University of Bahrain

College of Information Technology

Department of Computer Science

ARABIC LLM OPTIMIZATION FOR BROWSER DEPLOYMENT

Prepared by

Fatima Falah A.majeed 202200527

Shahad Yasser A.Rahman 202210309

For ITCS 498 Senior Project

Academic Year 2025-2026 Semester 1

Project Supervisor: Dr. Abdulla Ebrahim Subah

Date of Submission 12 May 2026

Abstract

This work evaluates the effectiveness of several model compression and optimization techniques on large language models, namely quantization, pruning, and knowledge distillation, with a focus on Arabic natural language performance and browser deployment. The primary methods investigated are 8-bit and 4-bit quantization with several methods such as GPTQ, LLM.int8(), and QLoRA's NormalFloat4, SparseGPT and Wanda for pruning, and a knowledge distillation pipeline based on a Qwen2.5-32B-Instruct teacher model and a Qwen2.5-7B-Instruct student model. The proposed SelecTKD method filters low-confidence teacher tokens during token-level distillation to improve bilingual Arabic-English balance. The study also includes staged fine-tuning to adapt the model to Arabic, GCC, and Bahraini contexts using a broad-to-specific curriculum while preserving bilingual performance. The report concludes with a comprehensive comparative analysis between the tested compression methods, comparing their compression and accuracy tradeoff and discussing their practical effectiveness for limited-resource deployment.

Acknowledgments

We praise and thank Allah, Glorified and Exalted, for the blessing of knowledge, for divine guidance, and blessings in completing this research; indeed, Allah is the Bestower of success and the Director of affairs.

Our deepest gratitude and appreciation go to Dr. Abdulla Ebrahim Subah, who was with us every step of the way, supporting us with all his ability and zeal for this significant scientific achievement. Integrating our Arab Gulf Bahraini identity with artificial intelligence in this research is something a beautiful thing and a divine blessing, and it would not have been possible without his guidance and encouragement.

We also thank our families and friends, all the professors and supervisors, and everyone who contributed to us in spirit and heart in realizing this work. May Allah reward them abundantly and count it among their good deeds.

This project was made possible with the help of the computing resources from the University of Bahrain's Benefit Advanced AI and Computing Lab.

Our sincere gratitude and appreciation to all whom are responsible.

نحمدُ اللهَ سُبْحَانَهُ وَتَعَالَى، وَنشْكُرُهُ عَلَى سابِغِ نِعَمِهِ وحُسنِ ألطَافِهِ وتَيسِيرِهِ الدَائِم ورَحمَتِهِ الواسِعَة، ونِعْمَةِ الْعِلْمِ وَتَوْفِيقِهِ وَبَرَكَاتِهِ فِي إِتْمَامِ هَذَا الْبَحْثِ. إِنَّهُ هُوَ الْمُوَفِّقُ وَالْمُسَدِّد.

نعَبِّرُ عَنْ خَالِصِ شُكْرِنا الْجَزِيلَ لِلْدَّكْتُورِ\عبد الله إبراهيم صباح الَّذِي كَانَ مَعَنَا فِي كُلِّ لَحْظَةٍ، وَدَعْمَنَا بِكُلِّ مَا اسْتَطَاعَ، وَحَرَصَ عَلَى هَذَا الْإِنْجَازِ الْعِلْمِيِّ الْكَبِيرِ. إِنَّ دَمْجَ هُوِيَّتِنَا الْعَرَبِيَّةِ الْخَلِيجِيَّةِ الْبَحْرَيْنِيَّةِ مَعَ الذَّكَاءِ الْاصْطِنَاعِيِّ فِي هَذَا الْبَحْثِ أَمْرٌ جَمِيلٌ وَتَوْفِيقٌ مِنَ اللهِ، وَلَمْ يَكُنْ لِيَحْقُقْ بِدُونِ إِرْشَادِهِ وَحَثِّهِ.

كَمَا نشْكُرُ أَهْلنا وَأَصْدِقَائِنا، وَجَمِيعَ الأَساتِذَة وَالْمُشْرِفِينَ، وَكُلَّ مَنْ سَاهَمَ مَعَنَا نَفْسِيًّا وَرُوحِيًّا وَقَلْبِيًّا فِي تَحْقِيقِ هَذَا الْعَمَلِ. جَزَاهُمُ اللهُ خَيْرًا وَنسْأَلُهُ أَنْ يَجْعَلَهُ فِي مِيزَانِ حَسَنَاتِهِم.

نشِيرُ أَنَّ هَذَا الْمَشْرُوعَ تَمَّ بِفَضْلِ مَوَارِدِ مختبر الحوسبة والذكاء الاصطناعي الكائِن بِجَامِعَةِ البَحرَين فَشُكْرِنا وَتَقْدِيرِنا لِكُلِّ مَنْ كَانَ لَهُ دَوْرٌ فِي تَوْفِيرِهَا.

وَآخِرُ دَعْوَاهُمْ أَنِ الْحَمْدُ لِلَّهِ رَبِّ الْعَالَمِينَ.

Table of Contents

ABSTRACT ..... **II**

ACKNOWLEDGMENTS ..... **III**

LIST OF TABLES ..... **VII**

LIST OF FIGURES ..... **VIII**

CHAPTER 1 INTRODUCTION **1**

INSPIRATION **1**

1.1 PROBLEM STATEMENT **1**

1.1.1 ACCURACY IMPACT 1

1.1.2 COMPUTATIONAL EFFICIENCY 2

1.1.3 DEPLOYMENT FEASIBILITY 2

1.2 PROJECT OBJECTIVES **2**

1.3 SIGNIFICANCE OF THE PROJECT **2**

1.4 RESEARCH METHODOLOGY **3**

1.5 SCOPE AND LIMITATIONS **3**

1.6 REPORT OUTLINE **3**

CHAPTER 2 LITERATURE REVIEW **4**

2.1 INTRODUCTION TO LARGE LANGUAGE MODELS **4**

2.2 ARABIC NLP AND CHALLENGES OF ARABIC LLMS **5**

2.2.1 DIALECTAL VARIETY 5

2.2.2 COMPLEX MORPHOLOGY 6

2.2.3 ARABIC WEB CONTENT 6

2.2.4 AVAILABILITY AND QUALITY OF ARABIC DATASETS 6

2.3 EXISTING ARABIC AND MULTILINGUAL LLMS **7**

2.3.1 FANAR 7

2.3.2 ALLAM 7

2.3.3 ACEGPT 7

2.3.4 ARABIANGPT 8

2.3.5 GEMMA 3 8

2.3.6 QWEN3 8

2.4 MODEL COMPRESSION AND OPTIMIZATION **9**

2.4.1 Pruning 9
2.4.2 Knowledge Distillation 9
2.4.3 Quantization 10
2.5 EVALUATION METRICS 10
2.5.1 Evaluation Frameworks 10
2.6 COMPARATIVE ANALYSIS 11
2.6.1 Combined Compression Strategies and Synergistic Effects 11
CHAPTER 3 METHODOLOGY 12
3.1 THE CHOSEN MODELS 12
3.2 EXPERIMENTAL FRAMEWORK 12
3.3 MODEL COMPRESSION AND OPTIMIZATION TECHNIQUES 14
3.3.1 Quantization 14
3.3.1.1 Post-Training Quantization (PTQ) 14
3.3.1.2 Quantization-Aware Training (QAT) 15
3.3.2 Pruning 16
3.3.2.1 SparseGPT 16
3.3.2.2 Wanda 18
3.3.3 Knowledge Distillation 19
3.3.3.1 Models and Training pipeline 19
3.3.3.2 Experimental Framework and Implementation 21
3.3.4 Fine-tuning 22
3.4 EVALUATION METHODS 23
3.4.1 Computational Metrics 23
3.4.2 Error Measurement 23
3.4.3 Task Benchmarks 23
3.5 ACAI - WEB APPLICATION DESIGN AND ARCHITECTURE 24
CHAPTER 4 RESULTS AND DISCUSSION 27
4.1 EXPERIMENTAL ENVIRONMENT AND SETUP 27
4.1.2 Quantization Setup 27
4.1.2 Pruning Setup 27
4.1.3 Knowledge Distillation Setup 27
4.2 QUANTIZATION 28
4.2.1 Compression Ratio 28

4.2.2 PERPLEXITY 28
4.2.3 TASK PERFORMANCE 29
4.3 PRUNING 31
4.3.1 COMPRESSION RATIO 31
4.3.2 PERPLEXITY 31
4.3.3 TASK PERFORMANCE 32
4.4 KNOWLEDGE DISTILLATION 35
4.4.1 TASK PERFORMANCE 35
4.4.1.1 General Reasoning Benchmark 35
4.4.1.2 Arabic and GCC-specific Evaluation 36
4.4.1.3 KD behavioral Metrices 37
4.4.1.4 Token-level fluency 38
4.4.2 REPRODUCIBILITY AND ABLATION 38
4.5 FINETUNING 39
4.6 DISCUSSION 41
CHAPTER 5 CONCLUSION AND FUTURE WORK 43
REFERENCES 45
APPENDIX A ADDITIONAL RESULTS 52

List of Tables

Table 1: Summary of the reviewed models. 8
Table 2: The models chosen for compression experiments. 12
Table 3: KD - Training Pipeline 20
Table 4: KD - Training Corpus 21
Table 5: Application - Overview of chat modes (agents) 25
Table 6: Quantization - Model sizes and compression ratios 28
Table 7: Quantization - Perplexity scores 29
Table 8: Quantization - Mean accuracy scores for ArabicMMLU and metabench 29
Table 9: Pruning - Original vs. pruned size of the model and the compression ratios. 31
Table 10: Pruning - Perplexity scores for baseline and SparseGPT/Wanda-compressed models 32
Table 11: Pruning - Mean accuracy scores of ArabicMMLU and metabench. 33
Table 12: KD - Layer 1 - Standard Benchmarks 36
Table 13:KD - AraDiCE-Culture Qatar and GCC Dialect 36
Table 14: KD - Habash Corpus Evaluation 37
Table 15: KD behavioral metrices: 30-pair parallel evaluation, greedy decoding 37
Table 16: KD – Perplexity scores 38
Table 17: KD - Seed reproducibility (n=3 seeds, 15-pair evaluation, greedy decoding) 38
Table 18: KD - Threshold, SelecTKD, and θ ablation (500 samples, 2 epochs) 39
Table 19: Complete evaluation result comparison on KD and fine-tuning 39
Table 20: Quantization - Per-task scores of ArabicMMLU and metabench. 52
Table 21: Pruning - Model parameter counts after pruning with SparseGPT/Wanda. 53
Table 22: Pruning – Per-task evaluation scores of ArabicMMLU and metabench. 53
Table 23: Multi-Layer Evaluation Results for Knowledge Distillation and Fine-Tuned Models 54

List of Figures

Figure 1: Evolution of Arabic Language Models since the 2000s and until 2024 (Al-Khalifa et al., 2025). .. 5
Figure 2: Overview of LLM evaluation methods. 10
Figure 3: Overview of the experimental framework (llmini) 13
Figure 4: Knowledge distillation experimental framework. 13
Figure 5: Fine-tuning experimental framework. 14
Figure 6: Application - Architecture overview. 24
Figure 7: Application - Data models diagram 26
Figure 8: Quantization – Average retained accuracies per method 30
Figure 9: Pruning - Perplexity scores comparison between dense and sparse variants. 32
Figure 10: Pruning - Accuracy retention per model. 34
Figure 11: Comparative performance analysis of baseline, knowledge distillation, and fine-tuned models across multilingual reasoning, Arabic/GCC evaluation, behavioral metrics, and perplexity benchmarks. 41

Chapter 1
Introduction

Inspiration

Large language models have fundamentally altered how machines process, understand, and generate text. Their growing scale, reach, and application especially in Arabic come with significant computational costs and nuanced linguistic challenges. Moreover, the Arabic language stands out with its remarkable richness, complex grammatical system, and profound linguistic sciences including derivation, metrics, prosody, and morphology. The significance of Arabic is further elevated by its status as the language of the Qur'an. As noted in the Qur'an: "**Indeed, we have sent it down as an Arabic Qur'an so that you may understand"** (Qur'an 12:2, Yusuf). [12:2 يوسف | **إِنَّا أَنزَلْنَاهُ قُرْآنًا عَرَبِيًّا لَّعَلَّكُمْ تَعْقِلُونَ**]. This divine selection underscores the language's historical and cultural prominence. Beyond its religious significance, Arabic has served as a vessel for codifying diverse sciences ranging from poetics and linguistics to jurisprudence (Fiqh فقه) and phonology, making it unique among global languages. Such depth and diversity present both inspiring opportunities and formidable challenges in designing and optimizing NLP systems tailored to Arabic, especially when aiming to preserve linguistic nuance and achieve state of the art performance.

1.1 Problem Statement

This report investigates the problem of Large Language Model (LLM) optimization for Arabic language processing in three critical aspects: accuracy impact, computational efficiency, and deployment feasibility.

1.1.1 Accuracy Impact

Accuracy, in the context of language models, refers to a measurement of a model's alignment with expected output, typically based on ground truth or human judgement (Hu and Zhou, 2024). Arabic LLMs possess unique linguistic and cultural complexities that are prone to degradation in performance in the event of model compression or optimization. Reducing the model size considerably while retaining at least 85% accuracy on Arabic NLP tasks like punctuation prediction, grammatical error correction, and culturally nuanced comprehension

remains an open issue. Existing multilingual models are noted to have unbalanced performance between high-resource languages like English and lower-resource languages like Arabic in dealing with different linguistic measurements and culturally nuanced contexts, which this project aims to address.

