UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LUCCAS DA SILVA LIMA

Evaluating Small and Large Language Models for Natural Language Inference: A Performance–Efficiency Analysis

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Engineering

Advisor: Prof. Dr. Luciana Regina Bencke

Porto Alegre
July 2026

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Profª. Márcia Cristina Bernardes Barbosa
Vice-Reitor: Prof. Pedro de Almeida Costa
Pró-Reitora de Graduação: Profª. Nadya Pesce da Silveira
Diretor do Instituto de Informática: Prof. Luciano Paschoal Gaspary
Coordenador do Curso de Engenharia de Computação: Prof. Paulo Francisco Butzen
Bibliotecário-chefe do Instituto de Informática: Alexsander Borges Ribeiro

“Programs must be written for people to read,
and only incidentally for machines to execute.”
— Sir Harold Abelson

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Prof. Dr. Luciana Regina Bencke, for her dedication, patience, and guidance throughout the development of this work. Her careful supervision, detailed feedback, and steady support were essential to the completion of this research.

I would also like to thank the Universidade Federal do Rio Grande do Sul for providing access to high-quality public education. My gratitude extends to the Instituto de Informática for offering an excellent academic environment and infrastructure during my undergraduate studies. I am equally thankful to all professors who shared their knowledge and contributed to my education, as well as to the administrative and technical staff whose essential work supports the university every day.

A special acknowledgment goes to the PET Computação, which I joined at the end of 2021. Being part of the program played a fundamental role in my academic, professional, and personal development, allowing me to improve practical and organizational skills through several projects while also building meaningful friendships.

I am deeply grateful to my family for their unconditional support, encouragement, and constant belief in me throughout this journey. In particular, I would like to thank my sister, Gabrielly, for her companionship and support during important moments of my academic life. More than that, she has always been a true example for me, inspiring me through her strength, kindness, and determination. I also want to thank my girlfriend, Ana Vitória, for her love, patience, encouragement, and for always being by my side, especially during the most challenging moments of this journey. Without the support, understanding, and constant presence of both of them, I truly would not have been able to accomplish this work.

Finally, I would like to thank my friends, both those who have been part of my life for many years and those I met during university. Thank you for the companionship, shared study sessions, conversations, and especially for all the memorable moments that made this academic journey much lighter and more meaningful.

ABSTRACT

Large Language Models (LLMs) have demonstrated strong performance in Natural Language Processing tasks, but their computational and environmental costs remain significant challenges for scalable deployment. This study evaluates the trade-offs between performance and efficiency by comparing Small Language Models (SLMs) and a Large Language Model (LLM) baseline in Natural Language Inference (NLI) tasks. Experiments were conducted on the SNLI dataset using instruction-tuned decoder-only models under zero-shot, few-shot, and parameter efficient fine-tuning configurations. The evaluation considered both predictive performance metrics, including accuracy and F1-score, and efficiency-related measures such as inference latency, memory consumption, energy usage, and estimated carbon emissions. Additionally, robustness analyses were performed through semantically equivalent input perturbations and repeated inference consistency tests. The results show that compact language models achieve competitive performance in NLI tasks. Specifically, following parameter-efficient fine-tuning, all evaluated SLMs surpassed the zero-shot baseline of the selected 120B-parameter LLM, achieving accuracy gains of up to 12.7%, with models like Mistral 7B improving from 55.32% to 87.00%. Furthermore, fine-tuned architectures in certain cases offered significant reductions in computational and environmental costs, such as the Phi-3 3.8B model requiring nearly four times less energy than its base version. These findings suggest that smaller architectures represent a practical avenue for exploring computational efficiency in NLI applications when resources are limited.

Keywords: Small Language Models. Natural Language Inference. Computational Efficiency. Sustainable AI.

LIST OF ABBREVIATIONS AND ACRONYMS

AI	Artificial Intelligence
API	Application Programming Interface
CoT	Chain-of-Thought
CO_2	Carbon Dioxide
CPU	Central Processing Unit
CSV	Comma-Separated Values
DPO	Direct Preference Optimization
GPU	Graphics Processing Unit
kWh	Kilowatt-Hour
LLM	Large Language Model
SLM	Small Language Model
MoE	Mixture-of-Experts
NLP	Natural Language Processing
NLI	Natural Language Inference
NLU	Natural Language Understanding
RAM	Random Access Memory
RTE	Recognizing Textual Entailment
SFT	Supervised Fine-Tuning
BERT	Bidirectional Encoder Representations from Transformers
BF16	Bfloat16
LoRA	Low-Rank Adaptation
LLMs	Large Language Models
PEFT	Parameter-Efficient Fine-Tuning
RLHF	Reinforcement Learning from Human Feedback

SLMs	Small Language Models
SNLI	Stanford Natural Language Inference
C-RLFT	Conditioned Reinforcement Learning from Fine-Tuning
$kgCO_2e$	Kilograms of Carbon Dioxide Equivalent
MSCINLI	Multi-domain Scientific Natural Language Inference
MultiNLI	Multi-Genre Natural Language Inference

LIST OF FIGURES

Figure 4.1 Overview of the methodology adopted in this study. 21

Figure 5.1 Zero-shot prompting accuracy across the evaluated SLMs 38
Figure 5.2 Zero-shot prompting F1 Macro across the evaluated SLMs..... 39
Figure 5.3 Fixed few-shot prompting accuracy across the evaluated SLMs..... 40
Figure 5.4 Fixed few-shot prompting F1 Macro across the evaluated SLMs 40
Figure 5.5 Random few-shot prompting accuracy across the evaluated SLMs 41
Figure 5.6 Random few-shot prompting F1 Macro across the evaluated SLMs..... 41
Figure 5.7 Total latency across the evaluated SLMs..... 43
Figure 5.8 Total energy spent across the evaluated SLMs 43
Figure 5.9 Accuracy across all trained SLMs 45
Figure 5.10 F1 Macro across all trained SLMs..... 45
Figure 5.11 Accuracy across all trained SLMs using Few-shot prompt 46
Figure 5.12 F1 Macro across all trained SLMs using Few-shot prompt..... 46
Figure 5.13 Confusion matrices obtained for all fine-tuned Small Language Models. .. 48

Figure 6.1 Best model configurations 56
Figure 6.2 Comparison of execution duration across evaluated models..... 56
Figure 6.3 Comparison of total energy consumption across evaluated models 57

LIST OF TABLES

Table 4.1 Total number of instances and proportional label distribution across the training, validation, and test splits of the SNLI subset 23
Table 4.2 Parameter sizes and primary references of the five instruction-tuned Small Language Models selected for the experimental evaluation 25
Table 4.3 Example of the hardware and environmental telemetry schema captured by CodeCarbon during a representative inference execution, detailing the specific metrics logged for power consumption, estimated carbon emissions, and system utilization 32

Table 5.1 Summary of predictive performance, operational execution metrics, and estimated hardware energy consumption for the GPT-OSS-120B baseline evaluated under a zero-shot configuration 37
Table 5.2 Comparison of predictive performance metrics achieved by the evaluated Small Language Models across zero-shot, fixed few-shot, and random few-shot prompting strategies 42
Table 5.3 Environmental and operational efficiency metrics across the evaluated language models, detailing execution duration, component-level energy consumption and total estimated carbon emissions during inference 44
Table 5.4 Comparison of individual model predictions on specific *contradiction* instances from the test set 50
Table 5.5 Results of the prefix-completion test for potential benchmark familiarity, presenting the average semantic similarity, lexical overlap, and normalized Levenshtein distance between the generated continuations of the Small Language Models and the ground-truth SNLI suffixes 52
Table 5.6 Results of the robustness analysis evaluating the impact of semantic-preserving noun synonym substitutions on the predictive performance of the fine-tuned Small Language Models compared to the original SNLI test subset 53

Table 6.1 Comprehensive summary of predictive performance across all evaluated Small Language Models under zero-shot, few-shot, and fine-tuned configurations, with the best-performing setup for each architecture highlighted in bold, alongside the GPT-OSS-120B zero-shot baseline for comparison 55
Table 6.2 Comparison of environmental and operational efficiency metrics between original and fine-tuned Small Language Models, highlighting changes in execution duration, estimated carbon emissions, and component-level energy consumption 57

CONTENTS

1 INTRODUCTION **11**
1.1 Objective **12**
1.2 Structure **12**
2 BACKGROUND **13**
2.1 Natural Language Processing and the Transformer Paradigm **13**
2.2 Instruction-Tuning **14**
2.3 Prompt Engineering **15**
2.4 Evaluation Metrics for Classification **16**
2.4.1 Accuracy 16
2.4.2 F1-Score 17
3 RELATED WORK **18**
3.1 Natural Language Inference as an Evaluation Benchmark **18**
3.1.1 Recent Studies 19
4 METHODOLOGY **21**
4.1 Dataset Selection **21**
4.2 Model Selection **23**
4.2.1 Small Language Models 24
4.2.2 Large Language Model 25
4.3 Experimental Environment **26**
4.4 Supervised Instruction Tuning with LoRA **27**
4.5 Inferences **28**
4.5.1 Zero-Shot Prompting 29
4.5.2 Few-Shot Prompting 30
4.6 Computational Efficiency Measurements **30**
4.7 Potential Benchmark Familiarity Analysis **32**
4.7.1 Experimental Design and Prompt Engineering 32
4.7.2 Evaluation Metrics 33
4.8 Robustness Analysis **34**
5 EXPERIMENTS AND RESULTS **36**
5.1 LLM Baseline **36**
5.2 Prompt Strategies **37**
5.2.1 Zero-shot 38
5.2.2 Fixed Few-shot 39
5.2.3 Random Few-shot 40
5.2.4 Environmental Efficiency 42
5.3 Supervised Instruction Tuning **44**
5.3.1 Zero-shot Strategy 44
5.3.2 Few-shot Strategy 45
5.3.3 Confusion Matrices 47
5.4 Potential Benchmark Familiarity Analysis **51**
5.5 Robustness Analysis **52**
6 DISCUSSION **54**
6.1 Predictive Performance **54**
6.2 Environmental Performance **56**
7 CONCLUSION **59**
7.1 Next Steps **60**
7.2 Use of Artificial Intelligence **61**
REFERENCES **63**

1 INTRODUCTION

Language models have become central to modern natural language processing (NLP), enabling breakthroughs in fundamental tasks such as text classification, question answering, and natural language inference. In recent years, deep transformer-based architectures have largely replaced traditional approaches, achieving strong performance gains through increased model scale and training data, but at the cost of significantly increased computational cost, as demonstrated in large-scale transformer models (BROWN et al., 2020).

Despite their effectiveness, large language models (LLMs) present significant practical limitations. Training and deploying models with hundreds of billions of parameters require extensive computational resources, specialized hardware, and high energy consumption, raising concerns related to environmental impact, economic accessibility, and real-world scalability (STRUBELL; GANESH; MCCALLUM, 2019; PATTERSON et al., 2021). In response, recent research has increasingly focused on Small Language Models (SLMs), which aim to balance performance and efficiency through architectural optimizations, parameter reduction, and improved training strategies (WANG et al., 2024a). Techniques such as knowledge distillation (HINTON; VINYALS; DEAN, 2015; SANH et al., 2019) and quantization (ZAFRIR et al., 2019) have shown that more compact and computationally efficient models can retain much of the performance of their larger or higher-precision counterparts.

Natural Language Inference (NLI) is a NLP a widely used benchmark for evaluating whether models can identify semantic relationships between sentence pairs, typically categorized as entailment, contradiction, or neutral (BOWMAN et al., 2015). Previous studies suggest that, under specific task and deployment conditions, smaller language models can achieve performance close to that of larger reference models, particularly when supported by task-specific fine-tuning or knowledge distillation (SANH et al., 2019). In agentic systems, SLMs have also been argued to be suitable for specialized and repetitive tasks, where their lower computational cost may make them preferable to general-purpose LLMs (BELCAK et al., 2025). These findings suggest that competitive performance on specific language understanding tasks does not necessarily require massive model scale and highlight the relevance of efficiency-oriented evaluation.

GPT-OSS (OpenAI, 2025) was selected as a large open-weight baseline under the expectation that its scale would provide a strong zero-shot reference point against

smaller language models. However, preliminary results showed that some SLMs achieved comparable or even superior performance in the same zero-shot setting. This motivated a closer analysis of GPT-OSS itself, particularly its Mixture-of-Experts (MoE) architecture, in which only part of the model is activated during inference. Besides raising questions about the relationship between scale and task performance, this architecture also suggests potential computational and environmental efficiency benefits.

1.1 Objective

This work evaluates the feasibility of replacing Large Language Models with Small Language Models in selected NLP tasks. While the adoption of a more compact architecture is largely motivated by the desire to reduce computational cost, energy consumption, and deployment overhead, this study focuses on empirically measuring these expected efficiency gains against actual predictive performance. To this end, the research reviews recent work on performance–efficiency trade-offs across models of different scales and presents an experimental comparison between SLMs and an LLM in terms of accuracy and computational efficiency on NLI tasks.

Furthermore, the analysis investigates the specific impact of parameter-efficient adaptation on computational overhead and predictive accuracy. It also explores whether model performance may be influenced by prior familiarity with benchmark-specific data and evaluates robustness against text perturbations.

1.2 Structure

Chapter 2 presents the fundamental concepts related to language models and Natural Language Inference. Chapter 3 reviews recent studies and related work involving Small Language Models and efficiency-oriented NLP research. Chapter 4 describes the adopted methodology and experimental setup. Chapter 5 details the empirical experiments and presents the obtained results. Chapter 6 provides a comprehensive discussion, consolidating the predictive and environmental performance findings across all evaluated models. Finally, Chapter 7 concludes the work and outlines potential future directions.

2 BACKGROUND

This chapter details the theoretical foundations underlying modern language modeling. It covers the core concepts of NLP, the mathematical formulation of instruction-tuning, the principles of prompt engineering, and the metrics utilized for classification performance evaluation. Together, these mechanisms support general-purpose architectures in addressing language tasks like NLI, offering a practical basis for empirical evaluation.

2.1 Natural Language Processing and the Transformer Paradigm

Natural Language Processing is an interdisciplinary subfield of artificial intelligence (AI) and computational linguistics dedicated to enabling computational systems to process, interpret, and generate human language. At its core, NLP converts unstructured textual data into structured, machine-readable representations, mapping the non-deterministic syntax and semantics of natural language into mathematical spaces. This process requires resolving complex linguistic phenomena, such as polysemy, syntactic ambiguity, and context-dependent semantics, allowing models to perform operations ranging from structural parsing to abstract semantic reasoning.

Modern NLP is predominantly built upon the Transformer architecture (VASWANI et al., 2017), which operates via the multi-head self-attention mechanism. Unlike sequential architectures, the Transformer computes dynamic, bidirectional dependencies between all tokens in a sequence simultaneously, mapping how each word relates to every other word within a given context.

Under this paradigm, language processing is framed as an autoregressive token prediction task. Given a sequence of tokens, the network computes a probability distribution over a vocabulary to predict the most likely subsequent token. Formally, for a sequence of tokens $T = (t_1, t_2, \ldots, t_n)$, the joint probability of the sequence is modeled as the product of conditional probabilities, as expressed in Equation 2.1:

$$P(T) = \prod_{i=1}^{n} P(t_i \mid t_1, t_2, \ldots, t_{i-1}) \tag{2.1}$$

By scaling this generative objective across massive datasets, the model internalizes syntactic structures, facts, and latent semantic relationships. Consequently, diverse

downstream tasks, such as translation, classification, and logical inference, are unified under a single generative framework, where textual generation serves as the mechanism for computational reasoning.

