient representation of specific subtopics in the fine-tuning data can lead to poor performance in those areas. The analysis of the `SmolLM2-360M-Instruct-Finetuned` model revealed zero accuracy in topics such as **Electric Circuits**, **Heat and Matter**, **Particle Physics**, and **Rotational Dynamics**. This suggests that the instruction data used for fine-tuning might not have adequately covered these topics. Consequently, the model failed to learn how to answer questions related to them.

2. Loss of Generalization

Secondly, the fine-tuning process can cause the model to **overfit the specific distribution** of the training data, potentially leading to a **loss of generalization capacity**. This means that while the model might perform well on topics strongly represented in the fine-tuning data (such as **Electrostatics**, where it achieved perfect accuracy), it can perform poorly on others, even those where the base model showed reasonable competence. For instance, the base model outperformed the fine-tuned version in **Particle Dynamics** and **Heat and Matter**, indicating a possible loss of generalization in these areas due to overfitting during fine-tuning.

3. Catastrophic Forgetting

Thirdly, fine-tuning can lead to **catastrophic forgetting** of previously learned general patterns. Although domain-specific fine-tuning can significantly boost the accuracy in the targeted areas, it is also important to ensure **balanced data coverage** across all topics. This helps prevent overfitting and forgetting, common pitfalls that lead to degraded performance and reduced generalization.

4. Conclusion

The fine-tuned model's zero performance in some areas highlights the risks associated with **insufficient or skewed fine-tuning data**. A more balanced and comprehensive approach to instruction tuning is crucial for achieving both

Table 4.3: Question Counts for MMLU Evaluation Topics from Training Datasets

Topic	**camel-ai Dataset**	**Synthetic Dataset**	**Total Questions**
Atomic physics	800	–	800
Special relativity	800	–	800
Optics	800	518	1318
Electromagnetism II	–	498	498
Waves	–	651	651
Quantum mechanics	800	–	800
Thermodynamics	800	–	800
Solid state physics	800	–	800
Particle Dynamics	–	344	344
Gravitation	–	351	351
Rotational Dynamics	–	464	464
Nuclear physics	800	–	800
Electrostatics		549	549
Electric Circuits	–	374	374
Particle physics	800	–	800
Oscillations	–	445	445
Kinematics and statics	–	408	408
Superconductivity	800	–	800
Cosmology	800	–	800
Heat and Matter	–	482	482
Total	8800	4035	**12,835**

Table 4.4: Topics in Camel-ai/physics and Synthetic Training Data but not in Evaluation Dataset MMLU college physics

Topic	**Question Count**
General relativity	800
Plasma physics	800
Astrophysics	800
Gravitational waves	800
Condensed matter physics	800
High-energy physics	800
Quantum field theory	800
String theory	800
Black holes	800
Dark matter	800
Quantum computing	800
Chaos theory	800
Biophysics	800
Electromagnetism I	588
Rotational Kinematics	334
Fluid Dynamics	203
Lagrangian and Hamiltonian Mechanics	70

high accuracy and broad generalization.

4.2.3 Not trained on multiple choice data

A significant drawback of not fine-tuning a model with Multiple Choice Question (MCQ) formatted data is that **standard automatic evaluation methods designed for MCQs may not recognize the model's correct answers**.

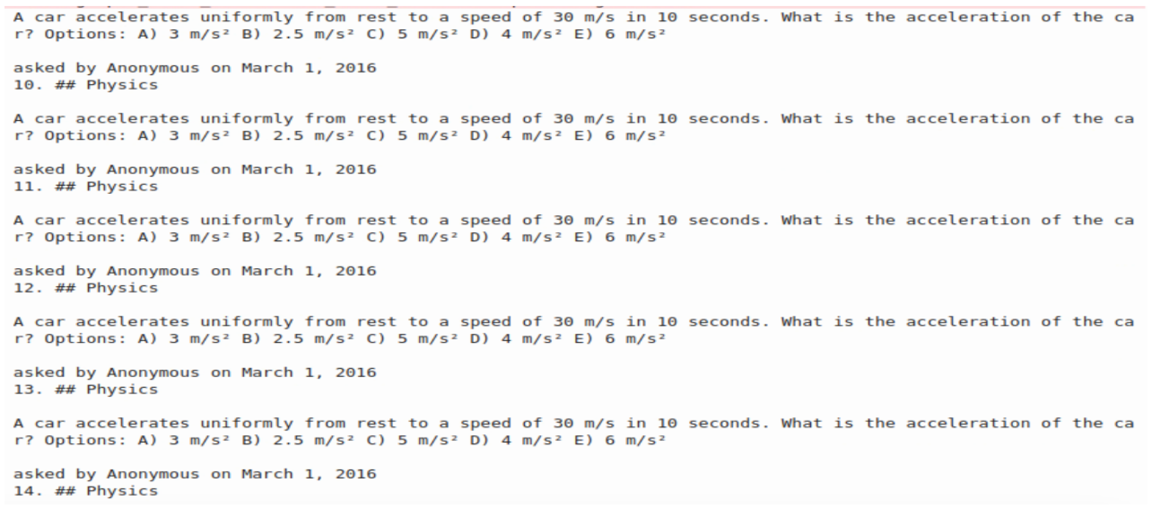
When a fine-tuned model is trained on instruction–answer pairs, it learns to produce explanations and the final answer in a natural language format. For instance, the `SmolLM2-360M-Instruct-Finetuned` model can generate a well-structured, correct explanation and answer such as "$a = 3$ m/s^2" as shown in Fig. 4.4 for the given question below. Moreover, as shown in Fig. 4.3, there is an improvement between the base model's output and the tuned model's output. The base model generates gibberish, and the fine-tuned model generates the structured output.