1.1.2 Computational Efficiency

The memory and computational demands of high-performing LLMs render them unsuitable to use effectively, particularly in offline and web settings. Techniques for optimization such as quantization, pruning, distillation, and recent techniques such as QLoRA and Flash Attention offer potential reduction in resources but need to be gauged specifically on Arabic and multilingual models in terms of effectiveness and trade-offs.

1.1.3 Deployment Feasibility

Implementing optimized Arabic and multilingual LLMs entirely offline on browser-compatible frameworks (such as TensorFlow.js and ONNX.js) imposes limitations on model size, latency, and compatibility. Achieving fast inference speed in offline web settings while preserving satisfactory accuracy in these settings presents considerable engineering and research obstacles that have yet to be adequately addressed for Arabic and multilingual LLMs.

1.2 Project Objectives

- Investigate various LLM optimization techniques including quantization, pruning, distillation, and recent innovations such as QLoRA and Flash Attention to enhance model computational efficiency.
- Select and adapt a leading open-source Arabic LLM (e.g., Jais, ALLaM, Fanar) for deployment within web browsers.
- Achieve at least a 50% reduction in model size while maintaining no less than 85% of the original accuracy on fundamental Arabic NLP tasks.
- Enable full offline operation by deploying the optimized model using browser-compatible frameworks such as TensorFlow.js or ONNX.js.
- Perform a comparative analysis of performance metrics, including model size, inference speed, and task accuracy, before and after applying optimization techniques.

1.3 Significance of the project

Arabic LLMs are useful for performing NLP tasks targeted at Arabic-speaking users, but significant computational demands limit low-resource local deployment. By methodically comparing the impact of various model compression techniques on computational demands and

task accuracy, this project can be used as a guide for reducing model sizes. Additionally, browser deployment addresses the challenge of device-specific deployment due to largely being device-agnostic (with the exception of requiring WebGPU support in the browser). Generally, such offline, low-resource setups are desired by companies and individuals seeking a level of information privacy and confidentiality that is not offered by third-party LLM providers.

1.4 Research Methodology

This study was conducted on four main steps. First, relevant literature was collected and reviewed. Second, a set of model compression techniques were applied on several chosen models. Third, the models were tested against a set of benchmarks including model size and tasks accuracy. Finally, a comparative analysis of the results was conducted.

1.5 Scope and limitations

The scope of this research project encompasses multilingual large language models based on the Transformers (Vaswani et al., 2017) architecture, with evaluation focusing on Arabic and English language tasks. The hardware resources available limits the choice models' to ones that can run on two A100 GPU with 40 GB of VRAM each, limiting the experiments to models not exceeding the size of around 32B parameters.

1.6 Report Outline

The outline of this report is as follows:

Chapter 2 presents a comprehensive review of the literature on all the different aspects of this research project: An overview of large language models, the unique nature of the Arabic language in the context of language modeling, the available Arabic-centered and multilingual LLMs, browser deployment, compression techniques, and evaluation metrics.

Chapter 3 describes the research methodology to be followed during every step of the project, including the used frameworks and libraries, in order to ensure reproducibility.

Chapter 4 goes through the results obtained from the conducted model compression and optimization experiments. The results obtained are discussed through their compression to accuracy tradeoff and relative to each other.

Chapter 5 concludes this report by summarizing the key results and highlighting the degree to which the research objectives were achieved. Future work and areas of potential research interest related to the topic of this report are also mentioned.

Appendix A contains additional results that are not shown in chapter 4.

Chapter 2
Literature Review

2.1 Introduction to Large language models

Large Language Models (LLMs) represent a breakthrough in Artificial Intelligence (AI) due to their outstanding performance over a wide array of natural language tasks. These models, through learning from massive amounts of text from the internet, are capable of understanding, generating, and responding to natural language through predicting the next sequence of words to very high levels of accuracy.

The evolution of language models has occurred through several phases. Beginning in the 1980s, Arabic Natural Language Processing (NLP) employed rule-based systems such as morphological analyzers that focused on computational instruments by extracting roots and addressing several tasks. Although rule-based systems handled some linguistic complexities, it struggled with scalability and ambiguity (Farghaly and Shaalan, 2009).

In the early 2000s, as shown in Figure 1, statistical language models (SLMs) were based on n-grams and stemming techniques, which focused on estimating, predicating the probability of text and machine translation. Although these models improved upon rule-based methods, it faced huge limitations on data availability, especially across Arabic dialects and linguistics complexities (Al-Khalifa et al., 2025).

A significant shift occurred around 2010, with the early Neural Language Models (NLMs) dealing with data by mapping words, and employing the Recurrent Neural Networks (RNNs) (Mikolov et al., 2010) with the latest techniques like Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNNs) reaching to pre-trained language models (PLMs). These developments increased accuracy in NLP tasks such as sentiment analysis, dialect identification, and machine translation. Despite these advances, persistent issues kept the necessity for more developed and scalable models.

This development led to emergence of PLMs such as word2Vec and Glove. These models enabled transfer learning, where it was pre-trained on large-scale data and then finetuned for specific tasks, achieving higher accuracy on these tasks than previous models.

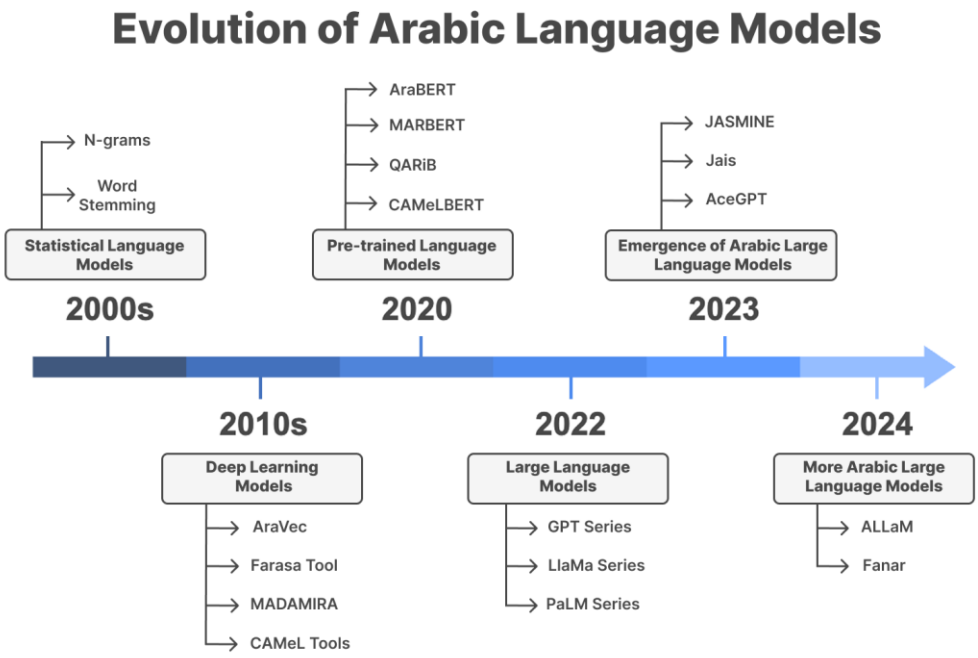


Figure 1: Evolution of Arabic Language Models since the 2000s and until 2024 (Al-Khalifa et al., 2025).

A revolutionary leap came with the Transformer architecture (Vaswani et al., 2017) with its robust framework for understanding and complex tasking, giving rise to models like BERT, and early transformer-based Arabic models such as AraBERT and QARIB, capable of leveraging large-scale transfer learning and handling intricate linguistic features unique to Arabic.

Despite being a relatively new technology and field of research, the rapid advancements in LLMs have led to their widespread across many applications which involves the generation of models for numerous languages and dialects. The fifth most spoken language in the world, Arabic, presents specific obstacles and possibilities for natural language processing (NLP).

2.2 Arabic NLP and Challenges of Arabic LLMs

The development of Arabic-centered LLMs and the performance of multilingual models in Arabic NLP tasks is limited by several factors and challenges. Most literature on Arabic language models includes a section on these challenges, and this section provides a summary of these challenges and limitations that affect the development of Arabic LLMs. The challenges include dialectal variety, complex morphology, Arabic web content, and the availability and quality of Arabic datasets.

2.2.1 Dialectal Variety

The Arabic language is spoken by around 422 million people globally, making it one of the top five most spoken languages in the world. Arabic comes in multiple forms, mainly standard

Arabic and dialectal Arabic, with over 30 dialects. This variety in dialects includes differences in vocabulary, grammar, and morphology, which poses a serious challenge for Arabic NLP.

2.2.2 Complex Morphology

The Arabic language has a rich and unique morphology that differs greatly from languages like English. In essence, Arabic consists of **roots جذور** that commonly consist of 3 constants (though sometimes 4 or 5). These roots are morphed using **patterns** or **templates أوزان** which generates most words in Arabic by substituting the root constants into the pattern. This uniqueness in morphology causes common tokenizers used in state-of-the-art language models to perform poorly on Arabic input (Asgari et al., 2025).

2.2.3 Arabic Web Content

One of the main limitations is the availability of Arabic data. LLMs are trained on internet-scale data; however Arabic web content accounts for only about 0.5% of the entire internet (W3Techs, 2025).

2.2.4 Availability and Quality of Arabic Datasets

The limited availability and quality of clean Arabic datasets are contributing factors to the low performance of LLMs in Arabic NLP tasks, and in the scope of benchmarking it poses a critical issue in the accuracy of Arabic LLMs evaluations (Al-Khalifa et al., 2025). Arabic data resources are often fragmented, inconsistently annotated, and unevenly distributed between Modern Standard Arabic (MSA) and dialectal verities, making task-specific training and accurate task performance evaluations difficult. Moreover, dialectal datasets suffer from a lack of availability and lower quality compared to MSA datasets, such as GCC datasets, or even more specifically Bahraini dialects dataset, where publicly available dataset is still scarce.

There have been some efforts to address this issue, such as the development of the ORCA (Elmadany et al., 2022) benchmark, which covers multiple Arabic varieties and Arabic NLU tasks across 60 datasets. In addition, Masader (Alyafeai et al., 2021) was developed as a publicly for improving discoverability and documentation of Arabic text and speech resources.

For dialect-specific work, the problem is even more severe. For instance, smaller dialect groups like the Bahraini dialect have received limited attention in comparison with larger dialect groups. However, there have been recent efforts that made important progress in collecting and releasing new data resources, including the large and culturally rich Bahrain corpus (Abdulrahim et al., 2022), recent Bahraini speech dataset (Barakat, 2026), and sentiment

datasets (Omran et al., 2023). Despite that availability is rising, the overall evidence still suggests that Arabic LLM performance is constrained not only by model design, but also by the limited scale, coverage, and cleanliness of the available datasets.

2.3 Existing Arabic and Multilingual LLMs

This section explores the available options for open-source Arabic and multilingual large language models. Table 1 provides a summary of the models discussed.

2.3.1 Fanar

Fanar (Fanar Team et al., 2025) is an Arabic-centric multimodal generative AI platform that supports language, speech, and image generation tasks. Two Arabic LLMs were developed for the platform: Fanar Star (7B) and Fanar Prime (9B). Fanar Star was trained from scratch on 1 trillion Arabic, English, and Code tokens. Fanar Prime was trained on the Gemma-2 9B base model on the same 1T tokens. Currently, the `QCRI/Fanar-1-9B-Instruct` model is available for use on Hugging Face[1]. The Open Arabic LLM Leaderboard (OALL) reports the model to have an average score of 70.83 across 7 benchmarks (El Filali et al., 2025).

2.3.2 ALLaM

ALLaM (Bari et al., 2024) is a family of large language models specially trained in Arabic, of which `humain-ai/ALLaM-7B-Instruct-preview` is available for use on Hugging Face[2]. The model is trained from scratch on 4T English tokens followed by 1.2T mixed Arabic/English tokens. The model's evaluation average score is 65.25 on OALL (El Filali et al., 2025).

2.3.3 AceGPT

AceGPT (Huang et al., 2023) is a family of models based on Llama2 and fine-tuned for the Arabic language. The models' sizes are 8B, 32B, and 70B, and are available for use on Hugging Face[3]. The evaluation average scores of the models are as follows: 62.35 for `AceGPT-v2-8B-Chat`, 70.88 for `AceGPT-v2-32B-Chat`, and 70.07 for `AceGPT-v2-70B-Chat` on OALL (El Filali et al., 2025).

[1] https://hf.co/collections/QCRI/fanar
[2] https://hf.co/humain-ai/ALLaM-7B-Instruct-preview
[3] https://hf.co/collections/FreedomIntelligence/acegpt-v2

2.3.4 ArabianGPT

ArabianGPT (Koubaa et al., 2024) is a series of large language models designed for Arabic. Several model sizes are available on Hugging Face[4]: 0.1B, 0.3B, 0.8B, and 1.5B parameter models. Unfortunately, the models are not submitted on OALL, but the paper reports the evaluation's average score to be 31.9 for the 0.1B model, and 32.7 for the 0.3B model.