2.2 Instruction-Tuning

Pretrained language models are primarily optimized through self-supervised learning objectives, typically autoregressive next-token prediction over massive, uncurated text corpora. While this pretraining phase allows the model to capture rich statistical regularities, syntax, and world knowledge, the resulting base model is fundamentally a text completer. Consequently, it often struggles to behave as a predictable assistant or to follow explicit task directions, frequently responding to a query by continuing the text or generating structurally irrelevant content.

Instruction-tuning is the process of aligning these pretrained base models with human intent by fine-tuning them on a structured dataset composed of explicit instructions and their corresponding ground-truth responses. This phase bridges the gap between next-token prediction and functional utility.

From an architectural and training perspective, instruction-tuning typically employs Supervised Fine-Tuning (SFT). Given an instruction input sequence X and a target response sequence Y, where $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_m)$, the model is optimized using a conditional language modeling objective. The loss function minimizes the negative log-likelihood of the target tokens sequentially, conditioned on both the prompt prefix and the preceding target tokens, as expressed in Equation 2.2:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{t=1}^{m} \log P_\theta(y_t \mid x_1, \ldots, x_n, y_1, \ldots, y_{t-1}) \tag{2.2}$$

where θ represents the trainable parameters of the model. Crucially, during gradient descent, the loss computation is masked so that optimization gradients are only calculated for the response tokens (Y), preventing the model from altering its internal representations based on the instruction formatting (X) itself.

Modern instruction-tuning routines often extend beyond SFT by incorporating preference optimization frameworks, such as Reinforcement Learning from Human Feedback (RLHF) or Direct Preference Optimization (DPO). These methods further refine the model's behavior by optimizing a reward function based on paired human preferences

(choosing an ideal response over an undesirable one), ensuring that the output is not only accurate but also helpful, harmless, and honest.

For compact architectures or SLMs, instruction-tuning is vital. Since these models have a more constrained parameter budget than LLMs, structural alignment ensures that their limited capacity is efficiently directed toward reasoning and executing tasks rather than decoding open-ended conversational structures.

2.3 Prompt Engineering

While instruction-tuning alters the internal weights of a model to make it responsive to commands, Prompt Engineering is the empirical discipline of designing, structuring, and optimizing the textual input provided to a language model to maximize its performance on a specific downstream task. In a unified prompting framework, the input prompt serves as the runtime environment that guides the model's autoregressive generation toward the desired latent space.

For decoder-only architectures, the prompt dictates the initial context window. The model processes the input sequence through its self-attention mechanism, establishing semantic dependencies between the task description, the input data, and the expected format. Through strategic composition of the context window, prompt engineering allows the control of downstream outputs without altering the frozen network weights (θ).

The configuration of the prompt is critical to anchoring the model's label space. By explicitly setting boundaries, keywords, and output constraints within the prefix, the engineering of the input syntax ensures that the token generation trajectory maps accurately to the target objective. This conditioning directly mitigates common generative anomalies, such as structural non-compliance, open-ended conversational drift, and output parsing failures.

Furthermore, advanced conditioning techniques, such as Chain-of-Thought (CoT) prompting, can be integrated into the prompt structure to explicitly guide the internal token generation mechanics. By instructing the model to generate sequential reasoning traces prior to producing the final answer, the prompt enforces a step-by-step state transition within the multi-head attention layers. This intermediate allocation of computational budget shifts the latent representations progressively toward a more accurate and logically stable token selection.

In resource-constrained environments or comparative studies between SLMs and

LLMs, proper prompt alignment ensures that variations in predictive accuracy reflect the actual difference in structural reasoning capacity between the models rather than a superficial failure to understand the execution constraints.

2.4 Evaluation Metrics for Classification

To rigorously quantify the performance of language models on discriminative and classification benchmarks, such as NLI, the generated textual outputs are mapped back to discrete classes and evaluated against ground-truth labels using statistical metrics. The fundamental metrics used to evaluate these classification tasks are Accuracy and the F1-Score, both computed from the elements of a confusion matrix: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

2.4.1 Accuracy

Accuracy is the most intuitive performance metric, representing the proportion of correctly predicted instances out of the total number of cases evaluated. Formally, it is defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

where the constituent terms of the formulation define the operational outcomes within the classification framework. True Positives (TP) denote the instances where the model correctly predicts the positive target class, while True Negatives (TN) represent the cases where the model correctly identifies the negative or alternative class, both aligning perfectly with the ground-truth labels. Conversely, errors in prediction are captured by False Positives (FP), which occur when the model incorrectly predicts the positive class for an actual negative instance (Type I error), and False Negatives (FN), which represent instances where the model mistakenly predicts the negative class for a true positive instance (Type II error).

While accuracy provides a straightforward assessment of global correctness, it exhibits a critical limitation: it is highly sensitive to class imbalance. In datasets where one class severely dominates, a naive model can achieve a deceptively high accuracy by simply predicting the majority class for all instances, failing to reflect actual discriminatory

performance.

2.4.2 F1-Score

To mitigate the limitations of accuracy in skewed distributions, the F1-Score is employed. The F1-Score provides a balanced measure of a model's performance by calculating the harmonic mean of two complementary metrics: Precision and Recall.

Precision measures the exactness of the model, quantifying the proportion of predicted positive instances that are truly positive:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.4}$$

Recall (or Sensitivity) measures the completeness of the model, quantifying the proportion of actual positive instances that were correctly identified:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.5}$$

The F1-Score unifies these two properties into a single metric. By utilizing the harmonic mean rather than the arithmetic mean, the F1-Score penalizes extreme imbalances between Precision and Recall, ensuring that a high score requires both metrics to be robust. It is mathematically expressed as:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN} \tag{2.6}$$

In multi-class evaluation scenarios, such as NLI (which typically contains three classes), the F1-Score can be aggregated across all categories using macro-averaging (treating all classes equally) or micro-averaging (weighting by class frequency), providing a comprehensive view of structural reasoning robustness across distinct semantic relations.

3 RELATED WORK

In this chapter, we motivate the discussion of Natural Language Inference as an evaluation task and provide a concise review of key developments related to Small Language Models, highlighting recent contributions.

3.1 Natural Language Inference as an Evaluation Benchmark

Natural Language Inference, also known as Recognizing Textual Entailment (RTE), is a fundamental task in Natural Language Processing that aims to evaluate a model's ability to reason about the semantic relationship between two sentences. Given a *premise* and a *hypothesis*, the model must determine whether the hypothesis is *entailed* by the premise, *contradicted* by it, or *neutral* with respect to it (DAGAN; GLICKMAN; MAGNINI, 2006; BOWMAN et al., 2015). This formulation explicitly targets semantic understanding and logical reasoning, rather than surface-level pattern matching.

The Stanford Natural Language Inference (SNLI) dataset (BOWMAN et al., 2015) played a key role in enabling large-scale supervised training for NLI. SNLI consists of over 570,000 sentence pairs annotated with entailment, contradiction, or neutral labels, constructed using human-written hypotheses grounded in image captions. This design encourages models to capture nuanced semantic relations while minimizing annotation artifacts. Subsequent datasets, such as Multi-Genre Natural Language Inference (MultiNLI) (WILLIAMS; NANGIA; BOWMAN, 2018), extended this framework to multiple text genres, further establishing NLI as a standard benchmark for evaluating general language understanding. More recently, specialized datasets such as MSCINLI (SADAT; CARAGEA, 2024) have adapted NLI to scientific domains, providing sentence pairs extracted from research articles across multiple disciplines. These scientific NLI datasets enable the study of domain-specific reasoning and domain shift, highlighting the challenges of applying general NLI models to specialized texts.

Beyond serving as a benchmark for Natural Language Understanding (NLU), NLI has been instrumental in the advancement of representation, transfer, and multi-task learning. Success in this task serves as a proxy for a model's ability to capture logical relationships that extend beyond simple surface-level word co-occurrences. Furthermore, research demonstrates that proficiency in NLI can significantly improve performance in downstream tasks, such as fact verification and citation intent classification (SADAT;

CARAGEA, 2024). Within this study, NLI provides the ideal framework to investigate whether Small Language Models can maintain the semantic integrity required for precise inference or if parameter reduction inherently compromises these complex reasoning capabilities.

3.1.1 Recent Studies

Subramanian, Elango and Gungor (2025) present a comprehensive survey of approximately 160 studies on SLMs, focusing on models with parameter counts typically ranging from 1 to 8 billion. The authors categorize SLMs into general-purpose and task-specific models and systematically analyze the techniques used to improve their efficiency, such as knowledge distillation, parameter-efficient training, and low-precision optimization. The survey shows that several SLMs can achieve performance comparable to, or in some cases better than, larger language models across different tasks.

Javaheripi and Bubeck (2023) introduce Phi-2, a 2.7B-parameter Small Language Model that exhibits strong reasoning and language understanding capabilities despite its compact size. The authors show that Phi-2 achieves state-of-the-art performance among base models below 13B parameters and matches or outperforms significantly larger models, including Llama-2 and Mistral variants, on benchmarks covering commonsense reasoning, language understanding, math, and coding.

Bai et al. (2023) introduce Qwen, a family of large and small language models developed by Alibaba Cloud, designed to balance performance and efficiency across a wide range of tasks. The Qwen series spans multiple parameter scales, from compact models suitable for edge deployment to larger systems, demonstrating strong performance in multilingual understanding, reasoning, mathematics, and coding benchmarks. The authors emphasize that smaller variants can achieve competitive results through high-quality data curation, instruction tuning, and efficient training strategies. In parallel, (ZHANG et al., 2024) present TinyLlama, a 1.1B-parameter open-source model that leverages community-driven optimizations to outperform comparable models in downstream tasks while maintaining high computational efficiency. Similarly, Google's Gemma family of lightweight language models demonstrates that compact architectures can retain strong reasoning and evaluation capabilities, with studies showing that Gemma models achieve performance correlated with larger systems in analytical and research-oriented tasks (THELWALL, 2025). Together, these works reinforce the trend toward efficient language models

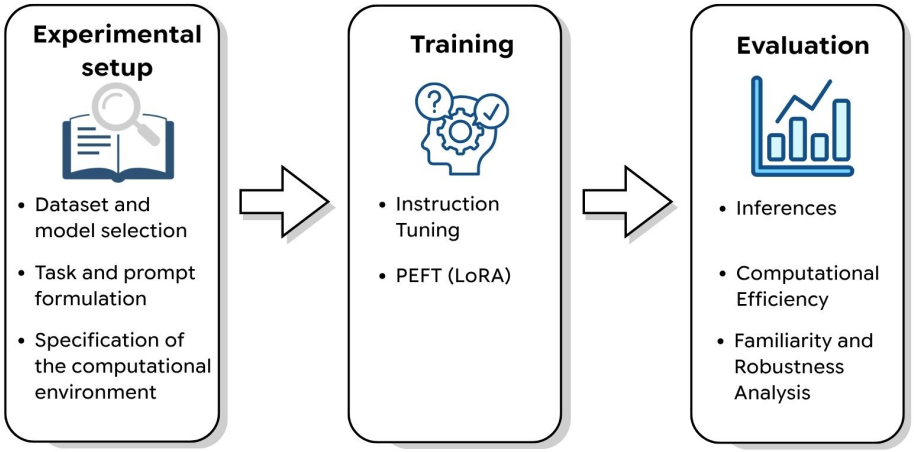
that reduce computational cost while preserving competitive performance.

Beyond architectural scaling, recent work has increasingly focused on training strategies designed to improve the reasoning capabilities of smaller language models. Magister et al. (2023) investigate whether reasoning behaviors observed in large language models can be transferred to smaller models through Chain-of-Thought (CoT) knowledge distillation. The authors fine-tune smaller T5 models using reasoning traces generated by large teacher models, such as PaLM 540B and GPT-3 175B, and report consistent performance gains across arithmetic, commonsense, and symbolic reasoning benchmarks. Hsieh et al. (2023) propose distilling step-by-step, a training strategy that leverages intermediate reasoning rationales generated by large language models as additional supervision. By jointly training smaller models to predict both final answers and step-by-step rationales, the authors show that substantially smaller models can outperform large language models while requiring significantly less training data.

4 METHODOLOGY

The experimental pipeline comprises three main stages. First, the experimental setup includes dataset and model selection, task and prompt formulation, and specification of the computational environment. Second, the selected SLMs are adapted to NLI through supervised instruction tuning using LoRA. Finally, the fine-tuned models are evaluated on a held-out test set and compared with their zero-shot counterparts in terms of predictive performance, computational efficiency, potential benchmark familiarity, and robustness under controlled conditions.

Figure 4.1: Overview of the methodology adopted in this study.



Source: The Author.

4.1 Dataset Selection

Since this work focuses on analyzing language models for the NLI task, the initial experimental design used three benchmark datasets: SNLI, MultiNLI, and MSCINLI. The objective was to evaluate model behavior under progressively more challenging and diverse inference scenarios.

The SNLI (BOWMAN et al., 2015) dataset was selected because it is one of the most traditional and widely adopted benchmarks for NLI research. It contains over 570,000 human-annotated premise–hypothesis pairs labeled as entailment, contradiction, or neutral, and is extensively used to evaluate semantic reasoning and language under-

standing capabilities.

The inclusion of MultiNLI (WILLIAMS; NANGIA; BOWMAN, 2018) was motivated by its broader diversity of text genres. Unlike SNLI, which is mainly composed of image-caption-derived sentences, MultiNLI incorporates texts from multiple domains such as fiction, government reports, spoken conversations, and travel guides. This characteristic introduces greater linguistic variability and increases the complexity of the inference task.

Additionally, MSCINLI (SADAT; CARAGEA, 2024) was considered in order to extend the evaluation to a more scientific context. The dataset contains inference examples extracted from academic and scientific literature, requiring models to deal with more technical vocabulary and domain-specific reasoning patterns.

Conducting complete fine-tuning, efficiency, benchmark familiarity, and robustness experiments across all three datasets would substantially increase the computational cost and experimental scope of the study. Therefore, the final experiments were restricted to a part of SNLI. This decision enabled a more detailed and controlled investigation of the selected models while retaining a benchmark that is well established in the NLI literature. Consequently, the conclusions of this study are limited primarily to the linguistic characteristics and domain represented by the sample used of SNLI.

Due to the large scale of the original SNLI dataset, using the full dataset for all experiments would substantially increase both computational cost and execution time, particularly because the complete experimental pipeline had to be repeated for five different SLMs, multiplying the required computational resources and execution time. Therefore, a reduced subset containing 17,000 instances was created and used on all experiments.

Because the experiments were conducted on a reduced subset, the sampling procedure prioritized premise-hypothesis pairs whose lengths were close to the average lengths observed in the original corpus. This criterion was adopted to focus the subset on instances with typical textual characteristics, reducing the influence of unusually short or long examples that could introduce additional variability into comparisons between models. The objective was not to reproduce the complete sentence-length distribution of SNLI, but to create a computationally manageable and controlled subset centered on representative-length instances. Before sampling, each premise-hypothesis pair received a score based on the distance between its sentence lengths and the target average lengths of the original corpus. Samples with scores closer to the dataset averages were prioritized during selection.

In addition, the sampling script enforced balanced label quotas across the three NLI classes (entailment, contradiction, and neutral), ensuring that each split maintained nearly identical class distributions. A fixed random seed was also used to ensure the reproducibility of the generated subset.

To illustrate the structure of an NLI instance contained in the generated subset, an example is presented below:

"Premise: `An elderly woman carrying plants on a long bridge over the river.`
Hypothesis: `A young woman is carrying plants.`
Label: `Contradiction"`

After the subset generation process, additional validation procedures were performed to ensure dataset integrity and consistency. Integrity checks were applied to guarantee the absence of null or missing values across all instances. Duplicate detection procedures were also performed to ensure that no repeated premise–hypothesis pairs existed either within or across the training, validation, and test splits, preventing potential data leakage between evaluation stages.