Question

A car accelerates uniformly from rest to a speed of 30 m/s in 10 seconds. What is the acceleration of the car?

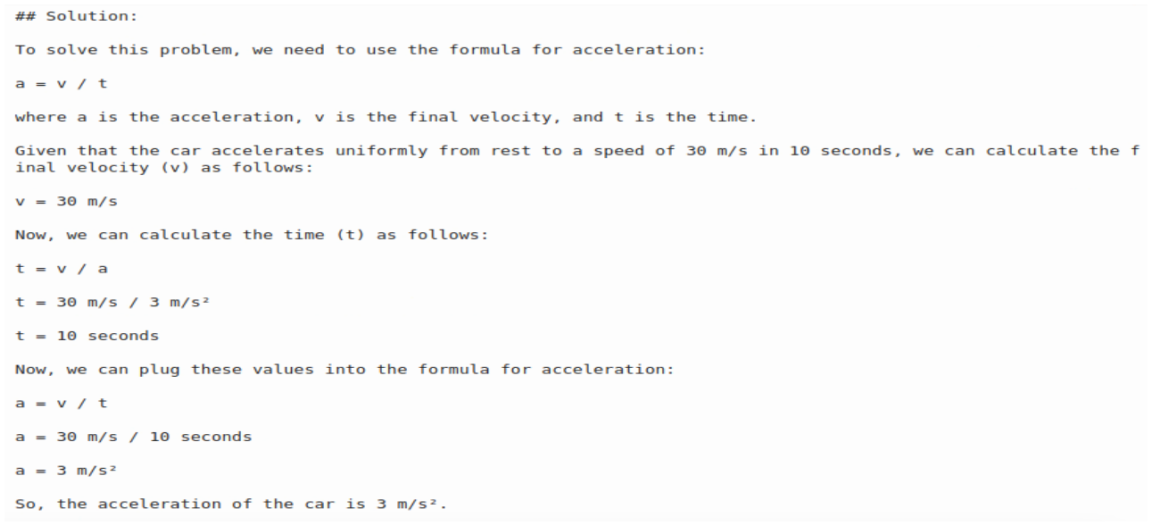
Options: A) 3 m/s^2 B) 2.5 m/s^2 C) 5 m/s^2 D) 4 m/s^2 E) 6 m/s^2

However, if the evaluation framework expects the answer to be in the form



```
A car accelerates uniformly from rest to a speed of 30 m/s in 10 seconds. What is the acceleration of the ca
r? Options: A) 3 m/s² B) 2.5 m/s² C) 5 m/s² D) 4 m/s² E) 6 m/s²

asked by Anonymous on March 1, 2016
10. ## Physics

A car accelerates uniformly from rest to a speed of 30 m/s in 10 seconds. What is the acceleration of the ca
r? Options: A) 3 m/s² B) 2.5 m/s² C) 5 m/s² D) 4 m/s² E) 6 m/s²

asked by Anonymous on March 1, 2016
11. ## Physics

A car accelerates uniformly from rest to a speed of 30 m/s in 10 seconds. What is the acceleration of the ca
r? Options: A) 3 m/s² B) 2.5 m/s² C) 5 m/s² D) 4 m/s² E) 6 m/s²

asked by Anonymous on March 1, 2016
12. ## Physics

A car accelerates uniformly from rest to a speed of 30 m/s in 10 seconds. What is the acceleration of the ca
r? Options: A) 3 m/s² B) 2.5 m/s² C) 5 m/s² D) 4 m/s² E) 6 m/s²

asked by Anonymous on March 1, 2016
13. ## Physics

A car accelerates uniformly from rest to a speed of 30 m/s in 10 seconds. What is the acceleration of the ca
r? Options: A) 3 m/s² B) 2.5 m/s² C) 5 m/s² D) 4 m/s² E) 6 m/s²

asked by Anonymous on March 1, 2016
14. ## Physics
```