2.3.5 Gemma 3

Gemma 3 (Gemma Team et al., 2025) is the latest version of the Gemma family; a family of small, open-weight, multimodal models developed by Google DeepMind with pre-trained and instruction-tuned variants. It supports a 128K tokens context window, over 140 languages, and is available in multiple parameter sizes: 270M, 1B, 4B, 12B, and 27B. The `google/gemma-3-27b-it` model is evaluated on OALL with an average score of 71.4 (El Filali et al., 2025).

2.3.6 Qwen3

Qwen3 (Yang et al., 2025) is a series of LLMs and is the latest version of the Qwen family, with parameter sizes ranging from 0.6B to 235B. It follows a dense and Mixture-of-Experts (MoE) architecture, and combines thinking-mode and non-thinking mode into a single framework, eliminating the need for switching between a dedicated reasoning model and a chat-optimized model. Qwen3 also expands on multilingual capabilities, increasing the supported languages to 119 from the previous 23 languages in its predecessor Qwen2.5.

Table 1: Summary of the reviewed models.

Model	Parameter Size	OALL Score
Fanar	9B	70.83
ALLaM	7B	65.25
AceGPT	8B	62.35
	32B	70.88
	70B	70.07
ArabianGPT	0.1B	31.9
	0.3B	32.7
	0.8B, 1.5B	-
Gemma 3	270M – 27B	71.4 (27B)
Qwen2.5-3.0	0.6B – 235B	-

[4] https://hf.co/collections/riotu-lab/arabianllm-series-native-arabic-large-language-models-66cc491302db9143d09c1a5a

2.4 Model Compression and Optimization

Model compression comprises of many machine learning techniques that aim to reduce the size of large models with minimal performance degradation.

2.4.1 Pruning

Pruning (both structured and unstructured) is a compression method that reduces the effective model parameters by removing or setting the less important weights or layers to zero based on some pruning criteria or through learning-based approaches. Pruning methods are generally classified into unstructured, semi-structured, and structured.

Research on neural network pruning can be traced back to the Optimal Brain Damage (LeCun et al., 1989) method, which layered the foundation for later works such as the Optimal Brain Surgeon (Hassibi and Stork, 1992) method. Since then, pruning as a compression technique for early neural networks has been heavily investigated, proving to be an effective method for significantly reducing model sizes while retaining high accuracy.

While pruning can significantly degrade accuracy, strategies such as iterative pruning, layer merging, and fine-tuning with self-distillation were shown to aid with performance recovery. The literature shows pruning is often most effective when combined with other compression techniques, such as knowledge distillation and quantization for balanced compression.

2.4.2 Knowledge Distillation

Knowledge distillation is a method compression technique where a smaller student model is trained to mimic the behavior of the larger teacher model.

The concept of “Model compression.” was introduced first by Rich Caruana et al. (2006), Which utilized supervised learning and state-of-art classification, where a large ensemble model compromised hundreds of classifiers to label a dataset, to be trained by single neural network. Compressing the large complex ensembles into smaller faster models.

Then formalized by Hinton et al. (2015), where they named the term of “distillation”. This term refers to the process of transferring knowledge from large complex model to smaller more efficient model, by matching the teacher’s soft probabilities using temperature parameter in the SoftMax function to reveal richer inter-class relationships, to enable student to learn the structural similarities which they called “dark knowledge”.

Knowledge distillation is presented in the edge of Large language models as pivotal methodology for transferring advanced capabilities from leading LLMs, often boosting or maintaining accuracy despite reductions in size. Research highlights various distillation techniques including self-distillation, task-specific optimizations, and integration with other compressions like

quantization and pruning. It shows promise in mitigating the accuracy drop caused by other compression methods and enhancing generalization (d'Aloisio et al., 2024; Movva et al., 2022; Xu et al., 2024; Tina et al., 2025).

2.4.3 Quantization

Quantization is extensively studied as a key method to reduce model size and accelerate inference in LLMs, with approaches ranging from post-training quantization (PTQ) to quantization-aware training (QAT). Studies demonstrate that lower-bit quantization (e.g., 4-bit or below) challenges accuracy retention, but methods like mixed-precision quantization and low-rank adaptations effectively mitigate these losses. Quantization also enables hardware-friendly deployments on edge and mobile devices, balancing performance and resource constraints (Donisch et al., 2024; Saxena et al., 2024; Chen et al., 2024; Bondarenko et al., 2024; Liu et al., 2024).

2.5 Evaluation Metrics

There are multiple ways to evaluate the performance of an LLM for a specific task. Generally, LLM task evaluation can be grouped into benchmark-based evaluation and judgement-based evaluation, as shown in Figure 2.

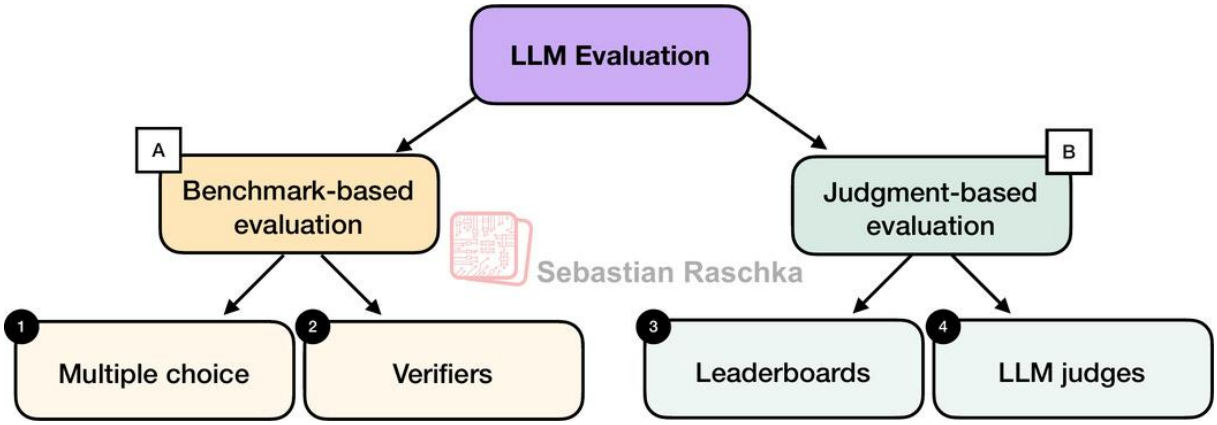


Figure 2: Overview of LLM evaluation methods.

2.5.1 Evaluation Frameworks

A number of general evaluation frameworks for LLMs exist, some of which are Confident AI's deepeval (Ip and Vongthongsri, 2026) which focuses on LLM-as-judge evaluations and also includes popular benchmark evaluation metrics, EleutherAI's evaluation harness (Sutawika et al., 2024) which is mostly benchmark-focused and includes a large number and variety of tasks in multiple languages including Arabic, while also being highly customizable with the support of custom tasks, and Hugging Face's lighteval (Habib et al., 2023) which currently supports benchmark-based evaluations only but with over 1000+ supported tasks.

2.6 Comparative Analysis

Comparative analyses from previous studies show that quantization and knowledge distillation generally outperform pruning in balancing accuracy and compression, with synergistic combinations yielding super- multiplicative benefits (Movva et al., 2022). Benchmarks like LLMCBench (Yang et al., 2024) provide valuable insights into compression method performance across various LLMs and tasks, facilitating informed method selection. Studies also highlight that larger models tend to better tolerate higher quantization ratios, suggesting that scaling laws also inform selection of compression strategies (Cao et al., 2025).

2.6.1 Combined Compression Strategies and Synergistic Effects

Multiple studies examine the joint application of quantization, pruning, and knowledge distillation, revealing super- multiplicative compression gains and improved trade-offs between model size and accuracy (Movva et al., 2022). Combined compression methods such as quantization with distillation (Polino et al., 2018) or pruning combined with quantization (Ma et al., 2023) demonstrate that integrated approaches may outperform individual techniques in terms of increased inference speed, decreased memory requirements, and retained accuracy. However, certain combinations may induce accuracy degradation, underscoring the need for careful method selection.

Chapter 3
Methodology

This section presents all aspects regarding the methodology of this research: The chosen models, model compression and optimization techniques, and model evaluation metrics.

3.1 The Chosen Models

In Table 2, the list of models chosen to perform the subsequent experiments on is shown. The models were chosen to cover a variety of architectures and sizes to generate a wider array of results and yield a more accurate analysis that differentiates between model-specific phenomena and structural effects. For models with pretrained and instruction-tuned variants such as Gemma and Llama, the instruction-tuned variant is used. In cases where the models are configured prior to experimentation, the applied configurations are mentioned.

Table 2: The models chosen for compression experiments.

Model	Arch.	Layers	Parameters	Configuration
ALLaM	Dense	32	7B	Base
Qwen3	Dense	36	8.2B	Base
Llama-3.1	Dense	32	8B	Base
Gemma 3	Dense	48	12B	Base
Qwen 2.5-Instruct	Dense	64	32B	NF4 quantization (BitsAndBytes, inference only)
Qwen2.5-Instruct+LoRA	Dense	32	5M (Trainable) 7.6B Total	r=16, α=32, q_proj + v_proj

3.2 Experimental Framework

A command-line experimental framework called `llmini`[5] was developed to facilitate the conduct of the compression experiments. The application takes a Hugging Face model identifier or the path to a local model as a required first argument and loads it using the `transformers` (Wolf et al., 2020) library. It also requires taking at least one flag that applies a compression method (e.g. `-prune, --quant`), and this compression method flag in turn takes as argument the specific compression algorithm from a set of defined options. Optionally, one can supply a configuration flag with a relevant list of comma-separated configurations, otherwise the default options apply. An overview of the application is illustrated in Figure 3.

[5] Code available at: https://github.com/S-Y-A-N/ar-llm-browser

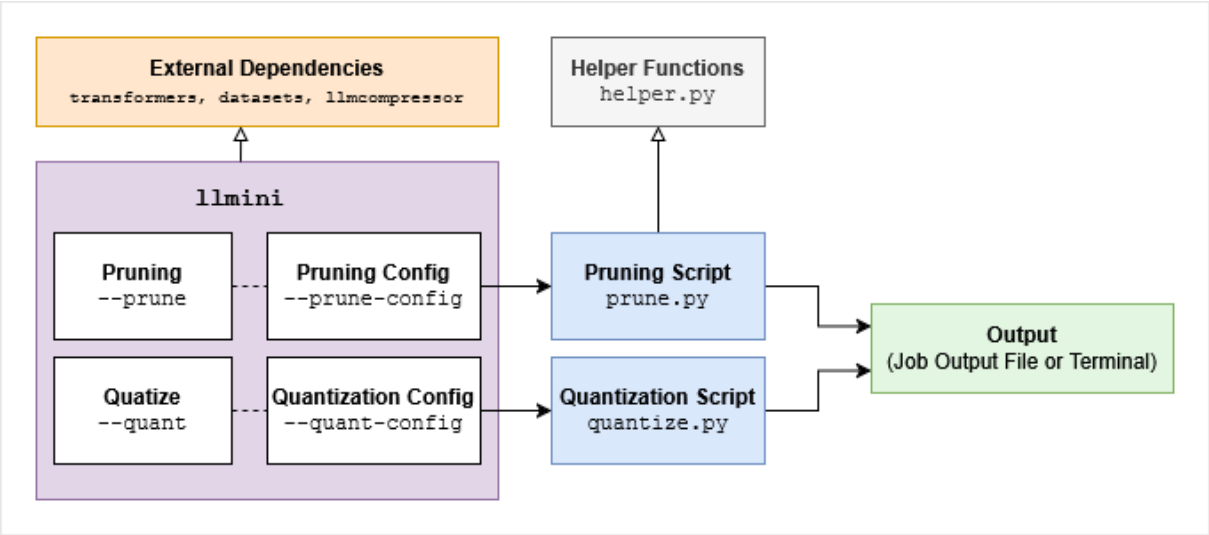


Figure 3: Overview of the experimental framework (llmini)

The diagram shown in Figure 3 follows the flow of the program at command execution with the shaded arrow, and the dependencies are denoted by the hollow arrow, and as of writing the llmini framework only supports model quantization and pruning. Knowledge distillation experiments followed a separate experimental framework, illustrated in Figure 4, and fine-tuning experiments followed the framework shown in Figure 5.