The final subset was divided into 10,000 training instances, 2,000 validation instances, and 5,000 test instances. Table 4.1 presents the resulting distribution of instances and labels across each split.

Table 4.1: Total number of instances and proportional label distribution across the training, validation, and test splits of the SNLI subset

Split	*Instances*	*Entailment (%)*	*Contradiction (%)*	*Neutral (%)*
Train	10,000	33.33	33.33	33.34
Validation	2,000	33.35	33.30	33.35
Test	5,000	33.34	33.33	33.33
Total	17,000	33.33	33.33	33.34

Source: The Author

4.2 Model Selection

This study evaluated six language models: five SLMs and one LLM used as a baseline. The selection aims to provide a comparative analysis of predictive performance and computational efficiency on the NLI task. To ensure a consistent baseline for compar-

ison, all selected models are instruction-tuned and based on the decoder-only transformer architecture. However, it is important to note that while the SLMs rely on dense parameter configurations, the baseline LLM features a Mixture-of-Experts (MoE) architecture, introducing a sparse activation mechanism into the comparative framework.

4.2.1 Small Language Models

Although there is no universally accepted definition of what constitutes an SLM in the literature, different studies adopt distinct parameter ranges, architectural constraints, and evaluation criteria when categorizing compact language models.

For the purposes of this study, SLMs were defined as decoder-only transformer models containing up to 7 billion parameters. This threshold was primarily determined by the computational resources available for fine-tuning and evaluation, as well as by the costs associated with accessing larger-scale training and inference infrastructure. Models within this range could be trained and evaluated under the project's resource constraints while remaining sufficiently capable to support meaningful comparisons on NLI tasks.

The decision to exclude encoder-based architectures was motivated by the characteristics of the experimental setup scope of this work. Encoder models such as BERT (DEVLIN et al., 2019) and RoBERTa (LIU et al., 2019) were designed as bidirectional representation models and are commonly fine-tuned for discriminative language understanding tasks. They have historically achieved strong performance on sentence-pair classification benchmarks such as NLI (GUBELMANN et al., 2024). When the premise and hypothesis are concatenated into a single input sequence, bidirectional self-attention enables direct interaction between their token representations, making these architectures particularly well suited to classification-oriented settings.

In contrast, decoder-only models are primarily trained through autoregressive next-token prediction and can formulate NLI as a prompted label-generation task. Since this work aims to compare compact and larger generative language models under a common prompting framework, the evaluation was restricted to decoder-only architectures. This restriction avoids combining models trained under substantially different objectives and inference protocols, thereby improving the consistency of the comparisons.

Based on the selection criteria described, five SLMs were chosen for the experimental evaluation. The selected models vary in parameter count while sharing the same decoder-only architecture and instruction-tuned training paradigm, ensuring a uniform ba-

sis for comparison. Table 4.2 summarizes the main characteristics of the selected SLMs.

Table 4.2: Parameter sizes and primary references of the five instruction-tuned Small Language Models selected for the experimental evaluation

Model	*Parameters*	*Reference*
Mistral-7B-Instruct-v0.3	7B	(JIANG et al., 2023)
OpenChat-3.5-0106	7B	(WANG et al., 2024b)
Phi-3-mini-4k-instruct	3.8B	(ABDIN et al., 2024)
Qwen2.5-7B-Instruct	7B	(YANG et al., 2025)
Qwen2.5-3B-Instruct	3B	(YANG et al., 2025)

Source: The Author

Mistral-7B-Instruct-v0.3 (JIANG et al., 2023) is an instruction-tuned version of the Mistral 7B architecture designed to improve reasoning and instruction-following capabilities while maintaining computational efficiency. The model introduces an updated tokenizer with an expanded vocabulary and improved multilingual support.

OpenChat-3.5-0106 (WANG et al., 2024b) is based on the Mistral-7B architecture and fine-tuned using the Conditioned Reinforcement Learning from Fine-Tuning (C-RLFT) framework, which enables the model to learn from datasets with varying quality levels.

Phi-3-mini-4k-instruct (ABDIN et al., 2024) is a compact 3.8B-parameter decoder-only transformer trained on approximately 4.9 trillion tokens, emphasizing high-quality and reasoning-oriented synthetic data designed to improve logical reasoning capabilities.

Qwen2.5-3B-Instruct and **Qwen2.5-7B-Instruct** (YANG et al., 2025) are part of Alibaba Cloud's Qwen2.5 family of language models, pretrained on large-scale multilingual corpora and later aligned through supervised fine-tuning and preference optimization procedures.

4.2.2 Large Language Model

To provide a large-scale reference for comparison, this work adopted GPT-OSS-120B (AGARWAL et al., 2025) as the LLM baseline. GPT-OSS-120B is based on a Mixture-of-Experts architecture (FEDUS; ZOPH; SHAZEER, 2022), in which only a subset of the total parameters is activated for each input token. Although the model contains approximately 120 billion parameters in total, only a fraction of them (approximately 5.1 billion per token, depending on routing) are used during inference. This design allows the model to maintain a very large parameter capacity while reducing the effective com-

putational cost per token compared to dense models of similar scale.

All inference requests involving the LLM baseline were executed through the GroqCloud infrastructure (Groq, 2026) using an external Application Programming Interface (API). It should be noted that the model was evaluated as a black-box system, since no direct access was available to the underlying model weights, deployment configuration, quantization settings, or hardware infrastructure. Although the service identified the deployed model as GPT-OSS-120B, the experimental evaluation was necessarily limited to the observable input-output behavior exposed by the API. As a result, infrastructure-dependent characteristics could not be independently verified, and only externally observable metrics such as response outputs and inference latency could be measured directly.

Because the execution environment and underlying hardware infrastructure were not directly accessible, precise hardware-level energy measurements for the large model could not be obtained. The energy consumption and carbon emissions associated with the LLM baseline were therefore estimated indirectly from the measured inference execution times observed during the experiments. To approximate the underlying hardware consumption, this work adopted the power characteristics of NVIDIA H100 GPUs (NVIDIA, 2024), since GPUs in this class are commonly used for serving large-scale language models with hundreds of billions of parameters and provide sufficient memory and computational capacity for deploying larger models. The estimation procedure further considered the reported deployment requirements of GPT-OSS-120B together with publicly available NVIDIA H100 power consumption specifications. It is important to highlight that GPT-OSS-120B is based on a Mixture-of-Experts architecture, in which inference compute does not scale linearly with the total number of parameters. Consequently, although hardware-level estimations were based on large-scale GPU deployments, the effective compute footprint per inference may be substantially lower than that of a dense 120B parameter model. Therefore, the reported environmental impact estimates should be interpreted as approximations for a hypothetical H100 scenario rather than as measurements of the actual GroqCloud infrastructure.

4.3 Experimental Environment

All experiments conducted throughout this study were executed using Jupyter Notebook environments hosted on Google Colab. The cloud-based infrastructure was adopted to simplify environment setup, dependency management, and access to com-

putational resources that exceeded those available on local hardware. Specifically, the experiments were performed on instances equipped with NVIDIA L4 GPUs, providing approximately 22.5 GB of dedicated GPU memory and 53.0 GB of system RAM. This hardware configuration was sufficient to support both inference and LoRA-based supervised instruction tuning for all selected Small Language Models.

Although the primary experiments were conducted on Google Colab, some auxiliary preprocessing and analysis scripts were executed locally. These included scripts responsible for downloading the original datasets, generating the reduced SNLI subset, validating the generated splits, and producing graphical visualizations of the obtained results. Since these preprocessing and visualization tasks did not require GPU acceleration or high-performance computational resources, executing them locally reduced unnecessary cloud resource utilization.

4.4 Supervised Instruction Tuning with LoRA

Supervised instruction tuning was also investigated as a strategy to improve the performance of SLMs on the NLI task. Since full fine-tuning of modern transformer models requires substantial computational resources and memory consumption, the experiments adopted a Parameter-Efficient Fine-Tuning (PEFT) strategy based on Low-Rank Adaptation (LoRA) (HU et al., 2021). LoRA reduces the number of trainable parameters by freezing the original base instruction-tuned model weights and introducing trainable low-rank decomposition matrices into selected linear transformation layers. Instead of updating the full parameter set of the model, the adaptation process only learns small low-rank matrices that approximate weight updates during training. This approach substantially decreases GPU memory requirements and training costs while still enabling effective task adaptation.

The supervised instruction tuning experiments were conducted using the Hugging Face Transformers ecosystem together with the PEFT and TRL libraries. All experiments adopted a fixed random seed in order to improve reproducibility across runs. Each training instance was converted into a structured prompt containing an instruction, the premise, the hypothesis, and the expected textual response corresponding to one of the three NLI classes: *entailment*, *neutral*, or *contradiction*. The dataset was structured into prompts and completions, and prompt tokens were ignored in the loss calculation.

The LoRA configuration used a rank $r = 16$, scaling factor $\alpha = 32$, and dropout

rate of 0.05. Only the adapter parameters were updated during training, while the original model weights remained frozen. Training was performed for up to five epochs using a per-device batch size of 1 and gradient accumulation over 16 steps, resulting in an effective batch size of 16 samples. Gradient checkpointing was enabled to reduce memory consumption, and mixed-precision training was performed using the bfloat16 (BF16) format. The models were loaded in BF16 precision, and no additional quantization technique was applied.

Optimization employed the AdamW optimizer with a learning rate of 1×10^{-4}, weight decay of 0.01, a cosine learning-rate scheduler, and a warmup ratio of 0.1. Model performance was evaluated every 200 training steps using the validation set, and checkpoints were saved at the same frequency. Early stopping was applied with a patience of three consecutive evaluations, monitoring the validation loss (*eval_loss*) as the optimization criterion. The checkpoint achieving the lowest validation loss was automatically selected and restored at the end of training.

This supervised instruction tuning procedure was applied independently to all five selected language models considered in the experimental setup. After training, inference experiments were conducted with each LoRA-adapted model under the same evaluation conditions used for the original base instruction-tuned models. This design enabled direct comparisons between pretrained and fine-tuned variants across all evaluated architectures.

4.5 Inferences

All inference experiments were conducted using the test split of the generated SNLI subset, containing 5,000 instances. During inference, generation was constrained to short responses corresponding to the target NLI labels. The generated outputs were normalized and mapped to the three valid classes: *entailment*, *neutral*, and *contradiction*.

The experiments evaluated both the original instruction-tuned Small Language Models and their corresponding LoRA-adapted versions on the selected SNLI subset. To maintain consistency across all evaluation settings, the same instruction template adopted during supervised instruction tuning was also used during inference. Furthermore, all models were evaluated using identical decoding configurations, including the same temperature, sampling strategy and maximum output length, ensuring that performance differences could be attributed to the models themselves rather than to variations in inference settings.

Each selected SLM was evaluated in two different states: its original instruction-tuned version and its LoRA-adapted version obtained through supervised instruction tuning. For both model versions, two prompting strategies were investigated: zero-shot prompting (see Section 4.5.1) and few-shot prompting (Section 4.5.2). This experimental design allowed the analysis of both the impact of supervised adaptation and the influence of providing demonstration examples during inference.

Unlike the SLMs, the LLM was exclusively evaluated using a zero-shot prompting strategy. The implementation of few-shot experiments was precluded by strict time constraints coupled with API rate limits on the Groq platform. During the experimental phase, paid developer plans were temporarily unavailable, necessitating the use of a free-tier account. Consequently, to process the entire test subset under these restrictive quotas, the zero-shot inferences had to be partitioned and executed in batches over several days, with the intermediate outputs iteratively saved and aggregated.

Before settling on the final inference prompt, we experimentally evaluated several alternative formulations. Testing more elaborate prompts, which included detailed instructions, task descriptions, or strict formatting requirements, frequently triggered undesired behaviors across the selected models. These issues ranged from models simply completing the hypothesis text to generating labels outside the valid NLI classes.

In contrast, shorter and more direct prompts consistently yielded the expected behavior across all models. Consequently, we adopted a concise prompt structure that explicitly defines the classification task and the required response format.

4.5.1 Zero-Shot Prompting

Under the zero-shot configuration, the models received only the instruction together with the target premise–hypothesis pair and were required to directly generate the predicted label without any additional demonstration examples.

The prompt template adopted in this setting is shown below.

```
"Instruction: Given the premise and hypothesis, choose one:
entailment, neutral, or contradiction.
Premise: [premise sentence]
Hypothesis: [hypothesis sentence]
Answer: "
```

4.5.2 Few-Shot Prompting

The few-shot configuration provided demonstration examples together with the target premise–hypothesis pair. Two different few-shot strategies were evaluated.

In the first strategy, referred to as *fixed few-shot prompting*, the same set of five manually selected NLI examples was included in all prompts during inference. These examples were selected from the training split and remained constant throughout the entire evaluation procedure for all models.

The second strategy adopted a *random few-shot prompting* approach. A dedicated pool containing 50 examples sampled from the training split was first established, ensuring balanced representation across all three classes. During inference, five examples were randomly selected from this 50-example pool for each incoming prediction instance. Consequently, the contextual demonstrations varied dynamically throughout the evaluation process. This strategy was designed to reduce potential biases associated with a single fixed demonstration set

The general prompt structure used in the few-shot experiments is illustrated below.

```
"Instruction: Given the premise and hypothesis, choose one:
entailment, neutral, or contradiction.
Examples: [five examples]
Premise: [premise sentence]
Hypothesis: [hypothesis sentence]
Answer: "
```

During each inference experiment, a set of predictive, computational, and environmental metrics was collected. Predictive performance was assessed using accuracy and macro F1-score, whereas computational performance was evaluated through total inference latency and average latency per prediction instance. Environmental impact was quantified through estimated energy consumption and carbon emissions.

4.6 Computational Efficiency Measurements

To quantitatively evaluate the environmental and computational trade-offs of using compact language models, tracking hardware-level resource consumption during training and inference is critical. In this study, the environmental impact and power metrics

were estimated using the CodeCarbon library (COURTY et al., 2024). CodeCarbon is an open-source Python package that tracks the electricity consumption of computational infrastructure and estimates the associated carbon footprint based on the carbon intensity of the local regional grid.

For each experimental run involving the SLMs, an automatic monitoring instance was initialized using the EmissionsTracker class. The tracker was configured as follows:

```
tracker = EmissionsTracker(
    project_name=f"{dataset_name}_{model_name}",
    output_dir=str(log_path),
    output_file=emissions_csv.name,
    log_level="error",
)
```

The tracking process was initiated using the tracker.start() command immediately prior to execution and stopped upon completion of the respective task. This setup captured fine-grained hardware metrics, logging them into a structured comma-separated values CSV file. The tracked variables included execution duration, total energy consumed, estimated carbon emissions (measured in kilograms of CO_2 equivalent, or $kgCO_2e$), and power draw breakdown for individual hardware components.

Specifically, the library independently queried the underlying system architectures estimated power draw of the Central Processing Unit (CPU), Graphics Processing Unit (GPU), and Random Access Memory (RAM). These power metrics (measured in Watts) were integrated over the execution duration to compute the total energy consumed by each component in kilowatt-hours (kWh).

To ensure complete transparency regarding the experimental setup, Table 4.3 presents the specific telemetry schema logged by the tracker during a representative few-shot inference run on the SNLI subset using the LoRA-adapted Mistral-7B model.

By utilizing these hardware telemetry logs, estimated energy consumption and carbon emissions can be compared across different models and inference configurations. These measurements are intended to support relative efficiency analyses rather than provide exact physical estimates.