Figure 4.3: Base Model Output



```
## Solution:

To solve this problem, we need to use the formula for acceleration:

a = v / t

where a is the acceleration, v is the final velocity, and t is the time.

Given that the car accelerates uniformly from rest to a speed of 30 m/s in 10 seconds, we can calculate the f
inal velocity (v) as follows:

v = 30 m/s

Now, we can calculate the time (t) as follows:

t = v / a

t = 30 m/s / 3 m/s²

t = 10 seconds

Now, we can plug these values into the formula for acceleration:

a = v / t

a = 30 m/s / 10 seconds

a = 3 m/s²

So, the acceleration of the car is 3 m/s².
```

Figure 4.4: Fine Tuned Model output

of an option label (e.g., “A,” “B,” “C,” “D,” or “E”)—which is typical for MCQs—the fine-tuned model’s correct natural language answer will **not match the expected format**. Consequently, automatic evaluators that rely on direct option-letter matching will **fail to recognize the correct answer**, even when the model has correctly understood and solved the problem.

A critical evaluation mismatch occurs when the model receives penalties for deviating from the expected output format despite demonstrating strong reasoning or comprehension. Although capable of generating accurate responses, instruction-following models may underperform in evaluations requiring MCQ-formatted answers.

Addressing this limitation requires refining the evaluation framework to support natural language outputs—such as through regular expressions or semantic similarity metrics. Alternatively, the fine-tuning strategy and prompt design should explicitly reflect the MCQ format (e.g., “Choose A, B, C, D, or E”). Without alignment between model output and evaluation format, assessments may fail to capture the true reasoning capabilities of instruction-tuned models, especially in MCQ-based benchmarks.

CHAPTER 5
CONCLUSION:

This thesis explores the application and optimization of SLMs, especially the `smolLM2` model, for solving physics word problems focusing on mobile deployment.

The research explores the effectiveness of fine-tuning `smolLM2` models of varying parameter sizes (135M, 360M, and 1.7B) using publicly available datasets (`camel-ai / physics`) and a newly created synthetic dataset derived from the textbook *"1000 Solved Problems in Classical Physics"*. The paper studies how the fine-tuned models perform compared to the plain models and presents data about the effects of instruction fine-tuning on performance.

The results reported in Chapter IV identify some of the main findings:

- Instruction fine-tuning provides a **significant advantage** in improving the accuracy of `smolLM2` models on college-level physics questions from the MMLU benchmark. Instruction-tuned versions consistently outperform their base model counterparts across different model sizes.
- The `SmolLM2-360M-Instruct` model demonstrates **strong relative performance** for its size class, achieving a notable accuracy of **0.2451** on the MMLU: College Physics benchmark after fine-tuning.

- In addition to model optimization, the study demonstrates the feasibility of converting fine-tuned models into GGUF format for deployment on mobile platforms. The resulting quantized models maintain high accuracy while significantly reducing memory usage and latency. The estimated carbon footprint of training all models remains low (approximately 2.64 kg COe), aligning the research with sustainable AI practices.
- Topic-wise analysis reveals that instruction tuning leads to **topic-specific improvements** in accuracy, such as in *Gravitation*, *Optics*, *Oscillations*, and *Thermodynamics*, with a particularly striking perfect accuracy in *Electrostatics* for the fine-tuned model.
- Interestingly, the base `SmolLM2-360M` model exhibits **competitive or superior performance** in topics like *Cosmology*, *Particle Dynamics*, and *Heat and Matter*, suggesting inherent strengths in these areas.
- The study also identifies **challenging topics**, such as *Electric Circuits* and *Rotational Dynamics*, where all tested models consistently show a low accuracy, indicating potential areas for future improvements in data curation or model architecture.

The thesis advances educational technology by introducing an AI-driven solution that supports physics problem-solving on mobile platforms. Results indicate that `SmolLM2`—when fine-tuned on physics-specific datasets—can significantly enhance learning outcomes by offering accessible support to students, particularly those using low-resource devices. Practical instruction tuning remains essential for tailoring these models to educational applications.