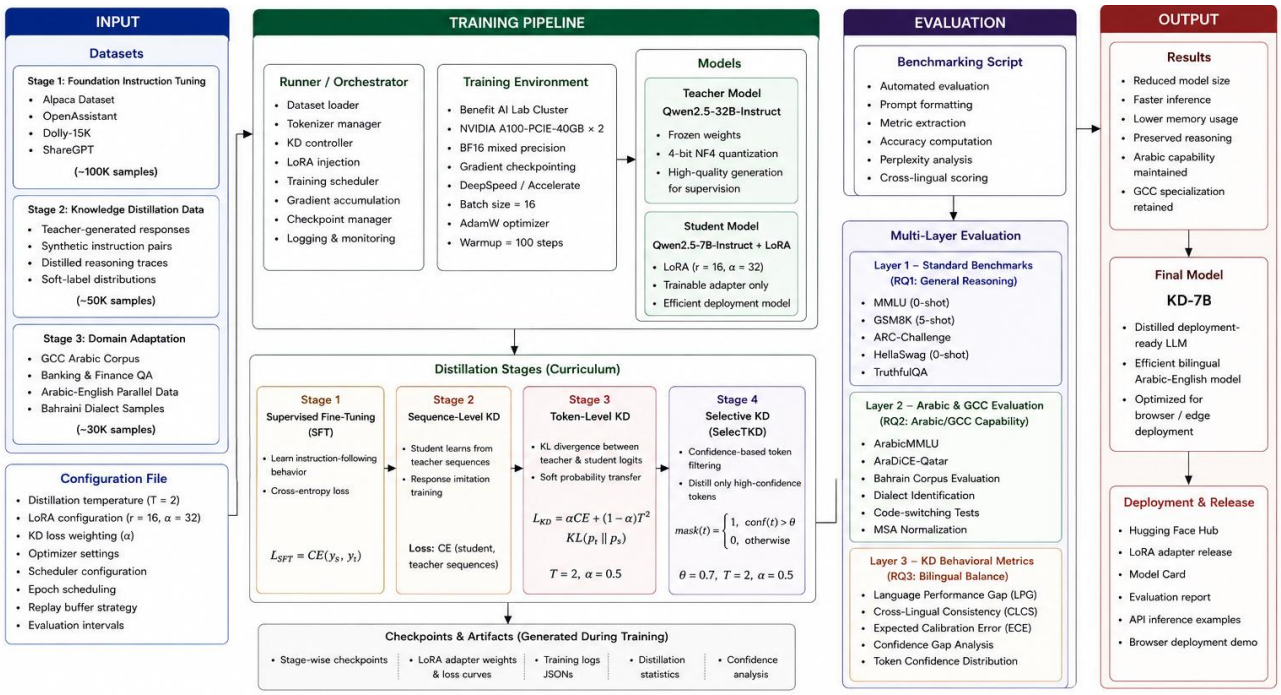


Figure 4: Knowledge distillation experimental framework.

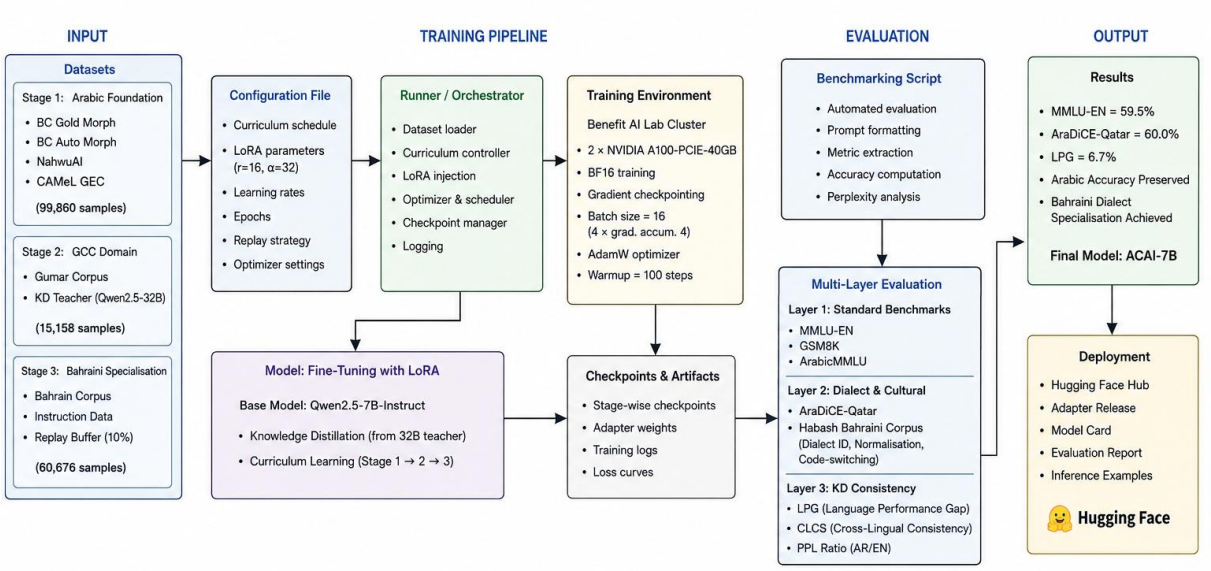


Figure 5: Fine-tuning experimental framework.

3.3 Model Compression and Optimization Techniques

3.3.1 Quantization

There are two main categories of quantization that are based on two different approaches: Quantization-Aware Training (QAT), and Post-Training Quantization (PTQ). Differentiating between the two is concerned with the need for retraining.

3.3.1.1 Post-Training Quantization (PTQ)

The mechanics of this method involve converting the pre-trained model from floating point format to the quantized format without needing retraining (Franter et al., 2023). Such method allows for more computational efficiency and a minimal calibration dataset for estimation of activation ranges (Xiao et al., 2023). Tackling the minimization of quantization errors is the main challenge of PTQ, and such is done by selecting clipping thresholds $[x_{min}, x_{max}]$. Common objectives include the Kullback-Leibler (KL) or the minimization of Mean Squared Error (MSE) to preserve the information content of the layer's output (Nagel et al. 2020).

Generative Post-Training Quantization (GPTQ)

GPTQ is a one-shot PTQ method which utilizes second-order information (especially the Hessian matrix) for minimization of the layer-wise reconstruction error (Franter et al., 2023).

$$\min_{\widehat{w}}(w - \widehat{w})^T H(w - \widehat{w})$$

Such that $H = \mathbb{E}$ is the Hessian matrix of the layer's inputs X (Franter et al., 2023). Negligible performance degradation can be acquired by GPTQ due to usage of Cholesky decomposition of the inverse Hessian to precompute update steps, thus quantizing massive models like OPT (175B) (Franter et al., 2023).

LLM.int8(): Outlier-Aware Mixed-Precision Decomposition

LLM.int8() identifies the "salient" feature dimensions (approx. 0.1% of all features) which then address outliers due to the exceedance of a fixed threshold (Dettmers et al. 2022). Mentioned "outliers" are then computed in high-precision $FP16$, unlike the remaining 99.9% which are turned into $INT8$ (Dettmers et al., 2022). Full performance for models up to 175B parameters is maintained by this mixed-precision approach while including the reduction to the memory requirements by 50% (Dettmers et al., 2022)

SmoothQuant

SmoothQuant is PTQ training-free solution which migrates the quantization difficulty from activations to weights using a mathematically equivalent transformation that results in smoothing the activation outliers (Xiao et al., 2023). A new per-channel factor (s_j) is introduced in SmoothQuant: $Y = (X \cdot S^{-1}) \cdot (S \cdot W)$, such that $S = \text{diag}(s_j)$ is a diagonal scaling matrix. The Equation: $s_j = \max(|X_j|)^{\alpha} / \max(|W_j|)^{1-\alpha}$ is used for calculation towards the mentioned smoothing factor where $\alpha = 0.5$ which balances the difficulty between activations and weights, allowing seamless $W8A8$ 8-bit weight and 8-bit activation quantization (Xiao et al., 2023).

3.3.1.2 Quantization-Aware Training (QAT)

QAT allows for model parameter rounding adaptation and clipping noise by integrating the quantization process directly into the training phase (Esser et al., 2020). Due to the non-differentiable nature of rounding functions, QAT utilizes Straight-Through Estimator (STE) that treats the quantization function as the identity during backpropagation (Jacob et al., 2018). Learned Step Size Quantization (LSQ) improves on this further by making the step size s a learnable parameter (Esser et al., 2020).

QLoRA: 4-bit NormalFloat Fine-Tuning

QLoRA quantizes the base model to 4-bit and uses Low-Rank Adapter (LoRA) for training, thus enabling efficient fine tuning (Dettmers et al., 2024). This method introduces the 4-bit NormalFloat (INF4) data type, that is optimized for the normally disturbed weights (Dettmers et al., 2024). QLoRA, in addition, employs Double Quantization which quantizes the quantization scales themselves to further reduce memory overhead (Dettmers et al., 2024).

Specialized Compression and Advanced Frameworks alongside best use cases include:

1. W4A16 (4-bit weights, 16-bit activations): minimization of latency and memory bandwidth (Dettmers et al., 2024)
2. W8A8 (8-bit weights, 8-bit activation): maximization of throughput in high-load server scenarios (Xiao et al., 2023)

3. AQLM (2-3 bit additive quantization): extreme compression for small memory devices (Egiazarian et al., 2024).

OmniQuant

OmniQuant uses block-wise error minimization to optimize the quantization parameters, in addition, it utilizes Learnable Weight Clipping (LWC) and Learnable Equivalent Transformation (LET) to make model more amenable to quantization without massive overhead of full QAT (Shao et al., 2024).

Additive Quantization of Language Models (AQLM)

AQLM represents groups of 8-16 weights as the sum of multiple vector codes from learned codebooks (Egiazarian et al. 2024). It is particularly effective for "extreme" compression (less than 3 bits per parameter), where it significantly outperforms standard scalar quantization methods like GPTQ (Egiazarian et al. 2024).

Integer-Only Inference (I-BERT)

For full hardware acceleration, models must transform non-linear operations into integer equivalent. I-BERT approximates transcendental functions like GELU and Softmax using quadratic polynomials (Kim et al. 2021). For example, the i-GELU approximation is defined as: $i\text{-GELU}(x) = \text{sgn}(x) \cdot (ax^2 + bx)$, where $a = -0.2888$ and $b = -1.769$ (Kim et al. 2021).

3.3.2 Pruning

Two methods of pruning were considered due to their general applicability on a wide array of model architectures and sizes, as well as their efficiency and accuracy. The two methods are *SparseGPT* (Frantar and Alistarh, 2023) and *Wanda* (Sun et al., 2024), which both utilize similar methods in the pruning criterion and mask selection, with Wanda considered a simplified approach of SparseGPT. The following sections provide a theoretical background for each method, and the specific implementation used for the two algorithms is discussed to ensure reproducibility of the results obtained in this report.

3.3.2.1 SparseGPT

The first method, *SparseGPT*, is a post-training, layer-wise pruning algorithm that uses a sparsity mask M to prune (zero-out) weights based on solving the weight reconstruction problem for every layer, then combining these layers together to obtain the compressed model. In this context, the mask M_i decides which weights to *keep* in row i, rather than which weights to prune, thus the compression is defined as:

$$compress\left(w^{(j)}\right)_i = 0 \; if \; j \notin M_i, otherwise \; w_{ij} \quad (1)$$

i.e., zero-out the weight if its column is not part of the mask, otherwise keep it.

The weight reconstruction problem is concerned with minimizing the **L_2 loss** between the output of an uncompressed model with weights $\mathbf{W}_\ell$ and that of the compressed one given an input $\mathbf{X}_\ell$. The problem is formulated where for each layer ℓ, a sparsity mask $\mathbf{M}_\ell$ of a specified sparsity and possibly updated weight $\hat{\mathbf{W}}_\ell$ must be found such that:

$$argmin_{M_\ell, \widehat{W_\ell}} \left\| W_l X_l - \left(M_l \odot \widehat{(W_l)}\right) X_l \right\|_2 \quad (2)$$

This method requires no retraining and simply uses a small set of calibration data as input to perform the weight updates. The solver for this problem calculates the optimal weights for a fixed sparsity mask $\mathbf{M}$ using the following formula:

$$W_{M_i} = \left(X_{M_i} X_{M_i}^\top\right)^{-1} X_{M_i} \left(W_{M_i} X_{M_i}\right)^\top \quad (3)$$

where $\boldsymbol{X_{M_i}}$ represents a subset of the input features whose corresponding weights, denoted $\boldsymbol{W_{M_i}}$, have not been pruned in row i.

However, this formula presents a computational problem because it requires inverting a Hessian matrix, denoted by $\boldsymbol{H_{M_i}} = \boldsymbol{X_{M_i} X_{M_i}^\top}$. Calculating $\boldsymbol{\left(H_{M_i}\right)^{-1}}$ scales cubically with the number of columns, taking $\boldsymbol{O\left(d_{col}^3\right)}$ time per row, for a total runtime of $\boldsymbol{O\left(d_{row} \cdot d_{col}^3\right)}$ per layer. For Transformer models, the runtime is roughly equivalent to $\boldsymbol{O\left(d_{hidden}^4\right)}$ for every layer, scaling by the hidden layer's dimension raised to the 4th power, which is computationally impractical and requires a time reduction of at least one factor for practicality. This problem stems from the fact that the Hessian computation is done for each row individually due to the row masks $\boldsymbol{M_i}$ being *different* and $\boldsymbol{\left(H_{M_i}\right)^{-1} \neq (H^{-1})_{M_i}}$, meaning that inverting a masked Hessian is not equivalent to masking an inverted Hessian. It is also worth noting that forcing the computation of a shared Hessian by restricting the row-masks to be the same would significantly impact the accuracy.

The SparseGPT paper presents an algorithm that overcomes the computational problem of the Hessian matrix inversion, achieving the goal of a full factor speedup for a running time of $\boldsymbol{O\left(d_{hidden}^3\right)}$, making it practical even for large Transformer models in the scale of hundred-billion parameters.