Table 4.3: Example of the hardware and environmental telemetry schema captured by CodeCarbon during a representative inference execution, detailing the specific metrics logged for power consumption, estimated carbon emissions, and system utilization

Metric Group	**Captured Experimental Values**
Project & Run IDs	snli_sample_17k_mistral7b-trained
Execution Duration	3373.38 seconds ($\approx$ 56.2 minutes)
Component Power Draw	CPU: 42.5 W | GPU: 71.68 W | RAM: 20.0 W
Energy Consumed	CPU: 0.0398 kWh | GPU: 0.0671 kWh | RAM: 0.0187 kWh
Total Energy Consumed	0.1257 kWh
Total CO_2 Emissions	0.0591 kgCO_2e
Hardware Specifications	1 $\times$ NVIDIA L4 GPU | 12 $\times$ Intel(R) Xeon(R) CPU @ 2.20GHz
Memory Allocation	Total System RAM: 52.95 GB | Active System RAM Used: 5.29 GB
Average Utilization	CPU: 10.77% | GPU: 97.29% | RAM: 10.00%
Operational System	Linux-6.6.122

Source: The Author

4.7 Potential Benchmark Familiarity Analysis

A critical challenge when evaluating modern language models on public benchmarks is data contamination. Data contamination occurs when test instances or their underlying textual inputs are inadvertently included in a model's pretraining or alignment datasets. If a model has memorized part of the evaluation corpus, its predictive performance may be artificially inflated, reflecting data memorization rather than actual general-purpose language reasoning capabilities (FU et al., 2025).

To investigate potential benchmark familiarity among the evaluated decoder-only SLMs, an empirical prefix-completion test was designed and executed.

This methodology assesses whether a model can reproduce or closely approximate textual continuations when prompted with only the initial segment of a sentence, thereby indicating possible prior exposure to or memorization of the evaluated content.

4.7.1 Experimental Design and Prompt Engineering

The benchmark familiarity analysis focused on the underlying textual premises extracted from the sampled training split of the SNLI dataset. A set of randomly selected instances (N_{samples}) was isolated from the dataset. For each instance, the original text of the premise was split into two sequential segments based on a predefined ratio ($R_{\text{prefix}} = 0.5$): an initial string acting as the prefix, and the remaining ground-truth string acting as the target suffix.

The models were evaluated under a zero-shot, greedy-decoding generation configuration to enforce deterministic replication behavior and prevent stochastic variations. Each model was provided with a structurally isolated generation prompt containing strict stylistic guidelines designed to suppress generalized creative text generation and isolate rote memorization tendencies. The prompt structure utilized is defined below:

```
You are completing a caption.
Write a short, literal continuation that fits the sentence.
Do not introduce new actions or story.
Stay consistent with the scene.
Do not repeat words unnecessarily.
Return ONLY the continuation.

[prefix]
```

The text generated by the model (Y_{gen}) was restricted to a maximum length of 50 tokens, stripped of trailing special characters, and truncated at the first newline token boundary to align directly with the target suffix format.

4.7.2 Evaluation Metrics

To measure the degree of text replication and verify if the generated completions matched the original ground-truth suffixes, three distinct linguistic alignment metrics were computed for each instance:

1. **Semantic Similarity:** Captures whether the generated completion shares the underlying meaning of the true suffix, independent of exact wording. This was computed using cosine similarity over dense vector representations generated by a frozen text embedding model (all-MiniLM-L6-v2), as formulated in Equation 4.1:

$$\text{Semantic Similarity}(Y_{\text{gen}}, \text{suffix}) = \frac{\mathbf{e}_{\text{gen}} \cdot \mathbf{e}_{\text{suffix}}}{\|\mathbf{e}_{\text{gen}}\| \|\mathbf{e}_{\text{suffix}}\|} \tag{4.1}$$

   where $\mathbf{e}_{\text{gen}}$ and $\mathbf{e}_{\text{suffix}}$ denote the high-dimensional embeddings of the generated output and the true suffix, respectively.

2. **Lexical Overlap:** Quantifies the proportion of unique words in the generated continuation that also appear in the ground-truth suffix, ignoring casing and token po-

sitioning:

$$\text{Lexical Overlap}(Y_{\text{gen}}, \text{suffix}) = \frac{|S_{\text{gen}} \cap S_{\text{suffix}}|}{|S_{\text{gen}}|} \tag{4.2}$$

where S_{gen} and S_{suffix} represent the lowercase word sets of the generated continuation and the ground-truth suffix, respectively.

3. **Normalized Levenshtein Distance:** Evaluates exact character-level alignment by calculating the minimum number of single-character operations (insertions, deletions, or substitutions) required to transform the generated continuation into the ground-truth target. The raw edit distance (Lev) was calculated using a standard dynamic programming matrix and normalized against the maximum string length to derive a similarity scale bounded between 0 and 1:

$$\text{Normalized Levenshtein}(Y_{\text{gen}}, \text{suffix}) = 1 - \frac{\text{Lev}(Y_{\text{gen}}, \text{suffix})}{\max(|Y_{\text{gen}}|, |\text{suffix}|)} \tag{4.3}$$

   A value approaching 1.0 signifies identical sequence replication.

The individual metrics collected across all evaluated instances were aggregated incrementally and saved into a structured format to calculate the global mean values (μ) for each language model. These results provide exploratory evidence of possible familiarity with SNLI textual content and potential memorization effects. However, they do not establish a causal relationship between such familiarity and the observed NLI performance.

4.8 Robustness Analysis

To evaluate whether model performance was sensitive to specific lexical patterns, a robustness analysis was conducted using synonym-based lexical perturbations applied to the test split. The objective was to assess the stability of the evaluated SLMs under input modifications designed to preserve the original semantic content.

The perturbation process relied on spaCy for linguistic analysis, WordNet (MILLER, 1995) for synonym retrieval, and LemmInflect for morphological adaptation. Each sentence was analyzed to identify eligible tokens while preserving named entities and other elements that could alter the factual context. Candidate synonyms were obtained from WordNet and filtered to retain only single-word lexical alternatives. The selected synonym was then inflected to match the grammatical properties of the original token, and

the modified sentence was reconstructed while preserving its original formatting.

To ensure lexical consistency within each instance, a controlled substitution mapping was implemented across both the premise and the hypothesis. Specifically, if a particular noun appeared multiple times within a given sentence or across the premise-hypothesis pair, it was systematically replaced by the exact same synonym in all of its occurrences. This constraint was necessary to prevent the introduction of artificial semantic mismatches that could unintentionally alter the logical relationship being evaluated.

The initial experimental design considered noun substitution, verb substitution, and a combination of both. However, manual inspection revealed that verb substitutions frequently introduced subtle semantic shifts capable of changing the original entailment relationship and, consequently, corrupting the ground-truth labels. In contrast, noun substitutions generally preserved the original semantic content while altering only surface lexical forms.

Therefore, the final robustness evaluation was restricted to noun substitutions, which reduced the likelihood of altering the logical relationships between premises and hypotheses while still enabling the assessment of robustness to lexical variation.

An example of the applied perturbation is presented below.

```
[ORIGINAL]
Premise   : Two men sharing a bicycle,
            carrying several buckets and containers.
Hypothesis: The bike is one being ridden by one person.
Label     : contradiction

[MODIFIED]
Premise   : Two humans sharing a bike,
            carrying several pails and containers.
Hypothesis: The bicycle is one being ridden by one individual.
Label     : contradiction
```

The modified version preserves the original semantic relationship and label while introducing lexical variations, allowing the evaluation of model sensitivity to surface-level wording changes.

5 EXPERIMENTS AND RESULTS

This chapter details the empirical results obtained from the experimental evaluation of both small and large language models on the NLI task. The analysis is structured to compare predictive performance metrics alongside computational and environmental efficiency vectors, establishing a comprehensive overview of the trade-offs involved across different architectural scales.

5.1 LLM Baseline

The evaluation of the GPT-OSS-120B was conducted under a zero-shot prompting configuration with the temperature set to zero. The model was tasked with classifying the complete test split of the generated SNLI subset, totaling 5,000 distinct premise–hypothesis instances.

During this evaluation process, an operational anomaly was observed: the model failed to return a valid prediction for 58 out of the 5,000 instances. A subsequent analysis of these specific cases revealed that the API simply returned an empty string. Despite this observation, the definitive root cause remains uncertain; it was not possible to assert whether these empty responses were the result of underlying infrastructure anomalies, intermittent API dropouts, or if the model itself failed to comprehend the prompt instructions and defaulted to a blank output. However, because these missing outputs represent a negligible fraction of the overall test set, their exclusion does not significantly alter the aggregate performance metrics or the statistical validity of the evaluation.

To illustrate this behavior, the following are two examples of instances where the model returned an empty response:

Instance index: 1414
Premise: A group of people running a bicycle race past a red building.
Hypothesis: Three people run past a red building.
Correct Label: neutral
Model Response: Empty string

Index: 15
Premise: Women with red-hair and white and black shirt with

an O face.

Hypothesis: Woman with red hair and black shirt.

Correct Label: entailment

Model Response: Empty string

Table 5.1 presents the observed execution metrics alongside the estimated energy consumption measured in kilowatt-hours (kWh). These telemetry estimates were mathematically modeled by anchoring the recorded execution windows to the nominal power draw characteristics of an enterprise NVIDIA H100 GPU architecture. It is important to note that the reported energy consumption values are estimates rather than direct measurements. Due to the sparse activation mechanism of GPT-OSS-120B, the adopted estimation methodology likely overestimates the actual energy consumed during inference. Therefore, these values should be interpreted as approximate indicators for comparative analysis rather than absolute measurements of energy usage.

Table 5.1: Summary of predictive performance, operational execution metrics, and estimated hardware energy consumption for the GPT-OSS-120B baseline evaluated under a zero-shot configuration

Metric Group	**Captured Experimental Values**
Model Name	openai/gpt-oss-120b
Evaluation Samples (N)	5,000
Predictive Accuracy	0.7688
Macro F_1-Score	0.7674
Execution Duration	3795.485 seconds ($\approx$ 63.3 minutes)
Inference Latency (Mean)	0.7591 seconds/sample
Estimated GPU Energy Consumed	0.066235 kWh
Hardware Baseline Profile	NVIDIA H100 GPU
Infrastructure Context	GroqCloud API

Source: The Author

These results establish both the predictive and computational baselines for the comparative analyses presented in the subsequent sections.

5.2 Prompt Strategies

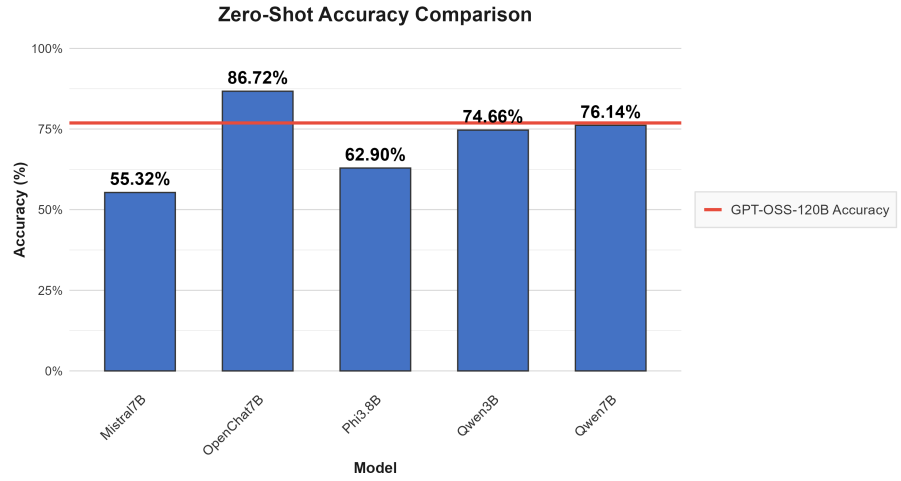
This section presents the empirical findings obtained from the evaluation of the five selected instruction-tuned SLMs. All evaluated models are general-purpose language models and were not specifically designed, specialized, or fine-tuned for NLI. The experiments were conducted under three prompting strategies: zero-shot prompting, fixed few-

shot prompting, and random few-shot prompting. The following subsections compare the predictive performance and environmental efficiency of each configuration against the GPT-OSS-120B zero-shot configuration. Since the evaluated prompting strategies differ only in prompt design, environmental efficiency remained approximately constant across the three prompting strategies for each model and is presented separately at the end of this section.

5.2.1 Zero-shot

Figures 5.1 and 5.2 summarize the predictive performance of the evaluated SLMs under zero-shot prompting. The former reports accuracy, whereas the latter presents the corresponding Macro F_1 Score values. In both figures, the red line indicates the threshold representing the performance obtained by the GPT-OSS-120B baseline under zero-shot prompting.

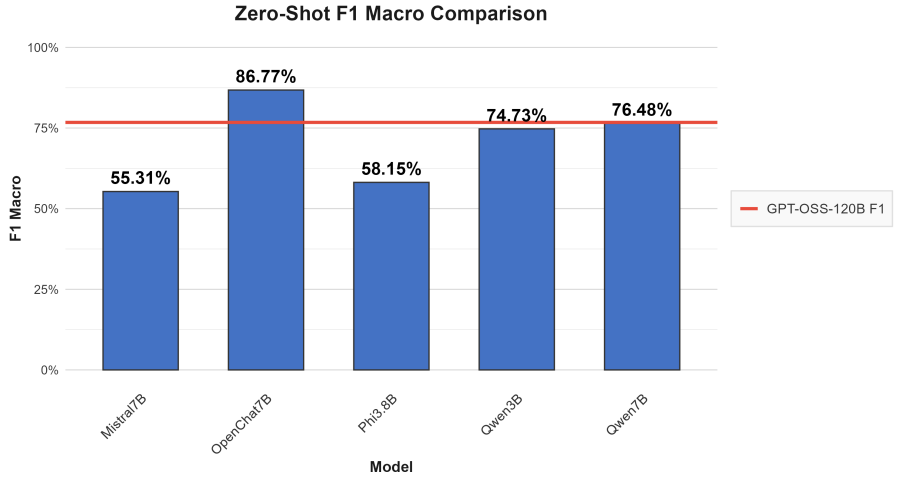
Figure 5.1: Zero-shot prompting accuracy across the evaluated SLMs



Source: The Author.

The zero-shot results reveal substantial variation in performance across the evaluated SLMs. OpenChat achieved the strongest overall performance, surpassing the GPT-OSS-120B baseline in both Accuracy and Macro F_1 score. The Qwen models also demonstrated competitive results, with Qwen2.5 7B closely matching the baseline despite its substantially smaller size. In contrast, Phi3-3.8B and Mistral 7B exhibited noticeably lower predictive performance. Although GPT-OSS-120B is a large-scale open-weight instruction-aligned model, its zero-shot performance did not consistently exceed that of

Figure 5.2: Zero-shot prompting F1 Macro across the evaluated SLMs



Source: The Author.

the smaller models. This result suggests that, for this specific NLI benchmark and for this instruction, model scale alone was not the dominant factor determining performance. Interestingly, the data indicates that the evaluated smaller models were more effective at adhering to the specific task instructions for the SNLI dataset compared to the larger GPT-OSS baseline.