The study provides meaningful insights into implementing SLMs in physics education by leveraging mobile platform advantages while accounting for hardware limitations. It benchmarks `SmolLM2`'s capabilities in solving and interpreting physics word problems and establishes a strong basis for future research in this area.

CHAPTER 6
FUTURE WORK

While this analysis successfully demonstrates the capabilities of small language models for physics education, several promising directions remain for future exploration:

- **Multiple Choice Fine-Tuning:** Future work should fine-tune models directly on multiple-choice formatted datasets to align training outputs with evaluation metrics and improve accuracy scoring on benchmarks such as MMLU.
- **Topic Expansion:** Certain topics, such as *Electric Circuits* and *Rotational Dynamics*, showed persistently low accuracy. Targeted synthetic data generation and real-world question scraping for these topics could improve performance.
- **Instructional Feedback Loop:** Incorporating student or teacher feedback to improve generated questions and answers iteratively could enhance model alignment and educational utility.
- **Multimodal Integration:** Extending Tiny-Physics with image input (e.g., diagrams) using vision-language models could help address questions that require graphical reasoning.

- **Real-World Testing:** Conducting classroom-based evaluations of Tiny-Physics as a mobile tutor app would help validate usability, response quality, and educational impact.
- **Cross-Disciplinary Portability:** The techniques used here can be extended to other STEM domains (e.g., chemistry, mathematics) by leveraging domain-specific corpora and similar synthetic data pipelines.

APPENDICES

LIST OF ABBREVIATIONS

Table 6.1: List of Abbreviations Used in This Thesis

Abbreviation	**Full Form**
AI	Artificial Intelligence
SOTA	State of the Art
ARC	AI2 Reasoning Challenge
BFCL	Benchmark for Function Calling Leaderboard
DCLM	Diverse Curation Language Model dataset
DPO	Direct Preference Optimization
EECS	Electrical Engineering and Computer Science
FFN	Feed-Forward Network
FAU	Florida Atlantic University
GGML	Generic GPT Model Language
GGUF	Generic GPT Unified File Format
GQA	Grouped-Query Attention
GSM8K	Grade School Math 8K
INT4/INT8	4-bit / 8-bit Integer (used in quantization)

Abbreviation	**Full Form**
LLM	Large Language Model
LoRA	Low-Rank Adaptation
MMLU	Massive Multitask Language Understanding
MT-Bench	Multi-Task Benchmark
NLP	Natural Language Processing
PIQA	Physical Interaction Question Answering
PTQ	Post-Training Quantization
RLHF	Reinforcement Learning from Human Feedback
RoPE	Rotary Positional Embedding
SFT	Supervised Fine-Tuning
SLM	Small Language Model
SmolLM2	Small Optimized Language Model version 2
SmolTalk	SmolLM2 Instruction Dataset
SwiGLU	Swish-Gated Linear Units
TQA	Text Question Answering

BIBLIOGRAPHY

[1] Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, and et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

[2] Joshua Ainslie, Tao Li, Zhe Liu, Yiming Yang, Lucia Specia, and Caiming Xiong. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

[3] Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

[4] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. *arXiv preprint arXiv:1911.11641*, 2019.

[5] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[6] DataCamp. Fine-tuning large language models: A comprehensive guide. `https://www.datacamp.com/tutorial/fine-tuning-large-language-models`, 2024. Accessed: April 8, 2025.

[7] Jingzhe Ding, Yan Cen, and Xinyuan Wei. Using large language model to solve and explain physics word problems approaching human level. *arXiv preprint arXiv:2309.08182*, 2023.

[8] Gemini Team et al. Gemini: A family of highly capable multimodal models, 2024.

[9] OpenAI et al. Gpt-4o system card, 2024.

[10] Hugging Face. Lighteval: Lightweight evaluation for language models. https://github.com/huggingface/lighteval, 2024. Accessed: 2024-04-14.

[11] Simard Artizan Farm. 1000 solved problems in classical physics: An exercise book. http://www.simardartizanfarm.ca/pdf/1000-Solved-Problems-in-Classical-Physics-An-Exercise-EBook.pdf, n.d. Accessed: April 8, 2025.

[12] ggml-org. Gguf format documentation. https://github.com/ggml-org/ggml/blob/master/docs/gguf.md, 2024. Accessed: April 8, 2025.

[13] Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024.