The key mechanism behind the weight reconstruction algorithm implemented by SparseGPT is to apply the Optimal Brain Surgeon (OBS) update (Hassibi et al., 1993) on only a subset $U \subseteq M$ of masked (unpruned) weights iteratively, which performs exactly optimal updates to the weights. The subsets U_j are decided recursively via:

$$U_{j+1} = U_j - \{j\} \; with \; U_1 = \{1, \ldots, d_{col}\} \quad (4)$$

where each subset U_{j+1} is created by removing the weight at the smallest index from the previous subset U_j. Thus, SparseGPT's weight reconstruction algorithm works by iterating through these subsets U_j and their corresponding inverse Hessians $\left(H_{U_j}\right)^{-1}$ in order and pruning w_j if $j \notin M_i$ for all rows i. Each inverse Hessian $\left(H_{U_j}\right)^{-1}$ only needs to be computed and is then reused to remove weight j in all rows where it is part of the pruning mask. This algorithm is one full factor more time efficient than simply inverting all the Hessians of Equation (3).

The final piece of the puzzle is the mask selection process, which directly impacts which weights are to be pruned. The mask is selected adaptively during weight reconstruction using *iterative blocking*, so that rather than selecting the mask for all columns in one go, the mask is selected iteratively per block, where the block size B_s represents the number of columns per block. This approach takes advantage of Hessian information and OBS weight updates to select the mask, thereby lowering the presence of uniformity in the mask, and thus preserving outlier features that are present in small amounts in large language models but with very large magnitudes (Dettmers et al., 2022a).

The criterion for selecting the weights per individual column j is magnitude since the diagonal inverse Hessian $(H^{-1})_{jj}$ is identical across the rows, and its formula is given by:

$$S_{ij} = \left[\frac{|W_{ij}|}{diag((X^T X + \lambda I)^{-1})} \right]_{ij} \quad (5)$$

Where the absolute value of the weight is divided by the corresponding value in the diagonal of the Hessian inverse, and λ being the Hessian dampening factor[6].

3.3.2.2 Wanda

The second method, Wanda (Sun et al., 2024), takes inspiration from SparseGPT (Frantar and Alistarh, 2023), but follows a simpler approach which further reduces the computational cost by one factor, down to $\boldsymbol{O\left(d_{hidden}^2\right)}$, while maintaining similar accuracy scores. One of the aspects that Wanda stands out in is that it does not perform any weight updates, and instead uses a small set of calibration data to calculate a simpler version of SparseGPT's pruning metric, given by the formula:

$$S_{ij} = |W_{ij}| \cdot \|X_j\|_2 \quad (6)$$

[6] The purpose of it is to prevent the collapse of the Hessian inverse computation (Sun et al., 2024). In the case of SparseGPT, it was shown to yield stable results for values of 0.001 to 0.1, and the value of 0.01 was chosen.

Where the absolute value of the weight is multiplied by the L2 norm of the corresponding input column to obtain the score of that weight. This metric is shown to greatly emphasize the importance of weight with low magnitudes if their corresponding features are of large magnitude. It is also very straightforward to compute compared to SparseGPT, and is also shown to be more robust when using lower calibrations samples (Sun et al., 2024).

3.3.2.3 Implementation

For SparseGPT and Wanda, the `llm-compressor` (Red Hat AI and vLLM Project, 2024) python library implementations were used for both methods at commit version #7182a55. The configurations of Wanda and SparseGPT's parameters are kept the same for fair comparison and follow the original papers of both, with all the main pruning experiments following a mask selection block size of 128, which was shown to perform near-optimally while minimizing the pruning process' memory requirements (Frantar and Alistarh, 2023). The calibration data used consists of 128 samples in segments of 2048-tokens, taken randomly from the first fragment of the C4 (Raffel et al., 2023) dataset.

3.3.3 Knowledge Distillation

In this part, we investigate whether Knowledge distillation preserves bilingual Arabic-English instructions-following capability when compressing Qwen2.5-32B-Instruct to Qwen-2.5-7B-Instruct+LoRA for Arabic browser deployment.

This Knowledge distillation approach compresses a large teacher model into smaller student model by transferring soft probability distribution 'dark knowledge' rather than hard labels, preserving reasoning patterns and bilingual behavior (Hinton et al.,2015) Then extended KD to a sequence-level generation (Kim and Rush, 2016). This method is relevant with bilingual LLMs, where compression must balance the dialect capability with English reasoning (Abdul-Mageed et al., 2021).

To preserve the dialectal capability further, fine-tuning on specific GCC and Bahraini dataset have been used in section 3.3.4 Fine-tuning**.**

3.3.3.1 Models and Training pipeline

For efficient deployment, **Qwen2.5-32B-Instruct** (32 billion parameters), was quantized using **4-bit Normal Float (NF4)** quantization for inference-only use (Qwen Team, 2024; Dettmers et al., 2023). NF4 Quantization maps continuous weight w to a lower-precision representation:

$$q = round\left(\frac{w - z}{\alpha}\right), \widehat{\boldsymbol{w}} = \boldsymbol{\alpha q} + \boldsymbol{z}$$

Where $\boldsymbol{\alpha}$ is a scaling factor, **q** is quantized value, and $\widehat{\boldsymbol{w}}$ is the reconstructed approximation. This method reduces the memory requirements from 64GB to ~20GB, enabling large model inference under constrained resources (Dettmers et al., 2023, p. 5).

While the student is **Qwen-2.5-7B-Instruct** with a total of 7.6 billion parameters, was fine-tuned using **Low Rank adoption (LoRA)** with rank=16 and $\alpha = 32$, targeting q_{proj} and v_{proj} (Hu et al., 2022). LoRA decomposes weight updates as:

$$W = W_0 + \frac{\alpha}{r} BA$$

Where $\boldsymbol{W_0}$ is the frozen pretrained weight, $\boldsymbol{A} \in \mathbb{R}^{d \times r}$ and $\boldsymbol{B} \in \mathbb{R}^{r \times k}$ are low-rank matrices trainable parameters: 5 million, 0.07% of total), and $\frac{\boldsymbol{\alpha}}{\boldsymbol{r}}$ is the scaling factor (Hu et al., 2022, p. 3).

The pipeline consists of four progressive stages: supervised fine-tunning (SFT), sequence-level KD(Seq-KD), token-level KD (Token-KD), and selective token KD(SelecTKD), and is shown in Table 3.

Table 3: KD - Training Pipeline

STAGE	LOSS FUNCTION	HYPERPARAMETERS	DESCRIPTION
SFT	$\mathcal{L}_{CE}(s_t, y_{teacher})$	$lr = 2 \times 10^{-5}$, epochs=3,batch=8	Supervised fine-tuning on teacher-generated responses
SEQ-KD	$\mathcal{L}_{CE}(s_t, \tau_t)$	$lr = 2 \times 10^{-5}$, epochs=3,batch=8	Sequence-level distillation using teacher-generated sequences
TOKEN-KD	$\alpha \times L_{CE} + (1-\alpha)\ T^2\ L_{KL}$	T=2,α=0.5,lr=1×10^{-5}	Token-level distillation from teacher distributions
SELECT-KD	Token-KD× $1(c_t > \theta)$	θ=0.7,T=2 ,α=0.5	Confidence-filtered token distillation

Note. s_t *denotes the student logits,* $y_{teacher}$ *denotes teacher responses,* τ_t *denotes teacher sequences, and* c_t *denotes teacher confidence at token position t. The SelecTKD mask is activated only when the teacher confidence exceeds the threshold* θ.

SelecTKD improves Token-KD, by filtering the KL divergence term so that only high confidence teacher tokens contribute to distillation. This design is motivated, as it observed during the training, there were a lower peak confidence for Arabic tokens compared to English tokens. For a token position t, teacher confidence is defined as

$$c_t = \max\left(softmax\left(\frac{z_t^{\ teacher}}{T} \right) \right)$$

Where $z_t^{\ teacher}$ denotes the teacher logits at position t and T represents temperature. The confidence mask is then defined as

$$m_t = \begin{cases} 1, & if\ \ c_t > 0.7 \\ 0, & otherwise \end{cases}$$

The SelecTKD objective therefore

$$\mathcal{L}_{SelecTKD} = \alpha \mathcal{L}_{CE}(s_t, y_{teacher}) + (1-\alpha)T^2 \times \sum_t KL(p_t^{student} || p_t^{teacher}) \times m_t$$

Where cross-entropy term is applied for all tokens, while KL-divergence is applied selectively to high-confidence tokens (Wu et al., 2025). Therefore, this formulation allows the student to continue the learning process with maintaining Arabic learning while reducing noisy distillation pressure.

Table 4: KD - Training Corpus

Source	Size	Language	Construction Method
opus-100 (Helsinki-NLP)	**2,428**	Arabic	Parallel corpus → 32B teacher responses (τ=2τ=2)
tatsu-lab/alpaca	**2,499**	English	Instruction → 32B teacher responses
GCC banking (generated)	10,000	AR+EN	Rule-based CBB regulations, banking, Bahraini dialect
Total	14,927	Bilingual	3 epochs, batch=8, LoRA

The corpus shown in Table 4 is utilized for all stages of the KD pipeline with LoRA-based student fine-tuning. Note that Teacher responses were generated using greedy decoding.

This corpus was chosen to reflect both general instructions following ability and domain-specific GCC usage. The use of Arabic parallel data supports cross-lingual adaption of models, whereas English Alpaca corpus would ensure the maintenance of general assistant behavior.

The generated GCC data would provide necessary information related to browser-based deployment in Bahrain and Gulf. Therefore, this resulting corpus supports the thesis goal of preserving bilingual capability under model compression.

3.3.3.2 Experimental Framework and Implementation

The distillation experimental framework[7] was implemented in Python as a modular pipeline covering data loading, preprocessing, model training, inference, and evaluation. The framework was executed on the Hayrat Benefit Lab cluster using two NVIDIA A100-PCIE-40GB GPUs, with experiments submitted through SLURM. The software environment was managed with Conda and included PyTorch for training, Hugging Face Transformers for model loading and inference, PEFT for LoRA-based fine-tuning, BitsAndBytes for quantized teacher inference, and lm-evaluation-harness for standard benchmark evaluation. The pipeline followed a staged design in which each stage was initialized from the previous checkpoint. Training used deterministic seeds and greedy decoding for evaluation to ensure reproducibility. Standard

[7] Code available at: https://github.com/Fatimadayan/arabic-kd-pipeline.git
Model Available at: https://huggingface.co/FFA37/Qwen2.5-7B-KD

benchmarks such as MMLU-EN were evaluated using lm-evaluation-harness, while Arabic- and dialect-specific benchmarks, including ArabicMMLU, AraDiCE, Habash corpus tasks, and bilingual behavioral metrics, were implemented with custom scoring scripts tailored to the thesis objectives.

3.3.4 Fine-tuning

Following knowledge distillation, the student model was further refined through parameter-efficient fine-tuning using LoRA (Hu et al., 2022). This choice is well suited to resource-constrained training environments and is commonly used in large language model adaptation workflows. We have fine-tuned the student model Qwen2.5-7B-Instruct[8] using three stage curriculum strategy instead of a single monolithic training run. The training follows a board to specific curriculum, Stage 1 focused on Arabic foundation learning, Stage 2 introduced GCC-domain adaptation, and Stage 3 targeted Bahraini dialect specializations while reducing catastrophic forgetting through replay from earlier stages (Aharoni and Goldberg, 2017; Kirkpatrick et al., 2017).

Table 18: Curriculum training stages: data sources, sample counts, and hyperparameter configuration[9]

Stage	Data Sources	Samples	Epochs	LR	Objective
Stage 1 Arabic Foundation	BC Gold Morph, BC Auto Morph, NahwuAI, CAMeL GEC	99,860	1	2×10^{-4}	General Arabic morphology, syntax, error correction
Stage 2 GCC Domain	Gumar Corpus, KD 32B teacher data	15,158	1	1×10^{-4}	Gulf Arabic domain adaptation via knowledge distillation
Stage 3 Bahraini Target	Bahrain Corpus (Habash et al., 2022), Hishambarakat, Stage 1 replay (10%)	**60,676**	2	5×10^{-5}	**Bahraini dialect specialisation with catastrophic forgetting mitigation**

The training setup used LoRA with rank r=16 and scaling factor α =32, applied to the attention and feed-forward projection layers. Training was conducted in bfloat16 precision using the AdamW optimizer with gradient checkpointing and stage-wise learning rate decay. Gradient accumulation was used to maintain the effective batch size , and warm-up period was performed at each stage to stabilize optimization. With this setup, the Benefit AI Lab GPU cluster could adapt steadily while maintaining a modest number of trainable parameters compared to the entire model size.

[8] Model available at: https://huggingface.co/FFA37/acai-7b-bahraini
Code available at: https://github.com/Fatimadayan/acai-bahraini-fine-tune

[9] **Note:** Stage 1 used Arabic foundation datasets including BC Gold Morph, BC Auto Morph, NahwuAI, and CAMeL GEC. Stage 2 used the Gumar Corpus, accessed via the CAMeL Lab annotated corpus download page. Stage 3 used the Bahrain Corpus, accessed via the official Bahrain Corpus website and the LREC 2022 corpus paper.

3.4 Evaluation Methods

The evaluation metrics used are divided into three groups: **Computational metrics**, **error measurement**, and **task benchmarks**. These evaluation categories were chosen to cover different aspects that would in turn produce a comprehensive analysis of the results.

3.4.1 Computational Metrics

The first group, **computational metrics**, takes into account the model size, its memory footprint, and its inference speed (tokens generated per second), and are used to evaluate how much a compression technique decreases the size of the model, and whether any speedup in token generation was gained.