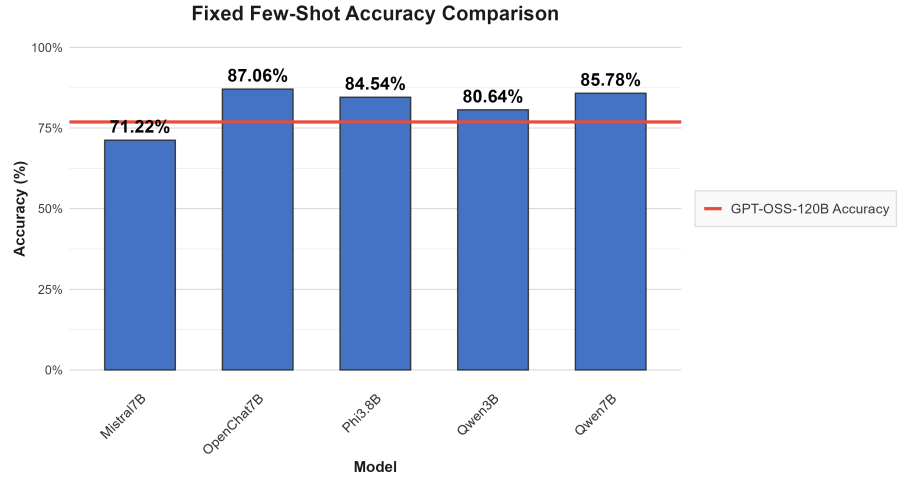
5.2.2 Fixed Few-shot

For the fixed few-shot evaluation, five distinct instances were manually selected from the training subset and embedded statically into the model's prompt alongside their respective gold labels.

Figures 5.3 and 5.4 illustrate the predictive performance achieved in this experiment, reporting accuracy and Macro F_1 score, respectively. The performance of the evaluated SLMs is compared against the zero-shot baseline of GPT-OSS-120B since, as detailed earlier, the larger model was not evaluated under few-shot conditions.

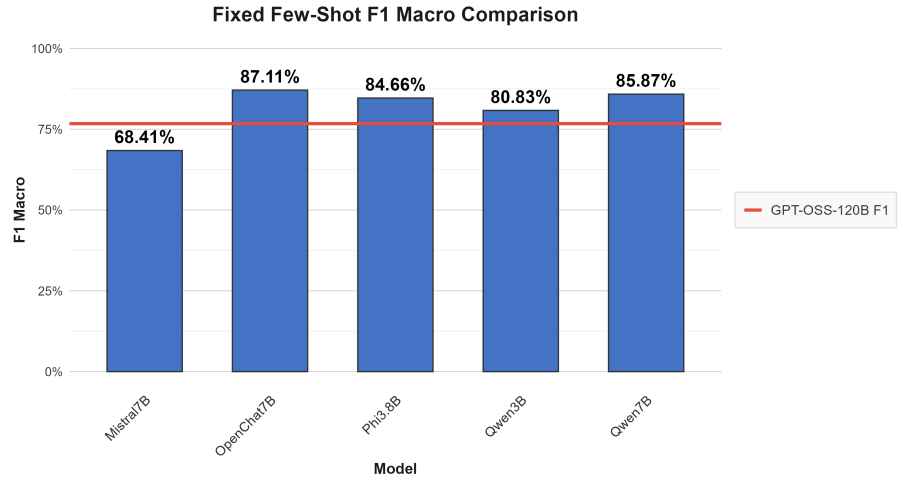
The fixed few-shot prompting strategy resulted in performance improvements across all evaluated SLMs, with every model achieving higher Accuracy and Macro F_1 score compared to its zero-shot counterpart. This suggests that the inclusion of representative examples provides useful task-specific guidance, enabling the models to better align their predictions with the natural language inference objective. Under this configuration, all evaluated SLMs except Mistral 7B surpassed the GPT-OSS-120B baseline. Although

Figure 5.3: Fixed few-shot prompting accuracy across the evaluated SLMs



Source: The Author.

Figure 5.4: Fixed few-shot prompting F1 Macro across the evaluated SLMs



Source: The Author.

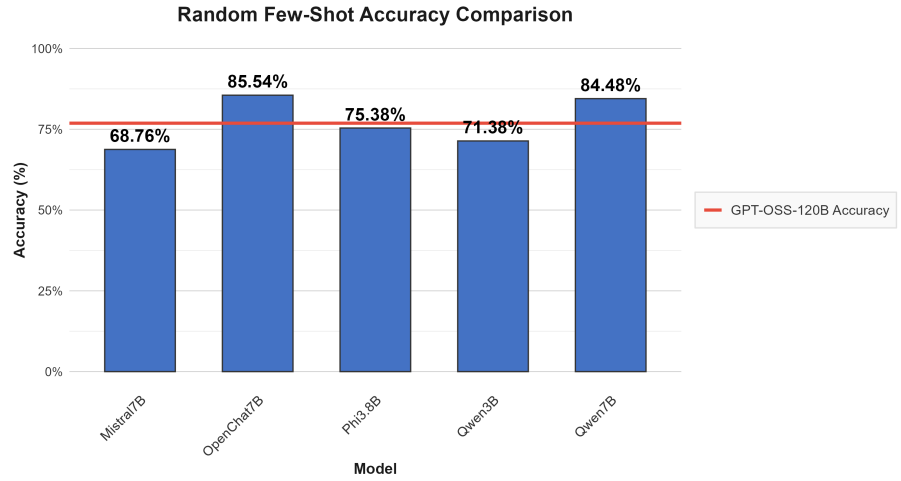
Mistral 7B also benefited from the additional examples, its performance remained below the baseline, indicating that further experimentation with alternative prompting strategies or model adaptation techniques may be required to achieve comparable results.

5.2.3 Random Few-shot

Under the random few-shot configuration, a pool of 50 distinct instances was initially selected from the training dataset. For each individual inference step, 5 instances were randomly sampled from this predefined pool and dynamically inserted into the model's prompt along with their corresponding gold labels.

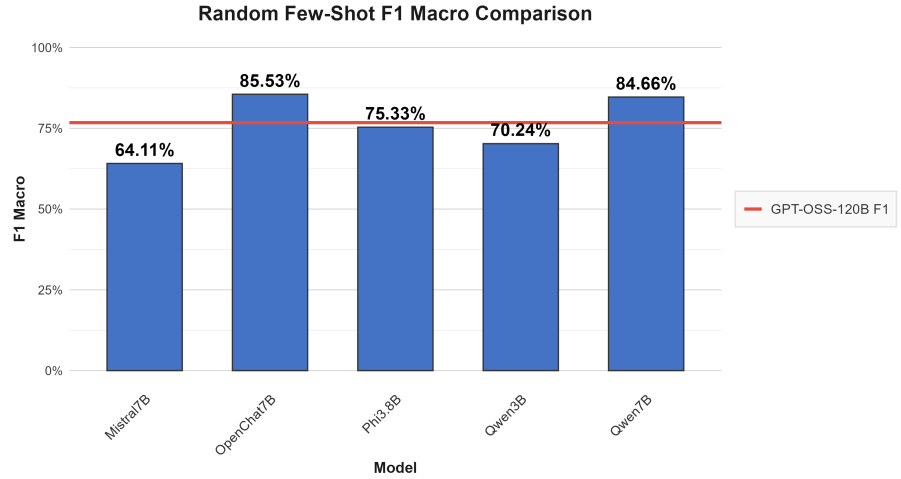
Figure 5.5 shows the accuracy for each model, as Figure 5.6 shows de F1 score obtained throught the experiment.

Figure 5.5: Random few-shot prompting accuracy across the evaluated SLMs



Source: The Author.

Figure 5.6: Random few-shot prompting F1 Macro across the evaluated SLMs



Source: The Author.

Compared to the fixed few-shot configuration, the random few-shot strategy produced consistently lower Accuracy and Macro F_1 score values across all evaluated models. However, it still outperformed the zero-shot setting in most cases, demonstrating that the inclusion of in-context examples can be beneficial even when selected at random. The observed performance gap likely stems from the variable quality and representativeness of the sampled demonstrations, as some of the selected instances may have provided limited guidance for the inference task.

Table 5.2 consolidates the predictive performance obtained across all evaluated SLMs and prompting configurations. For each model, the table reports the Accuracy and Macro F_1 score achieved.

Table 5.2: Comparison of predictive performance metrics achieved by the evaluated Small Language Models across zero-shot, fixed few-shot, and random few-shot prompting strategies

Model Configuration	**Accuracy**	**Macro F_1-Score**
Mistral 7B Zero-Shot	0.5532	0.5531
Mistral 7B Fixed Few-Shot	0.7122	0.6841
Mistral 7B Random Few-Shot	0.6876	0.6411
OpenChat 7B Zero-Shot	0.8672	0.8677
OpenChat 7B Fixed Few-Shot	0.8706	0.8711
OpenChat 7B Random Few-Shot	0.8554	0.8553
Phi-3 3.8B Zero-Shot	0.6290	0.5815
Phi-3 3.8B Fixed Few-Shot	0.8454	0.8466
Phi-3 3.8B Random Few-Shot	0.7538	0.7533
Qwen2.5 3B Zero-Shot	0.7466	0.7473
Qwen2.5 3B Fixed Few-Shot	0.8064	0.8083
Qwen2.5 3B Random Few-Shot	0.7138	0.7024
Qwen2.5 7B Zero-Shot	0.7614	0.7648
Qwen2.5 7B Fixed Few-Shot	0.8578	0.8587
Qwen2.5 7B Random Few-Shot	0.8448	0.8466

Source: The Author

5.2.4 Environmental Efficiency

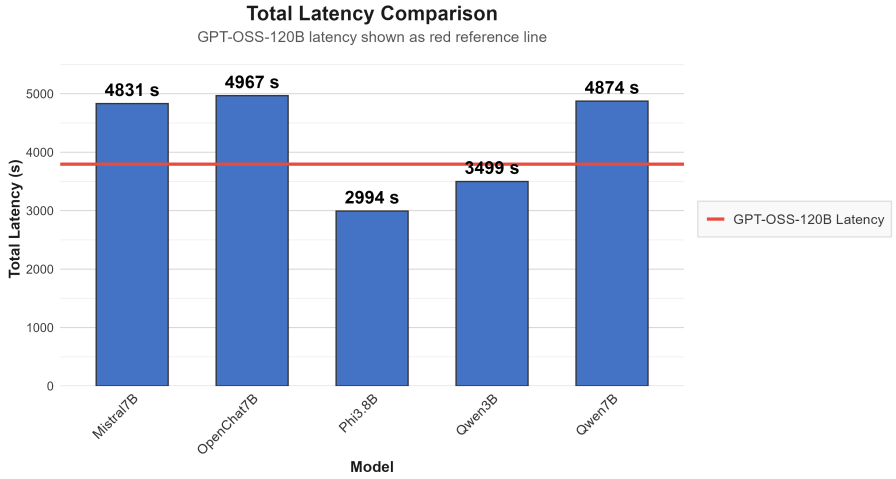
Figures 5.7 and 5.8 present the efficiency metrics of the evaluated models. Total inference latency is shown in the former, whereas estimated GPU energy consumption during the inference is reported in the latter.

As expected, inference latency and energy consumption exhibit a strong correlation, since the energy estimates were derived from the execution time under a fixed GPU power profile. Consequently, models requiring longer inference times also tend to present higher energy consumption.

Regarding execution efficiency, Phi3.8B and Qwen2.5 3B achieved lower total inference times than GPT-OSS-120B, resulting in correspondingly lower estimated energy consumption. In contrast, the remaining evaluated models required longer execution times and therefore consumed more energy than the GPT-OSS baseline.

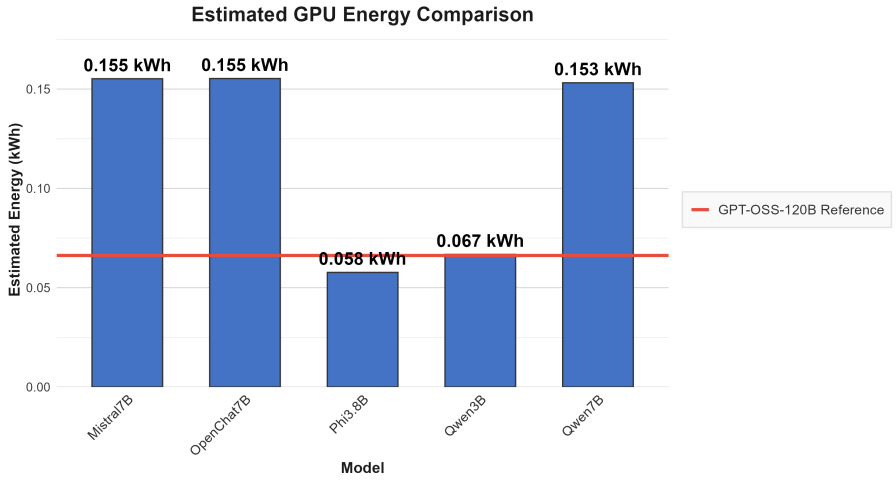
A plausible explanation for this efficiency may lie in the underlying architecture

Figure 5.7: Total latency across the evaluated SLMs



Source: The Author.

Figure 5.8: Total energy spent across the evaluated SLMs



Source: The Author.

of GPT-OSS-120B, which operates as a MoE model. Consequently, although GPT-OSS-120B has an exceptionally large total parameter count, at least 15 times larger than that of the largest evaluated SLM, the number of active parameters used during inference is substantially smaller due to its sparse activation mechanism. This sparse activation can significantly reduce the computational overhead per token, potentially allowing a massive MoE model to achieve end-to-end execution times that can rival or even outperform smaller, densely activated models. Therefore, it is highly possible that the observed efficiency metrics reflect this architectural distinction rather than a direct correlation with total model size.

Table 6.2 summarizes the environmental and operational efficiency metrics ob-

tained across all evaluated models. For each model, the table reports the total execution duration, estimated carbon emissions, CPU energy consumption, GPU energy consumption, RAM energy consumption, and total energy consumed during inference. These metrics provide a complementary perspective to predictive performance by quantifying the computational and environmental costs associated with each model.

Table 5.3: Environmental and operational efficiency metrics across the evaluated language models, detailing execution duration, component-level energy consumption and total estimated carbon emissions during inference

Model	Duration (s)	Emissions	CPU Energy	GPU Energy	RAM Energy	Energy Consumed
Mistral 7B	4836.88	0.0630	0.0571	0.0965	0.0269	0.1805
OpenChat 7B	4973.35	0.0874	0.0587	0.0993	0.0276	0.1856
Phi-3 3.8B	2998.73	0.0527	0.0354	0.0598	0.0167	0.1118
Qwen2.5 3B	3502.74	0.0455	0.0413	0.0696	0.0195	0.1304
Qwen2.5 7B	4878.43	0.0636	0.0576	0.0974	0.0271	0.1820
GPT-OSS-120B	–	–	–	0.0662	–	–

Source: The Author

Since GPT-OSS-120B was executed through an API-based deployment, access to infrastructure-level telemetry was not available. As a result, only GPU energy consumption could be estimated from the observed inference duration and the assumed NVIDIA H100 power profile.

5.3 Supervised Instruction Tuning

This section evaluates the impact of supervised instruction fine-tuning on the NLI capabilities of the selected Small Language Models. To this end, all five architectures were adapted using the custom SNLI subset described in Chapter 4, comprising 10,000 training instances and 2,000 validation instances.