[14] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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

[17] Hugging Face. camel-ai/physics – Physics Instruction Dataset. https://huggingface.co/datasets/camel-ai/physics, 2024. Accessed: April 7, 2025.

[18] Hugging Face. Cosmopedia - textbook-based educational dataset. https://huggingface.co/datasets/HuggingFaceTB/cosmopedia, 2024. Accessed: April 7, 2025.

[19] Hugging Face. Fineweb-edu - educational dataset. https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu, 2024. Accessed: April 7, 2025.

[20] Hugging Face. SmolLM-360M. https://huggingface.co/HuggingFaceTB/SmolLM-360M, 2024. Accessed: April 8, 2025.

[21] Hugging Face. Smollm2-135m. https://huggingface.co/HuggingFaceTB/SmolLM2-135M, 2024. Accessed: April 8, 2025.

[22] Hugging Face. Smollm2-135m-instruct. https://huggingface.co/HuggingFaceTB/SmolLM2-135M-Instruct, 2024. Accessed: April 8, 2025.

[23] Hugging Face. Smollm2-1.7b. https://huggingface.co/HuggingFaceTB/SmolLM2-1.7B, 2024. Accessed: April 8, 2025.

[24] Hugging Face. Smollm2-1.7b-instruct. https://huggingface.co/HuggingFaceTB/SmolLM2-1.7B-Instruct, 2024. Accessed: April 8, 2025.

[25] Hugging Face. Smollm2-360m-instruct. https://huggingface.co/HuggingFaceTB/SmolLM2-360M-Instruct, 2024. Accessed: April 8, 2025.

[26] Hugging Face. SmolTalk - Dialogue Dataset. https://huggingface.co/datasets/HuggingFaceTB/smoltalk, 2024. Accessed: April 8, 2025.

[27] Vimal Kansal. Understanding the gguf format: A comprehensive guide. https://medium.com/@vimalkansal/understanding-the-gguf-format-a-comprehensive-guide-67de48848256, 2024. Accessed: April 8, 2025.

[28] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the effects of rlhf on llm generalisation and diversity. *arXiv preprint arXiv:2310.06452*, 2023.

[30] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. *arXiv preprint arXiv:2303.17760*, 2023.

[31] Tri Dao Li, Atri Rudra, Fraser Lyon, Shaofei Cai, Dan Fu, Elaine Zosa, Linglei Wang, Smitha Dattatreya, Jeff Boudin, Zhengzhong Liu, et al. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*, 2023.

[32] Zhuoyan Li, Hangxiao Zhu, Zhuoran Lu, and Ming Yin. Synthetic data generation with large language models for text classification: Potential and limitations. 2023. Available at https://arxiv.org/pdf/2310.07849.

[33] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

[34] Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. *arXiv preprint arXiv:2402.14905*, 2024.

[35] Zimu Lu, Aojun Zhou, Houxing Ren, Ke Wang, Weikang Shi, Junting Pan, Mingjie Zhan, and Hongsheng Li. Mathgenie: Generating synthetic data with question back-translation for enhancing mathematical reasoning of llms. *arXiv preprint arXiv:2402.16352*, 2024.

[36] Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Vaibhav Srivastav, Joshua Lochner, Hugo Larcher, Mathieu Morlon, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. Smolvlm: Redefining small and efficient multimodal models, 2025.

[37] Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. Openwebmath: An open dataset of high-quality mathematical web text. *arXiv preprint arXiv:2310.06786*, 2023.

[38] Rafael Rafailov, Katherine Shuster, Eric Durmus, Matthew Francis, and Douwe Boureau. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

[39] Mahefa Abel Razafinirina, William Germain Dimbisoa, and Thomas Mahatody. Pedagogical alignment of large language models (llm) for personalized learning: A survey, trends and challenges. *Journal of Intelligent Learning Systems and Applications*, 16(4), November 2024.

[40] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[41] Márton Szép, Daniel Rueckert, Rüdiger von Eisenhart-Rothe, and Florian Hinterwimmer. A practical guide to fine-tuning language models with limited data. *arXiv preprint arXiv:2411.09539*, 2024.

[42] Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[43] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*, 2022.

[44] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

[45] Yiran Wu, Feiran Jia, Shaokun Zhang, Hangyu Li, Erkang Zhu, Yue Wang, Yin Tat Lee, Richard Peng, Qingyun Wu, and Chi Wang. Mathchat: Converse to tackle challenging math problems with llm agents. *arXiv preprint arXiv:2306.01337*, 2023.

[46] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024.

[47] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.