The **model size** refers to the total number of parameters of a model multiplied by its element size in bytes. The **memory footprint** refers to the actual size of a model, in bytes, when loaded into memory. The **inference speed** or **throughput** refers to the number of tokens generated per second. It is important to note that the numbers obtained are approximations and may have slight variations depending on the hardware, bandwidth, and other factors, and are only meant to be used for relative comparison between the base and compressed models.

3.4.2 Error Measurement

The second group, **error measurement**, refers mainly to the perplexity metric, a widely used metric in model compression literature that measures the uncertainty of a model, used to perform simple performance comparisons between the results and the baseline. However, perplexity does not accurately measure the real-world performance of a model, and it also lacks the insight needed for results interpretability.

Perplexity (PPL) is a metric used to measure the language modeling capabilities of a model by measuring its uncertainty in predicting the next token. The models were evaluated on the WikiText-2 (Merity et al., 2016) corpus following HuggingFace's perplexity metric implementation (HuggingFace, 2025). For replicating results, the script used (`ppl.py`) can be found in the report's code repository[10].

3.4.3 Task Benchmarks

The third category, **task benchmarks**, covers the shortcomings of perplexity by selecting a set of diverse task-based benchmarks that evaluate real-world performance. The benchmarks were chosen to represent a variety of tasks in both the Arabic and English language.

[10] Code available at: https://github.com/S-Y-A-N/ar-llm-browser/blob/master/llmini/evaluation/ppl.py

The task benchmarks used are **ArabicMMLU** (Koto et al., 2024) for Arabic language performance, which comprises over 40 different tasks across 5 general categories, and **metabench** (Kipnis et al., 2025) for English language performance, a light version of six popular benchmarks: ARC (Clark et al., 2018), GSM8K (Cobbe et al., 2021), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021), TruthfulQA (Lin et al., 2022), and WinoGrande (Sakaguchi et al., 2019). The implementation uses EleutherAI's evaluation harness (Sutawika et al., 2024) at commit version `#c1c4bea` to ensure result replicability. All the evaluations were performed zero-shot, with most of the tasks reporting the raw accuracy scores, with the exception of reporting the normalized accuracy scores for ARC and HellaSwag, and flexible extraction scores for GSM8K for more stable results.

3.5 ACAI - Web Application Design and Architecture

We propose a prototype design of an offline Bahraini AI platform, which we call Arabic Cognitive AI or ACAI for short, that integrates an Arabic model specially fine-tuned on Bahraini datasets to better assist the local populace of Bahrain in their native tongue. The platform is an offline web application designed to test and showcase the compressed and optimized models in a practical scenario, as well as the capabilities of offline local hosting of LLMs. The application components are structured in a layered architecture, where each layer represents a separate component from which data moves into from the previous layer or component, as illustrated in Figure 6.

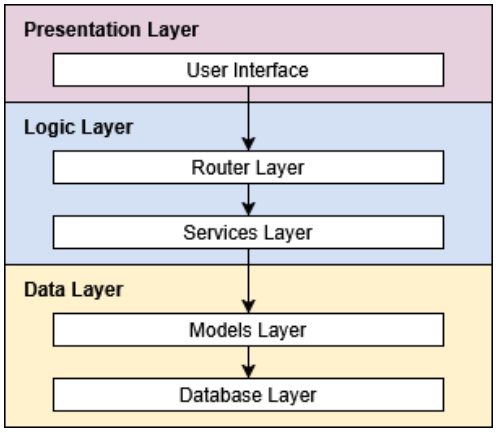


Figure 6: Application - Architecture overview.

The presentation layer handles the user interface, where the user can make requests, e.g. prompting a locally hosted LLM model. The logic layer is primarily separated into the router layer which handles routing logic, and the services layer which performs the business logic. The application will contain several LLM-related services that aid the generative abilities of the compressed models, such as orchestration, retrieval-augmented generation (RAG), web

search, and agentic pipeline building with intent classification (involving multiple chat modes that process a prompt differently based on their such as research, local law, dialect and language, reasoning, and knowledge extraction). Details about these chat modes, called agents, can be found in Table 5.

Table 5: Application - Overview of chat modes (agents).

Agent	Task	Example System Prompt
Bahith باحث (Researcher)	Use web search and RAG to respond with citations.	أنت باحث في ACAI .قدّم معلومات دقيقة. التنسيق: **الملخص:** (2-3 جمل)**النتائج الرئيسية:** نقاط + مصادر **التحليل:** سياق أعمق **الموثوقية:** X/10 لا تخترع مصادر. أجب بنفس لغة السؤال.
Musheer مشير (Advisor)	Use RAG with local policy and law documents to provide guidance.	أنت مشير، خبير أنظمة الخليج. مراجعك: CBB، DFSA، UAECB، SAMA. التنسيق: **الحكم:** [المنظم | الوثيقة | القسم] **التفاصيل:** شرح النظام **المتطلبات:** خطوات أو شروط هذا تحليل استرشادي. راجع متخصصاً قانونياً.
Lughawi لغوي (Linguist)	Analyze Arabic dialect and language.	أنت لغوي، خبير اللغة العربية واللهجات. **اللهجة:** [النوع] — الثقة: X% **المؤشرات:** الكلمات الدالة **الصرف:** كلمة = جذر + وزن + معنى (٣ كلمات) **الفصحى:** النص المطبوع **التحول اللغوي:** إن وجد **الثقافي:** ملاحظة
Hakeem حكيم (Reasoner)	Follow reasoning steps and present a confidence score of the reasoning.	أنت حكيم، عميل التفكير العميق. **خطوة ١:** التفكيك **خطوة ٢:** المعرفة **خطوة ٣:** الاستدلال **خطوة ٤:** التحقق **خطوة ٥:** الإجابة النهائية **الثقة:** X/10
Muraqib مراقب (Verifier)	Verify claims with evidence.	أنت مراقب، عميل التحقق. ☑ **صحيح:** الدليل ⚠ **غير محدد:** يحتاج مصدراً ✗ **خاطئ:** التصحيح **الحكم:** X/10

Bani باني (Knowledge Builder)	Extract meaningful knowledge from model responses.	أنت بانِ، عميل استخراج المعرفة. **الكيانات:**	الاسم	النوع	الثقة | **العلاقات:** → [أ] —[علاقة]→ [ب] **المفاهيم:** م١، م٢، م٣

The models layer is concerned with the data models that form the database of the application. Figure 7 shows a simplified diagram of the application's data models and their relationships. The application allows having multiple users or using it as a guest. Each user can have multiple chats, and each chat contains queries (prompts or messages) made by the user. Naturally, each query has one response related to it, and responses are created by an "agent" using the query. Each agent defined uses an underlying LLM to generate responses.

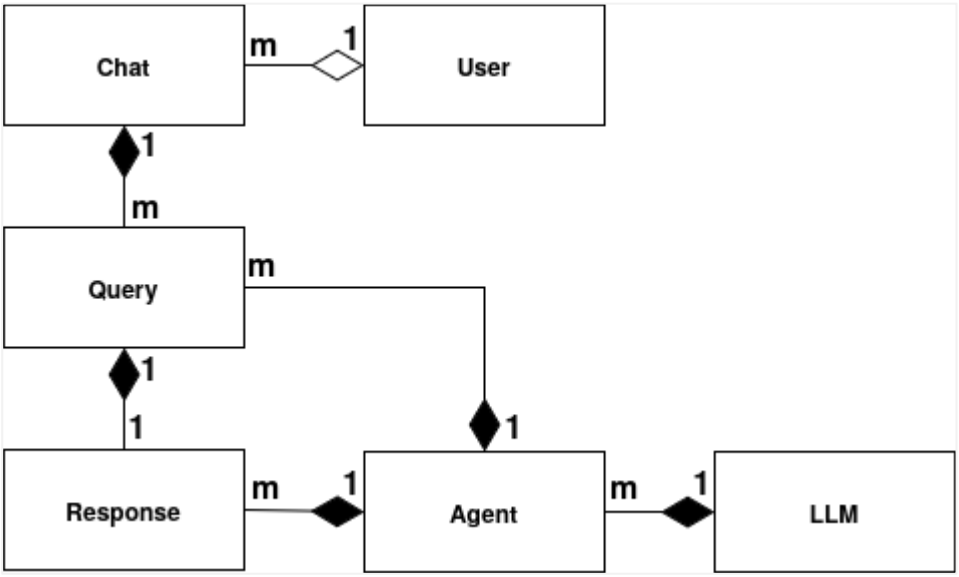


Figure 7: Application - Data models diagram

The unshaded diamond denotes an aggregation, and the shaded one a composition.

Chapter 4
Results and Discussion

This chapter discusses the results obtained from the base models, considered the baseline, and the compressed models' evaluations. The experimental environment and setups are first discussed, followed by sections dedicated to each compression method that discuss the results obtained from the compressed models in relation to the baseline. A final discussion section is dedicated to comparing between the compression methods and insights.

4.1 Experimental Environment and Setup

All the experiments were conducted on the University of Bahrain BENEFIT Advanced AI and Computing Lab's GPU cluster, using two 40GB NVIDIA A100 (Ampere 8.0) GPUs for compute-heavy experiments, and single 16GB T4 Tesla (Turing 7.5) GPU for light experiments. The workflow used our utility `llmini` using libraries such as Hugging Face's Transformers (Wolf et al., 2020) for model loading, PEFT (Mangrulkar et al., 2026) and TRL (von Werra et al., 2020) for fine-tuning, BitsAndBytes (Dettmers et al., 2022b) for quantized inference and other. All perplexity and task evaluations required the use of the more powerful A100 GPUs due to the data overhead. The specific setup for applying each of the compression methods' is provided in the following sections.

4.1.2 Quantization Setup

GPTQ was shown to have greater memory requirements that necessitated using the A100 GPU, and the weight-only GPTQ 8-bit quantization (W8A16) required the most compute.

4.1.2 Pruning Setup

Since the pruning methods used (SparseGPT and Wanda) are not computationally expensive, a single T4 GPU was sufficient for applying the pruning methods on the models.

4.1.3 Knowledge Distillation Setup

Knowledge distillation was used to adapt the teacher model Qwen2.5-32B to smaller student model Qwen2.5-7B efficiently. The teacher model-32B was quantized using BitsAndBytes compressing the memory needed from 64GB to approximately 20GB, which is suitable for the lab cluster GPU. In addition, this is the technique used in previous research as in (Dettmers et al., 2022b) which established that 4-bit quantization via (BitsAndBytes) is sufficient for inference and low-rank adoption. Also (Chhawri, et al., 2025) ,which investigate why 4-bit

quantization is often the "sweet spot" for these models. In other hand, the student model Qwen2.5-7B was trained progressively through supervised fine-tuning to sequence-level distillation, token-level distillation and the proposed SelecTKD stage.

This setup was chosen because it allows the study to isolate the effect of each stage on performance and bilingual behavior, however its suitable for the purpose of resources-efficient adaption.

4.2 Quantization

The Arabic model ALLaM-7B was quantized into two weight precisions, 8-bit and 4-bit, using several quantization algorithms, such as LLM.int8() (**Int8**), weight-only GPTQ (**W8A16**), and weight-and-activation GPTQ with SmoothQuant (**W8A8**) for the 8-bit algorithms, and QLoRA (**Int4**) and weight-only GPTQ (**W4A16**) for the 4-bit algorithms. The results compare baseline ALLaM-7B (**FP16**) with the post-quantization model sizes, perplexity, and task performance scores. For disambiguation, in the following sections 4-bit QLoRA refers to the quantization method introduced in the same paper (Dettmers et al., 2023), not the fine-tuning technique.

4.2.1 Compression Ratio

The baseline float16 ALLaM-7B model alongside its quantized counterparts are shown in Table 6, and the compression ratio (Original Size / Compressed Size) is calculated. It is expectedly found that 8-bit quantization methods decrease the model size by around half, and precisely by a factor of 1.86. On the other hand, the 4-bit GPTQ decreases the model size the most, even more than QLoRA's quantization. This is expected due to QLoRA's double quantization method being designed for fine-tuning and adding special adapters for that purpose, called trainable Low-Rank Adapters (LoRA).

Table 6: Quantization - Model sizes and compression ratios

ALLaM-7B	Model Size (GB)	CR
Fp16	13.04	1×
Int8	7.01	1.86×
W8A8	7.01	1.86×
W8A16	7.01	1.86×
Int4	4.37	2.98×
W4A16	4.09	**3.19×**

4.2.2 Perplexity

The perplexity values for each quantization method when applied on ALLaM-7B are reported in Table 7. From the reported results, it is observed that 8-bit quantized models perform

similarly to the base float16 model, with weight-only 8-bit quantization (W8A16) performing slightly better by 0.01 point. This suggests that, for ALLaM-7B in particular, the effect of 8-bit quantization is minimal on the model's quality, while providing a proportionally large reduction in the model's size, shown to be about 1.86× reduction in Table 6. On the other hand, the 4-bit quantization results were both higher and different from each other. For instance, the GPTQ weight-only 4-bit quantization yielded lower (better) perplexity compared to the QLoRA quantization. This can be explained by the fact that QLoRA-quantized models are expected to be subsequently fine-tuned with the fine-tuning method QLoRA, and in this case no fine-tuning took place, and inversely, GPTQ is a post-training quantization algorithm, and thus the better perplexity value without subsequent training or fine-tuning.