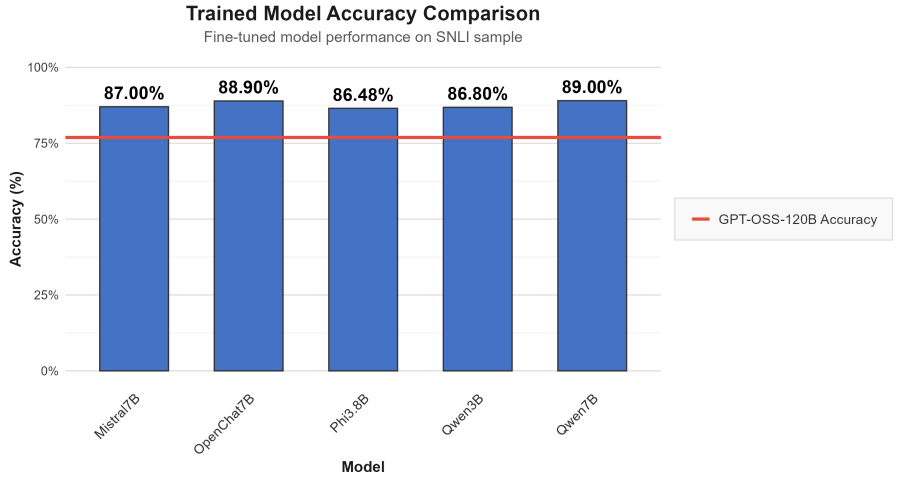
5.3.1 Zero-shot Strategy

After the training process, all fine-tuned models were evaluated using the same test split composed of 5,000 instances that was previously employed in the prompt engineering experiments.

Figures 5.9 and 5.10 shows the predictive performance obtained after supervised instruction tuning.

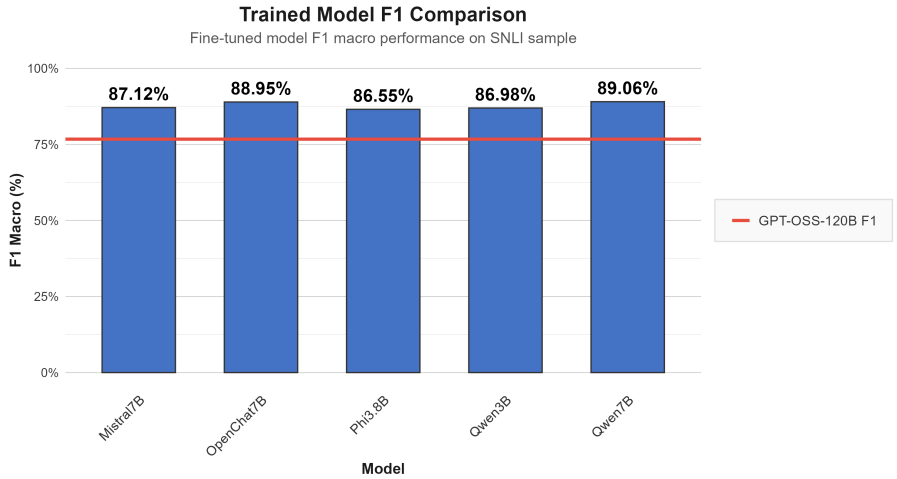
The results demonstrate that supervised instruction tuning produced substantial

Figure 5.9: Accuracy across all trained SLMs



Source: The Author.

Figure 5.10: F1 Macro across all trained SLMs



Source: The Author.

performance improvements across all evaluated SLMs. Every model achieved higher Accuracy and Macro F_1 score after training on the SNLI subset, and all fine-tuned models surpassed the GPT-OSS-120B baseline. The results reinforce the effectiveness of supervised fine-tuning as a practical approach for transforming compact language models into highly competitive task-oriented systems.

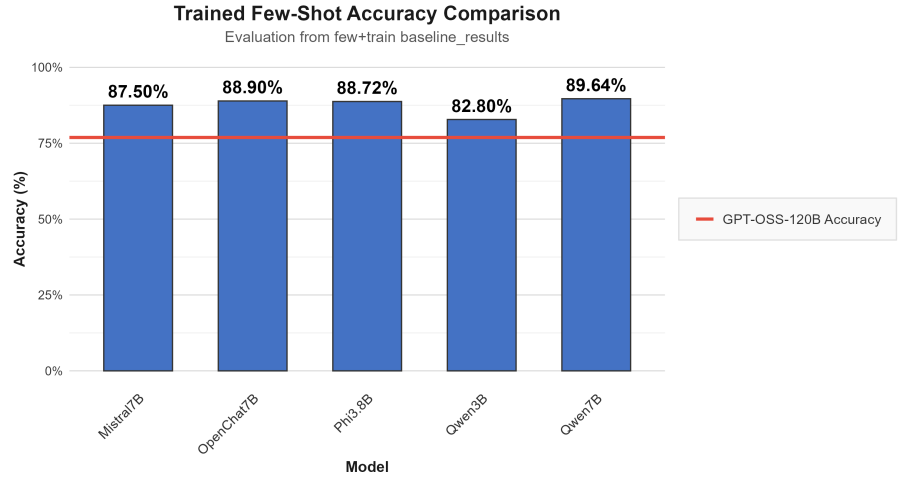
5.3.2 Few-shot Strategy

In addition to evaluating the fine-tuned models under a standard inference setting, an additional experiment was conducted using a fixed few-shot prompting strategy. In this

configuration, the prompt was augmented with five examples extracted from the training split, together with their corresponding gold labels. To ensure a consistent and fair comparison, these were the exact same five examples previously utilized during the few-shot evaluation of the original base models.

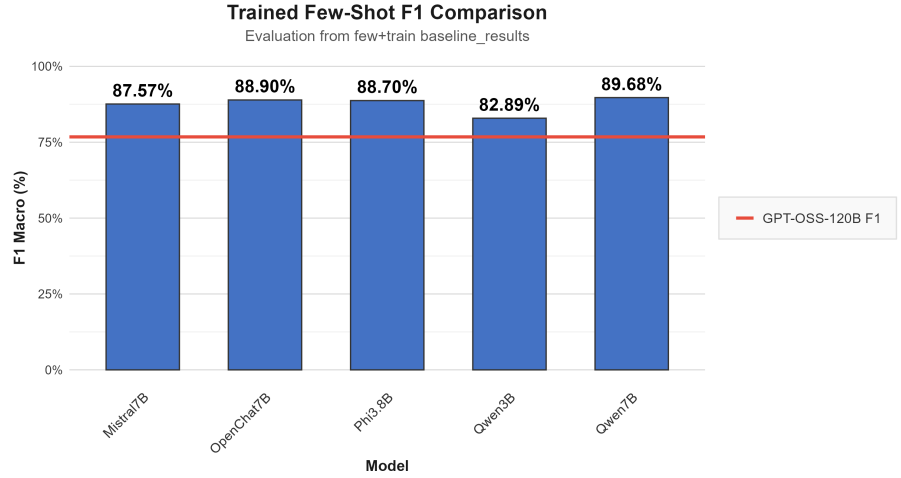
Although this approach introduces a degree of redundancy, since the demonstration examples had already been observed by the models during supervised instruction tuning, it was intentionally included to investigate whether providing explicit in-context examples at inference time could further improve predictive performance.

Figure 5.11: Accuracy across all trained SLMs using Few-shot prompt



Source: The Author.

Figure 5.12: F1 Macro across all trained SLMs using Few-shot prompt



Source: The Author.

The impact of introducing fixed few-shot demonstrations after supervised instruction tuning was not consistent across the evaluated models. While some architectures

exhibited modest improvements in Accuracy and Macro F_1-Score, others showed no changes or even performance degradation. This variability indicates that the effectiveness of combining supervised fine-tuning with additional in-context examples is highly model-dependent. A possible explanation is that certain architectures are able to leverage the demonstrations as complementary guidance, whereas others may become overly influenced by the specific patterns contained in the examples, reducing their ability to generalize to unseen instances. Consequently, the results suggest that the integration of supervised instruction tuning and few-shot prompting should not be regarded as universally beneficial. Instead, its effectiveness appears to depend on the underlying model architecture, as the same strategy can either improve performance, produce no measurable gains, or negatively affect generalization.

5.3.3 Confusion Matrices

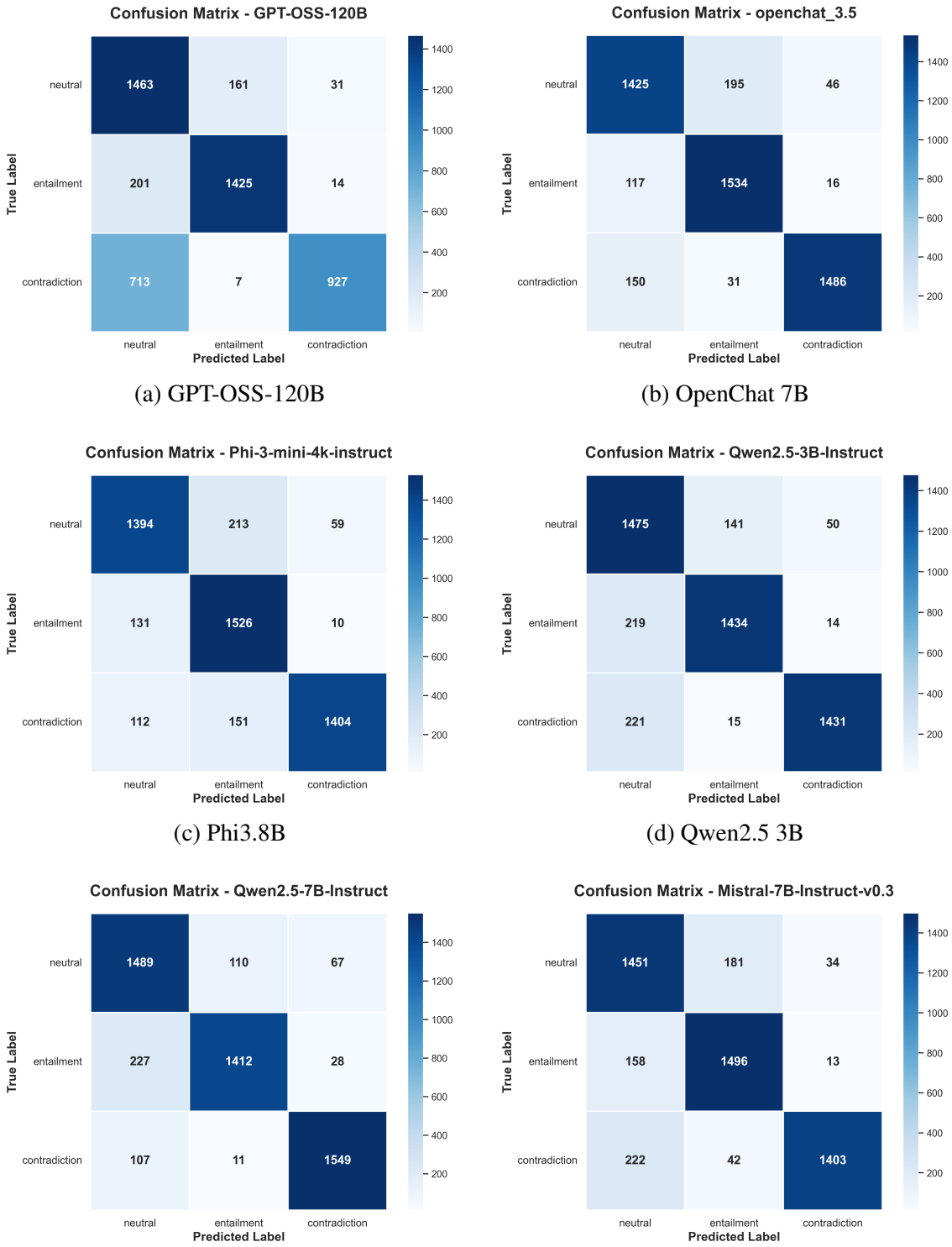
While aggregate metrics such as Accuracy and Macro F_1-Score provide an overall assessment of predictive performance, they do not detail how classification errors are distributed across the individual NLI classes. To obtain a more detailed understanding of model behavior, confusion matrices were generated for both each fine-tuned SLM and the LLM.

The confusion matrices enable the visualization of correct predictions and misclassification patterns among the three target classes. Through this analysis, it becomes possible to identify whether specific classes are more difficult to distinguish and to determine the most frequent error types produced by each model. This complementary evaluation provides additional insights into the reasoning behavior learned during supervised instruction tuning, allowing a more detailed interpretation of the performance improvements observed in the previous experiments.

The confusion matrices obtained for all evaluated fine-tuned models and for the LLM are presented in Figure 5.13. The figure provides a detailed visualization of the classification outcomes for each architecture, highlighting both correctly classified instances and the most common confusion patterns between the target classes.

An examination of the confusion matrices reveals a consistent error pattern across all evaluated small models. Most misclassifications are concentrated in the left portion of the matrices, indicating that the models primarily struggle to distinguish *entailment* from *neutral* examples, as well as *contradiction* from *neutral* examples. In contrast, direct

Figure 5.13: Confusion matrices obtained for all fine-tuned Small Language Models.



(a) GPT-OSS-120B

(b) OpenChat 7B

(c) Phi3.8B

(d) Qwen2.5 3B

(e) Qwen2.5 7B

(f) Mistral 7B

Source: The Author.

confusion between *entailment* and *contradiction* is relatively uncommon. This behavior suggests that the models are generally capable of identifying when two statements are semantically aligned or conflicting, but encounter greater difficulty when determining whether the relationship is sufficiently strong to warrant a definitive entailment or contradiction label rather than a neutral one.

Regarding the Phi3.8B model, the error distribution for the contradiction class is less concentrated. Instead of primarily misclassifying these instances as neutral, its errors are more distributed, with the model also incorrectly predicting a portion of them as entailment.

In contrast to the evaluated SLMs, the GPT-OSS-120B zero-shot baseline exhibits a distinctly different classification behavior. The confusion matrix reveals that this larger model produces a significantly higher number of errors when the true label is *contradiction*. Specifically, while the model correctly identifies 56% of the *contradiction* instances, the errors it makes within this class are not evenly distributed. It practically never confuses *contradiction* with *entailment*; instead, its misclassifications for this category are almost entirely concentrated in the *neutral* class.

As previously noted, the model produced 58 empty strings as outputs during evaluation. Because these invalid or missing responses could not be mapped to the predefined categorical labels, they were excluded from the confusion matrix, explaining why the displayed total falls slightly short of the full 5,000-sample test split.

To further illustrate this specific error pattern exhibited by the large model, the following presents two examples of instances where GPT-OSS-120B incorrectly classified a true *contradiction* as *neutral*:

Instance index: 20
Premise: A young boy, dressed in cowboy attire, rides a galloping horse.
Hypothesis: A woman chasing a horse down the street.
Correct Label: contradiction
Model Response: neutral

Instance index: 26
Premise: A man hides his face from the camera while eating dinner with friends.
Hypothesis: A woman is drinking coffee with her boss

Correct Label: `contradiction`

Model Response: `neutral`

An analysis of these specific instances across the other evaluated models highlights a notable difference in classification behavior. As demonstrated in Table 5.4, the fine-tuned Small Language Models generally predicted the correct label for these cases. This trend suggests that task-specific fine-tuning may help these models better navigate the nuanced categorical boundaries of the dataset. While the exact internal mechanisms driving these predictions remain an open question, the SLMs appeared more adept at capturing the semantic conflicts necessary to distinguish a *neutral* statement from a direct *contradiction*, a boundary that the zero-shot GPT-OSS seemed to find more challenging.

Table 5.4: Comparison of individual model predictions on specific *contradiction* instances from the test set

Model	Prediction	Correct?
Instance #20		
Premise: A young boy, dressed in cowboy attire, rides a galloping horse.		
Hypothesis: A woman chasing a horse down the street.		
Mistral 7B	*contradiction*	Yes
OpenChat 7B	*contradiction*	Yes
Phi3.8B	*contradiction*	Yes
Qwen2.5 3B	*contradiction*	Yes
Qwen2.5 7B	*contradiction*	Yes
GPT-OSS-120B	*neutral*	No
Instance #26		
Premise: A man hides his face from the camera while eating dinner with friends.		
Hypothesis: A woman is drinking coffee with her boss.		
Mistral 7B	*contradiction*	Yes
OpenChat 7B	*contradiction*	Yes
Phi3.8B	*contradiction*	Yes
Qwen2.5 3B	*neutral*	No
Qwen2.5 7B	*contradiction*	Yes
GPT-OSS-120B	*neutral*	No

Source: The Author.