Table 7: Quantization - Perplexity scores

ALLaM-7B	Perplexity
Fp16	4.75
Int8	4.80
W8A8	4.86
W8A16	**4.74**
Int4	6.37
W4A16	**5.21**

4.2.3 Task Performance

The mean accuracy scores of ArabicMMLU and metabench for ALLaM-7B base float16 and quantized versions are shown in Table 8. Additionally, more detailed observations can be drawn from the per-task results shown in Table 20 in Appendix A.

Table 8: Quantization - Mean accuracy scores for ArabicMMLU and metabench

ALLaM-7B	ArabicMMLU	metabench
Fp16	70.08	55.10
Int8	69.84	53.34
W8A8	69.73	**55.70**
W8A16	**70.16**	55.24
Int4	65.67	40.73
W4A16	**66.78**	**50.06**

Accuracy Retention. Weight-only **8-bit** quantization with GPTQ not only retained the full accuracy from the base float16 ALLaM-7B, but also marginally exceeded the original performance by approximately +0.11% and +0.25% on English and Arabic tasks respectively.

Moreover, the GPTQ 8-bit weight-and-activation quantization has shown remarkable performance with almost 99.5% retained accuracy in Arabic tasks and +1.1% accuracy increase in English tasks. The 8-bit quantized model with LLM.int8() also follows closely to the 8-bit GPTQ-quantized models, with almost 99.7% retained accuracy in Arabic tasks and 96.8% in English tasks. These scores show that GPTQ quantization, whether or not weight-only, is a highly accurate method of reducing weights while preserving performance on real-world tasks, with LLM.int8() being nearly as accurate while also taking virtually no time to apply unlike GPTQ. Since the increases present are marginal, they are not necessarily indicative of actual performance gains in real-world application, but they do suggest that the model likely retains a similar performance to the original model. The ALLaM-7B is also a fine-tuned Arabic model, and was shown in the pruning results to be more stable with compression in comparison to the other models.

For the **4-bit** quantized models, both the GPTQ and QLoRA variants retained a similar accuracy in Arabic tasks, about 95.29% and 93.71% mean accuracy retention. However, a starker difference is found in the English tasks' mean accuracy, with GPTQ retaining 90.85% of the original accuracy, in contrast to QLoRA retaining 73.92%. This wide margin suggests the same conclusion reached from the perplexity scores, in that GPTQ is a more complex quantization method and primarily a post-training technique, while QLoRA's quantization method is expected to be followed by fine-tuning, and likely thus the stark difference.

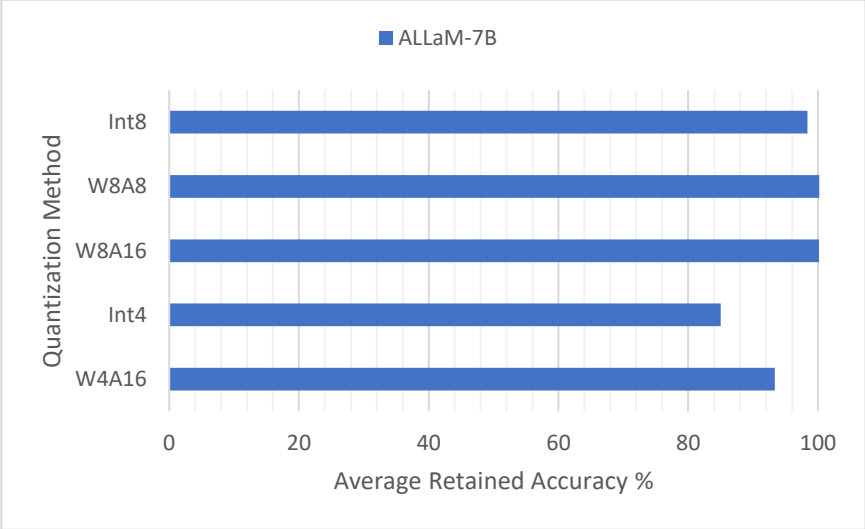


Figure 8: Quantization – Average retained accuracies per method

Compression-Accuracy Tradeoff. The average accuracy retention is calculated per quantization method for ALLaM-7B to analyze the tradeoff between compression and accuracy, and is shown in Figure 8. All the 8-bit quantization methods had the same compression ratio at about 1.86, making the tradeoff comparison between them straightforward,

with the method achieving the highest accuracy retention being the best. This is shown to be both the GPTQ quantized models, which are closely tied at ~100.2% accuracy retention, with LLM.int8() being very close as well at ~98.4% retention. On the other hand, it is clear that the 4-bit GPTQ quantized models also performs competitively with ~93.3% accuracy retention with the highest compression ratio (3.19×) of all quantized methods. QLoRA comparatively falls behind with ~85.0% retained accuracy at 2.98× compression.

Additionally, to better gauge the difference between 8-bit and 4-bit methods, the average accuracy of the three 8-bit quantization methods is compared with specifically GPTQ due its superior results. It is found that 4-bit GPTQ achieves about 95.52% of the average 8-bit quantization accuracy in Arabic tasks, and 90.85% in Arabic tasks, while decreasing the storage and memory requirements of the model a factor of 1.33 more than the 8-bit methods.

4.3 Pruning

The pruning methods, SparseGPT and Wanda, were applied on a subset of the models shown in Table 2, with the main results following a fixed sparsity level of 50% with unstructured pruning. The results compare the baseline models with the post-pruning model sizes, perplexity, and task performance scores.

4.3.1 Compression Ratio

The baseline model sizes alongside their post-pruning counterpart are shown in Table 9, and the compression ratio (Original Size / Compressed Size) is calculated. With a fixed sparsity level of 50%, the results expectedly follow that and show a similar compression ratio across all models, for an average of 1.79× compression. These compression ratios are shared between SparseGPT and Wanda since the difference in the number of pruned parameters is negligible. For parameter counts per pruning method, the reader may refer to Table 21 in Appendix A.

Table 9: Pruning - Original vs. pruned size of the model and the compression ratios.

	Original Size	Pruned Size	CR
ALLaM-7B	13.04	7.01	**1.86×**
Qwen3-8B	15.26	8.79	**1.74×**
Llama3.1-8B	14.96	8.46	**1.76×**
Gemma3-12B	22.70	12.68	**1.79×**

4.3.2 Perplexity

The perplexity scores are reported in Table 10 for all the dense and sparse variants of the models. From the reported results, it is observed that SparseGPT produces lower (better)

perplexity scores than Wanda. However, the amount of this difference varies by model, with ALLaM-7B having the lowest difference between its SparseGPT and Wanda variant (0.34), closely followed by Qwen3-8B (0.44), then by Llama3.1-8B (1.1), and finally by Gemma3-12B with the highest difference (2.7).

Table 10: Pruning - Perplexity scores for baseline and SparseGPT/Wanda-compressed models.

Method	Sparsity	ALLaM (7B)	Qwen3 (8B)	Llama-3.1 (8B)	Gemma 3 (12B)
Dense	0%	4.75	7.27	5.89	7.53
SparseGPT	50%	**6.36**	**8.99**	**8.55**	**9.28**
Wanda	50%	6.70	9.43	9.65	11.98

Comparing between the dense baseline and SparseGPT variant for each model, ALLaM-7B is found to have the lowest increase in perplexity (+1.61), followed by Qwen3-8B (+1.72), then by Gemma3-12B (+1.75), and finally by Llama3.1-8B (+2.66). Similarly for the Wanda variants, ALLaM had the lowest perplexity increase (+1.95) then followed by Qwen3-8B (+2.16), with the difference being that Llama3.1-8B had a lower increase compared to Gemma3-12B (+3.76 and +4.45 respectively). These results are visualized in Figure 9.

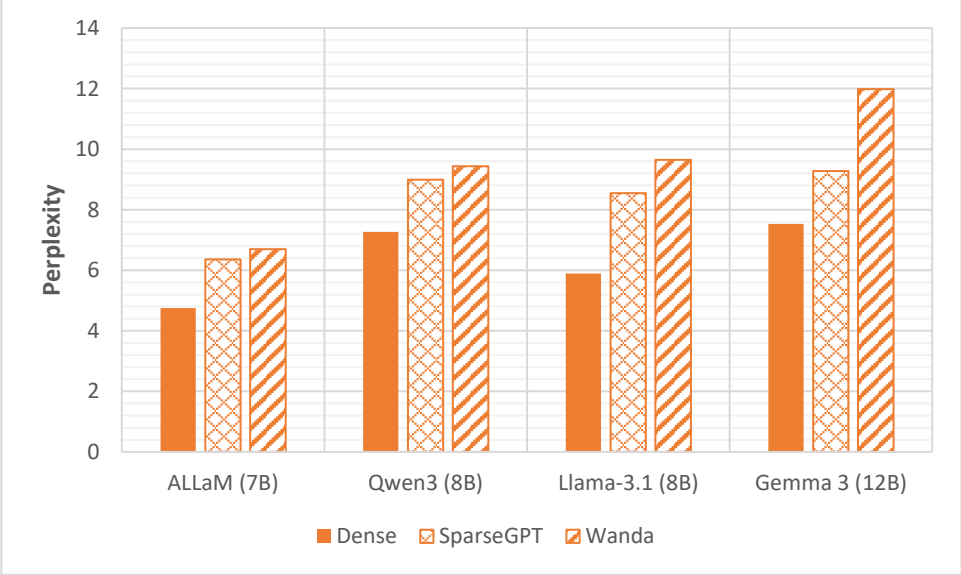


Figure 9: Pruning - Perplexity scores comparison between dense and sparse variants.

4.3.3 Task Performance

The pruning methods, SparseGPT and Wanda, were applied on the models shown in Table 2, with the main results following a fixed sparsity level of 50% with unstructured pruning.

Table 11 reports the mean accuracy scores of ArabicMMLU (denoted AR for Arabic) and metabench (denoted EN for English) for the dense baselines and the sparse variants.

Additionally, interesting observations can be drawn from the detailed per-task results for ArabicMMLU and metabench which are shown in Table 22 in Appendix A.

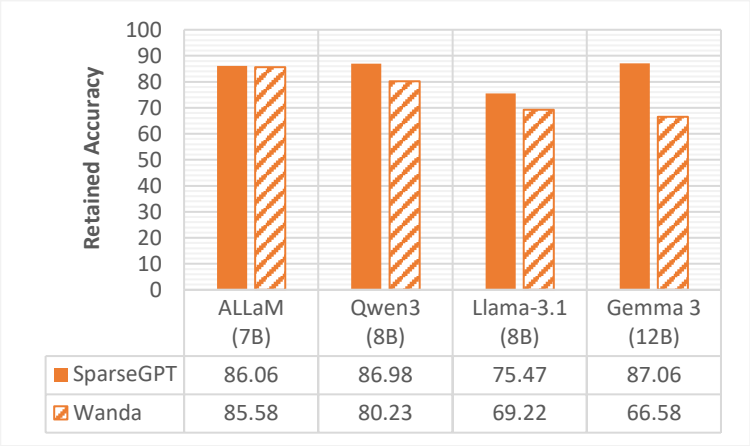
Table 11: Pruning - Mean accuracy scores of ArabicMMLU and metabench.

		ALLaM 7B		Qwen3 8B		Llama-3.1 8B		Gemma 3 12B	
Method	**Sparsity**	**AR**	**EN**	**AR**	**EN**	**AR**	**EN**	**AR**	**EN**
Dense	**0%**	70.08	55.10	62.38	57.38	56.69	56.61	65.04	64.14
SparseGPT	**50%**	62.62	**45.60**	**53.70**	**50.42**	**40.05**	**45.45**	**58.17**	**54.32**
Wanda	**50%**	**63.58**	44.33	50.52	45.60	36.52	41.91	47.00	39.06

Accuracy Retention. SparseGPT has shown to have a slightly higher accuracy retention rate, keeping an average of 83.9% of the original model's accuracy for both the English and Arabic tasks, and Wanda keeping comparatively an average of 75.4%. While there was no difference between the averaged performance of SparseGPT-compressed models between Arabic and English tasks (with ~83.9% accuracy retention in both), Wanda performed slightly better in Arabic tasks (77.1%) compared to English tasks (73.7%). This difference could be attributed to the imbalance in the evaluation metrics between the two languages, as ArabicMMLU mainly measures the knowledge of the model using choice-based question-answering, while metabench has more diverse metrics that can be more affected by compression, like the generative task GSM8K that measure multi-step mathematical reasoning, and HellaSwag which measure common-sense reasoning.

Points Difference. In terms of the raw score differences, SparseGPT scores higher (in comparison to Wanda) an average of ~3.36 points in the ArabicMMLU (AR) benchmark, and ~3.21 points in the metabench (EN) benchmark, with the exception of ALLaM's ArabicMMLU score, which is slightly higher in the Wanda variant both on average and on a per-task basis, as seen in Table 21. Theoretically, since Wanda has a simpler pruning metric and does not perform weight updates in contrast to SparseGPT, it is expected that its average model performance will be lower. This holds true in the conducted experiments, but these differences can in times be marginal, while in other times Wanda can perform remarkably better, as shown in the GSM8K score of Qwen3, which is significantly higher by 18.57 points compared to SparseGPT.