Despite the fine-tuned SLMs demonstrating improved robustness on certain categorical boundaries, there remains a subset of the dataset that proved challenging across all architectures. Specifically, an analysis of the aggregated predictions revealed 192 instances where every evaluated model, both the SLMs and the GPT-OSS-120B baseline, failed to predict the gold label. A qualitative review of these universally misclassified cases suggests that many of them harbor inherent semantic ambiguities, making the "correct" classification highly subjective. To illustrate this, the following two examples repre-

sent instances where the ground truth label is arguably ambiguous:

Instance index: 223
Premise: A native american indian playing a musical instrument on a stage.
Hypothesis: A Native American playing music.
Correct Label: neutral
Models' Consensus Prediction: entailment

Instance index: 95
Premise: A muslim woman is reading something out loud into a microphone.
Hypothesis: The woman is speaking into a microphone.
Correct Label: neutral
Models' Consensus Prediction: entailment

In examining these examples, the semantic relationship between the premise and the hypothesis lacks the definitive clarity required for strict categorization. Depending on the subjective interpretation of the context, unstated assumptions, or the boundaries of the actions described, human readers could reasonably argue for multiple valid labels. Consequently, the uniform failure of the models on these 192 instances may not strictly reflect a lack of semantic reasoning capabilities; rather, it highlights the inherent noise and subjectivity present in large-scale human-annotated datasets like SNLI, where the rigid trichotomy of *entailment*, *neutral*, and *contradiction* often struggles to capture the full nuance of natural language.

5.4 Potential Benchmark Familiarity Analysis

Table 5.5 presents the consolidated results for this evaluation, reporting the aggregated mean values for semantic similarity, lexical overlap, and normalized Levenshtein distance across each base architecture. It should be noted that, due to the operational and infrastructural constraints discussed earlier, this benchmark familiarity analysis could not be conducted on the GPT-OSS-120B baseline. Consequently, the similarity metrics and the subsequent analysis presented in this section apply exclusively to the Small Language Models.

Table 5.5: Results of the prefix-completion test for potential benchmark familiarity, presenting the average semantic similarity, lexical overlap, and normalized Levenshtein distance between the generated continuations of the Small Language Models and the ground-truth SNLI suffixes

Model	Semantic Mean	Lexical Mean	Levenshtein Mean
Mistral 7B	0.2682	0.1416	0.2616
OpenChat	0.3106	0.1888	0.2484
Phi-3 3.8B	0.2825	0.2038	0.2697
Qwen2.5 3B	0.2454	0.0703	0.2197
Qwen2.5 7B	0.2670	0.1112	0.2346

Source: The Author

The empirical findings demonstrate that the similarity scores remain consistently low across all evaluated systems. Across all models, the lexical overlap values range between 0.07 and 0.20, indicating that the architectures failed to accurately retrieve the specific vocabulary of the original suffixes. Similarly, the normalized Levenshtein distances center around 0.21 to 0.26, suggesting substantial character-level divergence between the generated continuations and the ground-truth targets.

Ultimately, these numerical results should be interpreted cautiously. Low similarity values only suggest that the generated suffixes are not close to the original suffixes within the representational space defined by the adopted similarity metrics. However, they do not confirm the absence of memorization, nor can this evaluation definitively rule out broader forms of benchmark familiarity, prior exposure, or indirect knowledge of similar training data.

5.5 Robustness Analysis

To assess whether the knowledge acquired during adaptation remains stable under surface-level textual variations, a robustness analysis was conducted. Consistent with the constraints discussed in previous sections, the GPT-OSS-120B baseline could not be subjected to this evaluation due to operational limitations. Consequently, this analysis was performed exclusively on the fine-tuned SLMs. Lexical perturbations were introduced through noun synonym substitutions while preserving the original NLI labels. The resulting 5,000-instance test set enabled the evaluation of model performance under controlled distribution shifts. Table 5.6 summarizes the results of this robustness assessment.

The results show a performance decrease across all evaluated fine-tuned models when compared to the original test set. Although the noun substitutions were designed

Table 5.6: Results of the robustness analysis evaluating the impact of semantic-preserving noun synonym substitutions on the predictive performance of the fine-tuned Small Language Models compared to the original SNLI test subset

Model	**Original SNLI Subset**		**Synonym Substitution**	
	Accuracy	**Macro F_1**	**Accuracy**	Macro F_1
Phi-3 3.8B Fine-Tuned	0.8648	0.8655	0.7484	0.7487
Mistral 7B Fine-Tuned	0.8700	0.8712	0.7630	0.7653
OpenChat 7B Fine-Tuned	0.8890	0.8890	0.7926	0.7941
Qwen2.5 3B Fine-Tuned	0.8680	0.8698	0.7612	0.7629
Qwen2.5 7B Fine-Tuned	0.8900	0.8906	0.7675	0.7691

Source: The Author

to preserve the semantic meaning of the sentences and did not alter the intended NLI labels, the modified lexical patterns negatively affected model predictions. These findings indicate that prediction quality is sensitive to lexical variations, even when the underlying semantic relationships remain unchanged.

Despite this degradation, the overall performance remained relatively stable, indicating that the fine-tuning process successfully captured the underlying task requirements beyond simple word matching. Moreover, most fine-tuned models achieved adversarial performance comparable to or higher than the baseline GPT-OSS-120B accuracy on the original test set (0.7688). These findings reinforce the importance of evaluating language models not only through their predictive accuracy but also through their robustness to input variations, as performance measured under a single evaluation configuration may provide an incomplete picture of model reliability (NALBANDYAN; SHAHBAZYAN; BAKHTURINA, 2025).

6 DISCUSSION

This chapter consolidates the results obtained across all experiments conducted in this study. The objective is to provide a unified comparison of all evaluated configurations, including zero-shot, few-shot, supervised instruction tuning, and the combination of trained models with few-shot prompting. In addition to predictive performance metrics, this section also summarizes the environmental and operational efficiency results, enabling a comprehensive assessment of the trade-offs between effectiveness and computational cost across the evaluated Small Language Models.

6.1 Predictive Performance

Table 6.1 summarizes the Accuracy and Macro F_1-Score achieved by each model under every experimental setting.

The consolidated results reveal that the best-performing configuration varied across the evaluated architectures. For Mistral 7B, the highest predictive performance was achieved after combining supervised instruction tuning with few-shot prompting, reaching an Accuracy of 0.8750 and a Macro F_1-Score of 0.8757. A similar pattern was observed for Phi-3 3.8B and Qwen2.5 7B, whose Fine-Tuned + Few-Shot configurations obtained the best results within their respective model families, achieving Accuracy values of 0.8872 and 0.8964, respectively.

For OpenChat 7B, supervised instruction tuning produced the strongest performance, with both the Fine-Tuned and Fine-Tuned + Few-Shot configurations reaching identical results. This indicates that the addition of few-shot demonstrations did not provide any measurable benefit after the model had already been adapted to the task.

In contrast, Qwen2.5 3B achieved its highest performance using only supervised instruction tuning. The introduction of few-shot demonstrations led to a noticeable decrease in performance, suggesting that the additional in-context examples may have interfered with the model's learned decision boundaries.

Overall, supervised instruction tuning was the most effective strategy across all evaluated models, consistently producing substantial improvements over both zero-shot and few-shot prompting approaches.

Figure 6.1 presents the highest Accuracy and Macro F_1-Score achieved by each evaluated model across all experimental configurations. For each architecture, only the

Table 6.1: Comprehensive summary of predictive performance across all evaluated Small Language Models under zero-shot, few-shot, and fine-tuned configurations, with the best-performing setup for each architecture highlighted in bold, alongside the GPT-OSS-120B zero-shot baseline for comparison

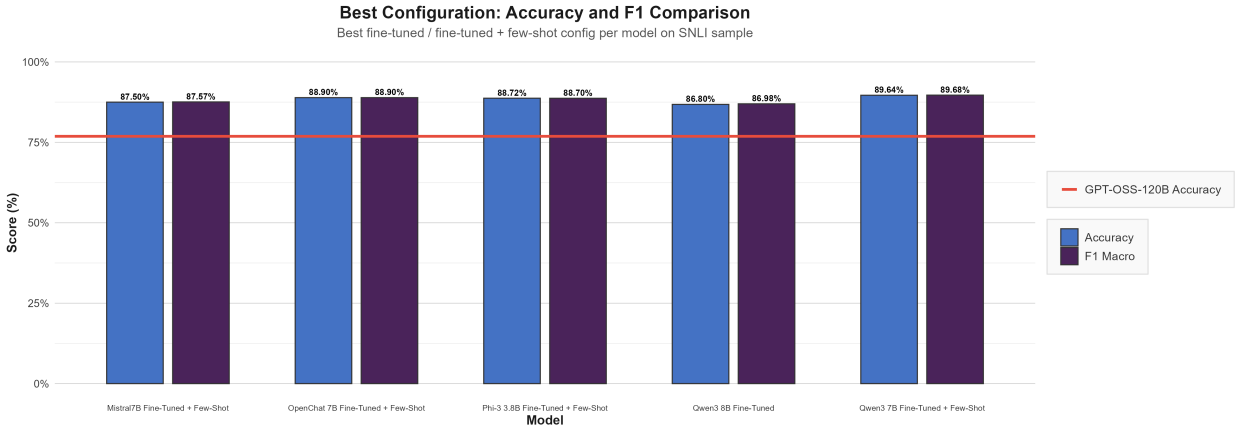
Model Configuration	**Accuracy**	**Macro F_1-score**
Mistral 7B Zero-Shot	0.5532	0.5531
Mistral 7B Fixed Few-Shot	0.7122	0.6841
Mistral 7B Random Few-Shot	0.6876	0.6411
Mistral 7B Fine-Tuned	0.8700	0.8712
Mistral 7B Fine-Tuned + Few-Shot	**0.8750**	**0.8757**
OpenChat 7B Zero-Shot	0.8672	0.8677
OpenChat 7B Fixed Few-Shot	0.8706	0.8711
OpenChat 7B Random Few-Shot	0.8554	0.8553
OpenChat 7B Fine-Tuned	**0.8890**	**0.8890**
OpenChat 7B Fine-Tuned + Few-Shot	**0.8890**	**0.8890**
Phi-3 3.8B Zero-Shot	0.6290	0.5815
Phi-3 3.8B Fixed Few-Shot	0.8454	0.8466
Phi-3 3.8B Random Few-Shot	0.7538	0.7533
Phi-3 3.8B Fine-Tuned	0.8648	0.8655
Phi-3 3.8B Fine-Tuned + Few-Shot	**0.8872**	**0.8870**
Qwen2.5 3B Zero-Shot	0.7466	0.7473
Qwen2.5 3B Fixed Few-Shot	0.8064	0.8083
Qwen2.5 3B Random Few-Shot	0.7138	0.7024
Qwen2.5 3B Fine-Tuned	**0.8680**	**0.8698**
Qwen2.5 3B Fine-Tuned + Few-Shot	0.8280	0.8289
Qwen2.5 7B Zero-Shot	0.7614	0.7648
Qwen2.5 7B Fixed Few-Shot	0.8578	0.8587
Qwen2.5 7B Random Few-Shot	0.8448	0.8466
Qwen2.5 7B Fine-Tuned	0.8900	0.8906
Qwen2.5 7B Fine-Tuned + Few-Shot	**0.8964**	**0.8968**
GPT-OSS-120B Zero-Shot	0.7688	0.7674

Source: The Author

best-performing setup identified during the experiments is reported, allowing a direct comparison of the maximum predictive capability attained by each SLM.

As shown in Figure 6.1, all best-performing configurations substantially outperformed the GPT-OSS-120B baseline in both Accuracy and Macro F_1-Score. This result demonstrates that, when properly configured through prompt engineering and/or supervised instruction tuning, Small Language Models can achieve predictive performance superior to that of a large-scale Mixture-of-Experts (MoE) language model, despite its considerably larger total parameter count.

Figure 6.1: Best model configurations

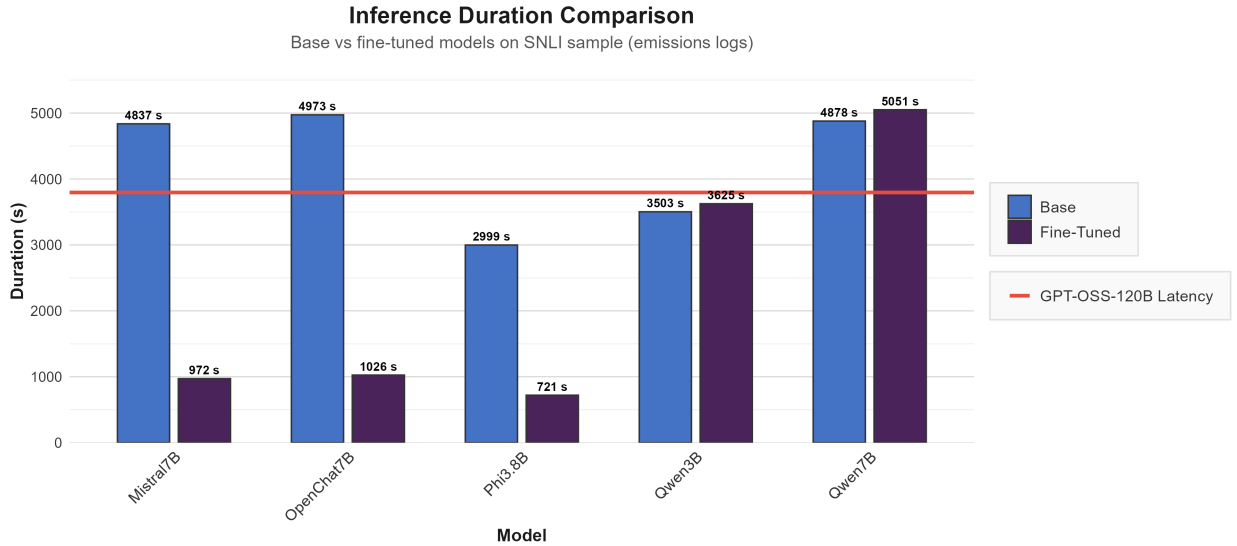


Source: The Author.

6.2 Environmental Performance

This section summarizes the environmental and operational efficiency metrics obtained for the evaluated models, including both the original architectures and their fine-tuned versions. Table 6.2 reports the execution duration, estimated carbon emissions, CPU energy consumption, GPU energy consumption, RAM energy consumption, and total energy consumed during inference.

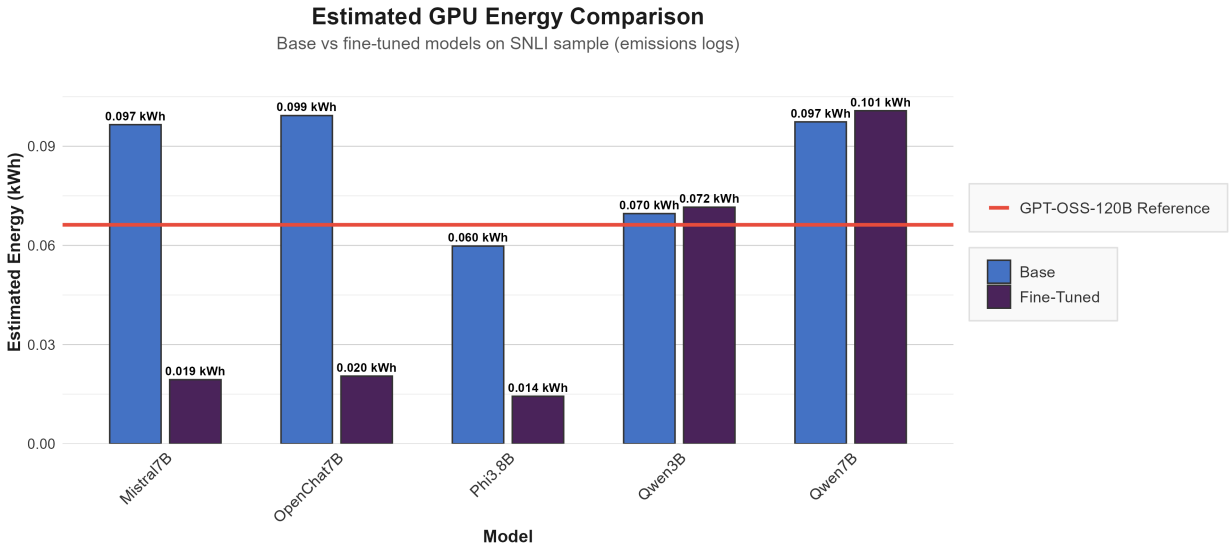
Figure 6.2: Comparison of execution duration across evaluated models



Source: The Author.

Table 6.2, together with Figures 6.2 and 6.3, shows that the environmental impact of supervised instruction tuning varied across the evaluated models. The figures provide a

Figure 6.3: Comparison of total energy consumption across evaluated models



Source: The Author.

Table 6.2: Comparison of environmental and operational efficiency metrics between original and fine-tuned Small Language Models, highlighting changes in execution duration, estimated carbon emissions, and component-level energy consumption

Model	Duration (s)	Emissions	CPU Energy	GPU Energy	RAM Energy	Energy Consumed
Mistral 7B	4836.88	0.0630	0.0571	0.0965	0.0269	0.1805
Mistral 7B Fine-Tuned	**972.42**	**0.0171**	**0.0115**	**0.0194**	**0.0054**	**0.0363**
OpenChat 7B	4973.35	0.0874	0.0587	0.0993	0.0276	0.1856
OpenChat 7B Fine-Tuned	**1026.38**	**0.0180**	**0.0121**	**0.0205**	**0.0057**	**0.0383**
Phi-3 3.8B	2998.73	0.0527	0.0354	0.0598	0.0167	0.1118
Phi-3 3.8B Fine-Tuned	**721.31**	**0.0127**	**0.0085**	**0.0144**	**0.0040**	**0.0269**
Qwen2.5 3B	3502.74	0.0455	0.0413	0.0696	0.0195	0.1304
Qwen2.5 3B Fine-Tuned	**3624.61**	**0.0633**	**0.0428**	**0.0716**	**0.0201**	**0.1345**
Qwen2.5 7B	4878.43	0.0636	0.0576	0.0974	0.0271	0.1820
Qwen2.5 7B Fine-Tuned	**5050.52**	**0.0887**	**0.0596**	**0.1007**	**0.0280**	**0.1884**
GPT-OSS-120B	–	–	–	**0.0662**	–	–

Source: The Author

visual comparison of the relationships between carbon emissions, execution duration, and total energy consumption, while the table reports the detailed measurements. For Mistral 7B, OpenChat 7B, and Phi-3 3.8B, the fine-tuned versions substantially lower execution time, energy consumption, and carbon emissions compared to the original counterparts. The most significant reduction was observed for Phi-3 3.8B, whose fine-tuned version consumed approximately four times less energy than the baseline model. In contrast, the fine-tuned versions of Qwen2.5 3B and Qwen2.5 7B exhibited slightly higher execution times, energy consumption, and emissions than their original versions.

These results suggest that the operational effects of fine-tuning are architecture-dependent. In some cases, instruction tuning was associated with both improved predictive performance and lower inference cost, whereas in others it introduced a modest

increase in computational overhead.

However, these efficiency differences should be interpreted cautiously, since they may also reflect variations in the number of generated tokens, inference configuration, output format, or early stopping behavior. Fine-tuning does not necessarily make the model computationally lighter, rather, it may lead the model to produce shorter, more direct, or better-formatted outputs for the target task.

7 CONCLUSION

This study provided a comprehensive empirical evaluation of the trade-offs between predictive performance and computational efficiency when comparing SLMs and an LLM on NLI tasks.

The experimental results demonstrate that massive model scale is not an absolute prerequisite for semantic inference in structured NLI tasks. Under zero-shot and few-shot prompting configurations, some SLMs such as OpenChat 7B and Qwen2.5 7B performed competitively, with OpenChat 7B directly surpassing GPT-OSS's zero-shot predictive accuracy. Furthermore, the application of Parameter-Efficient Fine-Tuning (PEFT) via Low-Rank Adaptation (LoRA) proved to be the most transformative strategy.

This transformative impact was particularly striking for models like Mistral 7B and Phi-3 3.8B, which exhibited tremendous performance improvements post-adaptation. Specifically, Mistral 7B's accuracy surged from an initial zero-shot baseline of 55.32% to 87.00% after fine-tuning, while Phi-3 3.8B demonstrated a similar leap, climbing from 62.90% to 86.48%. Ultimately, after being fine-tuned on a representative, class-balanced subset of the SNLI dataset, **all evaluated Small Language Models surpassed the GPT-OSS-120B zero-shot baseline**, achieving accuracy gains of up to 12.7% over the zero-shot GPT-OSS.

These results suggest that the operational effects of fine-tuning may vary depending on the architecture. In some instances, instruction tuning was associated with both enhanced predictive performance and a reduction in inference costs. For example, architectures such as Mistral, OpenChat, and Phi3.8B showed noticeable decreases in inference latency, energy consumption, and carbon emissions, with the fine-tuned Phi3.8B model requiring nearly four times less energy than its base version. Conversely, others, such as the Qwen models, displayed a modest increase in computational overhead post-tuning. Nevertheless, these efficiency differences should be interpreted with caution. They may also reflect variations in the number of generated tokens, inference configurations, output formats, or early stopping behaviors. Therefore, rather than inherently making the model computationally lighter, fine-tuning might simply guide it to produce shorter, more direct, or better-formatted outputs tailored to the target task.

Although the adversarial robustness analysis using noun synonym substitutions exposed a minor performance degradation across all adapted models, their overall accuracy remained highly competitive, frequently exceeding the GPT-OSS zero-shot perfor-

mance on the original test set.

In conclusion, this research confirms that when proper prompting mechanics and parameter-efficient fine-tuning strategies are aligned, Small Language Models provide a highly viable, sustainable, and powerful alternative for resource-constrained, deployment-sensitive, and reasoning-intensive NLP applications.

7.1 Next Steps

While this work provides a solid foundation for the development of task-specific language models under computational constraints, several avenues remain open for future research to further bridge the gap between predictive performance and computational efficiency:

- **Advanced Prompting and Reasoning Architectures:** Future investigations should explore the integration of Chain-of-Thought (CoT) prompting techniques directly into the fine-tuning loop of SLMs. Explicitly training compact models to generate intermediate, step-by-step rationales prior to outputting a final classification label could substantially boost logical stability and mitigate the residual vulnerabilities exposed during our robustness analysis.
- **Knowledge Distillation Regimes:** Rather than relying solely on supervised fine-tuning from gold-standard human annotations, incorporating dark knowledge from larger teacher models via structured Knowledge Distillation presents a promising direction. Transitioning reasoning traces, logit distributions, or attention maps from an LLM into an SLM could compress abstract cognitive capabilities into a fraction of the parameter budget.
- **Out-of-Subset Few-Shot Selection:** In this study, the demonstration examples for both fixed and random few-shot prompting were extracted from the training split of the selected 17,000-instance subset. For fine-tuned models, this introduces a structural redundancy since the models have already observed these pairs during the adaptation phase. Future research should address this limitation by sourcing few-shot examples completely outside the training, validation, and test splits. Specifically, these demonstrations should be harvested directly from the original SNLI dataset pool, ensuring they consist of instances completely excluded from the created subset to evaluate reasoning boundaries under strict out-of-distribution context

constraints.

- **Evaluation Against Large Dense Architectures:** Expanding the comparative baseline to include large dense models (e.g., in the 70B+ parameter class) evaluated under both zero-shot and few-shot prompting configurations. Comparing the few-shot capabilities of these massive dense architectures directly against task-specific, fine-tuned SLMs will provide deeper insights into the exact parameter thresholds where dense scaling yields diminishing returns for structured NLI tasks, further contextualizing the efficiency-performance trade-offs.
- **Fully Controlled Benchmarking Environments:** A prominent limitation of the current baseline evaluation was accessing GPT-OSS-120B through a black-box API infrastructure. This introduced variables like proprietary hardware accelerations, continuous batching, and speculative decoding that skewed direct latency comparisons. Future research should look to host both the large baseline models and the small target models within a single, fully controlled local cluster, if possible. Eliminating network dependencies and opaque API layers will allow for direct, unclouded physical measurements of hardware-level telemetry, precise GPU power draws, and holistic environmental impacts across all model scales.

7.2 Use of Artificial Intelligence

In compliance with academic integrity guidelines and scientific transparency, it is declared that AI tools were used in an assistive capacity during the development of this work. The use of these technologies was strictly limited to supporting technical writing and textual review of this study. The AI tools were applied to rephrase sentences, improve stylistic clarity, and correct grammar, ensuring that the text maintained the required academic rigor.

AI tools were also used as auxiliary resources during the literature review, particularly to support the identification of potentially relevant keywords, publications, and related studies in the fields of natural language processing and computational efficiency. All references and bibliographic information suggested through these tools were verified by the author using academic databases and the original publications. AI-generated content was not treated as a scientific source.

The author independently formulated the research problem, designed and imple-

mented the methodology, executed the training and inference processes, collected telemetry data, and interpreted the results. Additionally, the author critically evaluated all AI-assisted content, assuming complete responsibility for the accuracy, originality, and conclusions of this work.

REFERENCES

ABDIN, M. et al. Phi-3 technical report: A highly capable language model locally on your phone. **arXiv preprint arXiv:2404.14219**, 2024.

AGARWAL, S. et al. gpt-oss-120b & gpt-oss-20b model card. **arXiv preprint arXiv:2508.10925**, 2025.

BAI, J. et al. Qwen technical report. **arXiv preprint arXiv:2309.16609**, 2023.

BELCAK, P. et al. Small language models are the future of agentic AI. **arXiv preprint arXiv:2506.02153**, 2025.

BOWMAN, S. R. et al. A large annotated corpus for learning natural language inference. In: **Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing**. [S.l.]: Association for Computational Linguistics, 2015.

BROWN, T. B. et al. Language models are few-shot learners. **arXiv preprint arXiv:2005.14165**, 2020.

COURTY, B. et al. **mlco2/codecarbon**. Zenodo, 2024. Available from Internet: <https://doi.org/10.5281/zenodo.11171501>.

DAGAN, I.; GLICKMAN, O.; MAGNINI, B. The pascal recognising textual entailment challenge. In: **Machine Learning Challenges - Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment: First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Revised Papers**. Southampton, United Kingdom: Springer, 2006. (Lecture Notes in Computer Science (LNAI), v. 3944), p. 177–190. Available from Internet: <https://cris.biu.ac.il/en/publications/the-pascal-recognising-textual-entailment-challenge-2/>.

DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805**, 2019.

FEDUS, W.; ZOPH, B.; SHAZEER, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. **Journal of Machine Learning Research**, v. 23, n. 120, p. 1–39, 2022.

FU, Y. et al. Does data contamination detection work (well) for llms? a survey and evaluation on detection assumptions. In: **Findings of the Association for Computational Linguistics: NAACL 2025**. Association for Computational Linguistics, 2025. Available from Internet: <https://aclanthology.org/2025.findings-naacl.291/>.

Groq. **GroqCloud Console**. 2026. <https://console.groq.com/home>. Accessed: 2026-05-25.

GUBELMANN, R. et al. Capturing the varieties of natural language inference: A systematic survey of existing datasets and two novel benchmarks. **Journal of Logic, Language and Information**, 2024.

HINTON, G.; VINYALS, O.; DEAN, J. Distilling the knowledge in a neural network. **arXiv preprint arXiv:1503.02531**, 2015.

HSIEH, C.-Y. et al. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. **arXiv preprint arXiv:2305.02301**, 2023.

HU, E. J. et al. Lora: Low-rank adaptation of large language models. **arXiv preprint arXiv:2106.09685**, 2021.

JAVAHERIPI, M.; BUBECK, S. **Phi-2: The Surprising Power of Small Language Models**. 2023. Microsoft Research Blog.

JIANG, A. Q. et al. Mistral 7b. **arXiv preprint arXiv:2310.06825**, 2023.

LIU, Y. et al. Roberta: A robustly optimized bert pretraining approach. **arXiv preprint arXiv:1907.11692**, 2019.

MAGISTER, L. C. et al. Teaching small language models to reason. In: **Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)**. Association for Computational Linguistics, 2023. Available from Internet: <https://aclanthology.org/2023.acl-short.151/>.

MILLER, G. A. Wordnet: A lexical database for english. **Communications of the ACM**, v. 38, n. 11, 1995. Available from Internet: <https://doi.org/10.1145/219717.219748>.

NALBANDYAN, G.; SHAHBAZYAN, R.; BAKHTURINA, E. Score: Systematic consistency and robustness evaluation for large language models. In: **Proceedings of the 2025 Annual Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics (NAACL)**. [S.l.: s.n.], 2025.

NVIDIA. **NVIDIA H100 Tensor Core GPU**. 2024. <https://www.nvidia.com/en-us/data-center/h100/>. Accessed: 2026-05-25.

OpenAI. **Introducing gpt-oss**. 2025. <https://openai.com/index/introducing-gpt-oss>. Accessed: 2026-05-25.

PATTERSON, D. et al. Carbon emissions and large neural network training. **arXiv preprint arXiv:2104.10350**, 2021.

SADAT, M.; CARAGEA, C. Mscinli: A diverse benchmark for scientific natural language inference. In: **Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)**. Association for Computational Linguistics, 2024. Available from Internet: <https://aclanthology.org/2024.naacl-long.90/>.

SANH, V. et al. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. **arXiv preprint arXiv:1910.01108**, 2019.

STRUBELL, E.; GANESH, A.; MCCALLUM, A. Energy and policy considerations for deep learning in NLP. In: **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics**. [S.l.]: Association for Computational Linguistics, 2019.

SUBRAMANIAN, S.; ELANGO, V.; GUNGOR, M. Small language models (slms) can still pack a punch: A survey. **arXiv preprint arXiv:2501.05465**, 2025.

THELWALL, M. Can smaller large language models evaluate research quality? **arXiv preprint arXiv:2508.07196**, 2025.

VASWANI, A. et al. Attention is all you need. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2017.

WANG, F. et al. A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with LLMs, and trustworthiness. **arXiv preprint arXiv:2411.03350**, 2024.

WANG, G. et al. Openchat: Advancing open-source language models with mixed-quality data. **arXiv preprint arXiv:2309.11235**, 2024.

WILLIAMS, A.; NANGIA, N.; BOWMAN, S. R. A broad-coverage challenge corpus for sentence understanding through inference. In: **Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)**. Brussels, Belgium: Association for Computational Linguistics, 2018. p. 1112–1122. Available from Internet: <https://aclanthology.org/D18-1198/>.

YANG, A. et al. Qwen2.5 technical report. **arXiv preprint arXiv:2412.15115**, 2025.

ZAFRIR, O. et al. Q8bert: Quantized 8bit BERT. **arXiv preprint arXiv:1910.06188**, 2019.

ZHANG, P. et al. Tinyllama: An open-source small language model. **arXiv preprint arXiv:2401.02385**, 2024.