Compression-Accuracy Tradeoff. Since the compression ratios are largely similar across models, with a ratio of 1.79× on average, the average accuracy retention is calculated per pruned model to analyze the tradeoff between compression and accuracy. From Figure 10, we can aid



	ALLaM (7B)	Qwen3 (8B)	Llama-3.1 (8B)	Gemma 3 (12B)
SparseGPT	86.06	86.98	75.47	87.06
Wanda	85.58	80.23	69.22	66.58

Figure 10: Pruning - Accuracy retention per model.

the tradeoff analysis with **two** main observations.

Firstly, the **SparseGPT** variants of ALLaM-7B, Qwen3-8B, and Gemma3-12B are shown to have the highest retention rates, with a very similar retained accuracy of 86.06, 86.98, and 87.06 respectively. This suggests that these three models have a similar compression-accuracy tradeoff, which is the highest among all the pruned models. In comparison, Llama3.1-8B's had a significantly worse retention of 75.47.

Secondly, the **Wanda** variant of ALLaM-7B had the highest retention rate of 85.58, closely followed by Qwen3-8B (80.23), then distantly by Llama-3.1 (69.22) and Gemma3-12B (66.58). Combining these observations, the Wanda variants are found to differ in their tradeoff compared to the SparseGPT variants in **three** levels: ALLaM-7B with the **lowest** difference (0.48), Qwen3-8B and Llama3.1-8B with **moderate** difference (6.75 and 6.25 respectively), and Gemma3-12B with the **highest** difference (20.48). This variety of differences among the two pruned variants suggest that the resulting performance may be architecture-dependent, since Gemma3-12B has a more unique architecture compared to the other three, and a much higher number of layers. Interestingly, these findings show that models with larger sizes and more layers tend to have a worse tradeoff in Wanda compared to SparseGPT, with the difference diminishing for smaller models, but drawing this conclusion requires a larger number of observations of many models with varying architectures and sizes.

Models Comparison. To further compare between the sparse models, the performance gap between the two pruning methods is measured per model by calculating the difference in their accuracy retention scores, with smaller performance gaps indicating a higher similarity between

the two pruned variants. It is found that the Arabic-trained ALLaM-7B, despite being the smallest model, had the most stable results, with the highest similarity in results between its SparseGPT and Wanda variants in both English and Arabic tasks, with remarkable robustness in Arabic specifically that marginally favored Wanda. On the other hand, while Qwen3-8B has significantly higher accuracy retention than Llama-8B, they both have a similarly moderate performance gap between the two methods that favors SparseGPT, barring the exceptionally better performance of Qwen3-8B's Wanda variant on GSM8K. Lastly, Gemma3-12B had the most erratic difference between its SparseGPT and Wanda variants, with clearly worse performance in its Wanda version compared to its SparseGPT version.

Overall, ALLaM-7B had the best post-pruning (w/ Wanda) scores on the Arabic benchmark and by a large margin compared to the other models. For the English benchmark, Gemma3-12B had the best post-pruning (w/ SparseGPT) scores overall with Qwen3-8B performing competitively, with about 7.8% worse performance in English and 8.3% worse in Arabic on average, while being 44% smaller in size.

4.4 Knowledge Distillation

The experimental setup follows the multi-stage training pipeline described in Chapter 3, consisting of supervised fine-tuning, sequence-level distillation, token-level distillation, and the proposed confidence-filtered distillation method, SelecTKD. The evaluation framework is organized into three complementary layers: (1) general reasoning (MMLU-EN, ArabicMMLU,GSM8K via lm-evaluation-harness v0.4.11); (2) Arabic /GCC capability (AraDiCE-Culture Qatar, Habash Bahrain Corpus with CJK filtering); (3) Bilingual behavioral symmetry (xx-pair prompts measuring LPG = |ACC_EN - ACC_AR|, CLCS, ECE_AR) (Gao et al., 2023; Al-Ajmi et al., 2022).Additionally, perplexity is discussed as additional linguistic quality metric to assist in analyzing efficiency and frequency trade-off.

4.4.1 Task Performance

This section evaluates whether the distillation process retains the ability of general reasoning of the student model, while increasing its performance or at least not affecting its Arabic and GCC specific behavior.

4.4.1.1 General Reasoning Benchmark

The first layer of evaluation examines whether KD affects broad reasoning capability in both English and Arabic. Knowledge distillation preserves English reasoning without significant statistical degradation. Table 12 shows that English MMLU scores are comparable within errors bars (73.80 ± 0.60 base vs. 73.40 ± 0.60 SelecTKD), while ArabicMMLU (General Knowledge

task with 100 samples) scores remain stable at a range of 58 to 63% across stages, showing no systematic loss. In addition, GSM8K scores varied based on corpus composition, not distillation quality. Since GSM8K follows a different pattern, starting by supervised finetuning which improves the performance reaching from the base model 18 to 52, then later distillation reduces this gain reaching 26. However, it is still better than the base model, with raw untrained nor distilled data.

Table 12: KD - Layer 1 - Standard Benchmarks

Model	English-MMLU	ArabicMMLU	GSM8K
Base-7B	73.8 ± 0.60	63.0	18.0
sft-KD-7B	-	61.0	52.0
Seq-KD-7B	-	60.0	32.0
Token-KD	73.5 ± 0.60	63.0	18.0
SelecTKD	73.4 ± 0.60	58.0	26.0

4.4.1.2 Arabic and GCC-specific Evaluation

The second evaluation layer focuses on Arabic and GCC-specific capability using the AraDiCE-Culture (Qatar subset) benchmark. AraDiCE was introduced as benchmark for dialect and cultural evaluation in Arabic LLMs, its especially important for testing if compressing Arabic Large Language model preserves grounded knowledge (Mousi et al., 2025). In this thesis, the Qatar specific subset are employed as concise but meaningful for testing GCC culture understanding. As shown in Table 13, there is a noticeable difference between all the models. Where in the first result using 30 MCQ the percentage fluctuate between the models Base achieved 66.7 while the accuracy drops by 3.4% in Token-KD, then reduces after filtering to with a 13.4% drop in accuracy falling to 53.3% in SelecTKD.

Then we evaluate the model with Truthful modern standard Arabic (MSA), Physical Interaction Question Answering (PIQA) MSA, and HellaSwag MSA, with N=500 results are more stable across the models. The performance across all the three models remains clustered around 30s to mid-40s.showing the slightly impact of KD on the inherent tendency of the model to hallucinate or to the limited effect in Modern Standard Arabic reasoning. Although SelecTKD again shows a modest decline relative to the base model.

Table 13:KD - AraDiCE-Culture Qatar and GCC Dialect

Model	AraDiCE-Qatar (N=30)	Truthful MSA (N=500)	PIQA MSA (N=500)	HellaSwag MSA
Base-7B	66.7	42.4	45.2	40.2
Token-KD	63.3	43.1	44.4	40.2
SelecTKD	53.3	41.8	31.4	38.0

The Bahrain Corpus Evaluation

The Bahrain corpus (Abdulrahim et al., 2022) provides a more discriminative test of Bahraini dialect capability because it contains 300 prompt-level evaluation dialect identification, normalization, and code-switching as illustrated in Table 14. Due to keyword negation inflation, scores should be reported as upper bound to avoid overstating results, with manually checked subset used to confirm the direction of the findings. The main pattern is that Token-KD preserves dialect performance better than SelecTKD, while SelecTKD improves the bilingual balance at the cost of dialect specialization.

Table 14: KD - Habash Corpus Evaluation

(100 authentic Bahraini Sentences/task, CJK-filtered, UPPER BOUNDS)

Model	Dialect ID% ↑	Normalize%↑	Code-Switch% ↑	Overall%↑	CJK Failures
Base-7B	94.0	68.0	71.0	77.7	18/300 (6%)
Token-KD	95.0	70.0	68.0	77.7	21/300 (7%)
SelecTKD	80.0	53.0	79.0	70.0	22/300 (7.3%)

4.4.1.3 KD behavioral Metrices

The third evaluation layer measures bilingual behavioral symmetry using paired Arabic and English prompts. The importance in this layer represent in testing the model consistently across languages underlying the task is semantically same. This goes beyond traditional benchmark accuracy and asks whether the distillation method preserves the balance between English and Arabic capability.

From Table 15, the key finding is that standard Token-KD does not reduce the language performance gap. Both the base model and Token-KD obtain LPG of 23.4, which indicates that uniform Token-KD level distillation preserves the original imbalances rather than correcting it. In contrast, SelecTKD has an LPG of 0 in the 30-paird evaluation, However it has strong performance in CLCS and Arabic calibration. Hence, this is strong evidence that supports the proposed method, has the ability to change the distribution of learning in a way that has a direct impact on bilingual outcomes.

Table 15: KD behavioral metrices: 30-pair parallel evaluation, greedy decoding

Model	EN Accuracy	AR Accuracy	LPG	CLCS	ECE-AR	Confidence Gap
Base-7B	86.7	63.3	23.4	60.0	19.78	7.8
sft-KD-7B	86.7	80.0	6.7	73.3	3.60	6.5
Seq-KD-7B	83.3	76.7	6.6	66.7	16.16	8.6
Token-KD	86.7	63.3	23.4	60.0	19.53	8.1
SelecTKD	86.7	86.7	0.0	80.0	5.24	7.9

4.4.1.4 Token-level fluency

Perplexity is included as a supplementary measure to assess whether the different distillation stages affect token-level fluency. The result in Table 16 show that Token-KD and the baseline model are nearly identical, while SelecTKD produces a slightly higher perplexity value. This indicates that the cost of applying confidence filtering is small in terms of fluency ,although the increase is limited and should be interpreted in context rather than major degradation.

Table 16: KD – Perplexity scores

Model	Perplexity	Average NLL
Base-7B	5.19	1.6465
Token-KD	**5.20**	**1.6495**
SelecTKD	5.34	1.6751

Perplexity is a token-level measure of uncertainty, and not to be confused with a complete quality measure. However it is still useful as a supporting diagnostic, particularly when combined with reasoning and behavioral metrics. From the scores shown in Table 16, it is seen that the slight increase in perplexity for SelecTKD is balanced by its strong gains in bilingual symmetry, which indicates a targeted trade-off rather than a general collapse in language modelling ability.

4.4.2 Reproducibility and Ablation

To confirm the robustness of the behavioral results, seed reproducibility analysis was conducting using 3 different seeds as shown in Table 17. Due to the use of greedy decoding for the evaluation process, the output is deterministic, and that the standard deviation across seeds is zero. This aspect is useful in a thesis environment because it confirms that the results reported are reproducible. A threshold ablation was also performed to measure the sensitivity of SelecTKD to the confidence cutoff, shown in Table 18. The operating point was selected to be 0.7, which preserved the strongest bilingual improvement while still maintaining enough training signal for the model to learn effectively. Lower thresholds introduced too much noisy supervision, while higher thresholds removed too much signal. This supports the claim that SelecTKD is not a generic filtering rule, instead it is a carefully chosen balance between signal quality and signal quantity.

Table 17: KD - Seed reproducibility (n=3 seeds, 15-pair evaluation, greedy decoding)

Model	EN mean	AR mean	LPG mean
Base-7B	93.3	66.7	26.7
Token-KD	93.3	66.7	26.7
SelecTKD	93.3	80.0	13.3

Table 18: KD - Threshold, SelecTKD, and θ ablation (500 samples, 2 epochs)

θ (Token-KD)	AR sel%	LPG	Conclusion
0.0	100.0%	23.4%	No filtering baseline
0.5	~2.1%	40.0%	Too aggressive
0.7	7.41%	0.0%	Optimal full training
0.9	~0.3%	30.0%	Too restrictive

4.5 Finetuning

Since we have done further fine-tuning on the same model, we have done knowledge distillation, we adapted the same layer evaluation. Evaluation process is organized into 4 layers; The first layer tests general reasoning on standard benchmarks. The second layer evaluates Arabic and GCC-specific cultural competence using AraDiCE-Qatar. The third layer assesses Bahraini corpus-based linguistic capability using Habash-derived tasks. The fourth layer examines bilingual behavioral consistency using paired Arabic-English prompts and diagnostic metrics such as LPG and CLCS.

Table 19: Complete evaluation result comparison on KD and fine-tuning

LAYER	METRIC	BASE-7B	TOKEN-KD	SELECTKD	FT-V1	FT-V2
L1	**MMLU-EN**	73.8 ± 0.6	73.5 ± 0.6	73.4 ± 0.6	59.5	70.8
L1	**ArabicMMLU**	63.0	63.0	58.0	58.0	58.8
L1	**GSM8K**	18.0	18.0	26.0	20.0	20.0
L2A	**AraDiCE-Qatar**	**53.3**	**56.7**	**50.0**	**60**	**66.7**
L2B	**Habash Overall**	77.7	77.7	70.7	4.7	40.7
L2B	**Dialect ID**	94.0	95.0	80.0	7.0	84.0
L2B	**Normalize**	68.0	70.0	53.0	1.0	25.0
L2B	**Code-Switch**	71.0	68.0	**79.0**	**6.0**	**13.0**
L3	**EN Accuracy**	86.7	86.7	**86.7**	**86.7**	**86.7**
L3	**AR Accuracy**	63.3	63.3	**86.7**	**80**	**70.7**
L3	**LPG**	23.4	23.4	0.0	6.7	16.7
L3	**CLCS**	60.0	60.0	**80.0**	**66.7**	**60.0**
L3	**ECE-AR**	19.78	19.53	5.24	9.93	10.23
PERPLEXITY		5.19	5.20	5.34	3.79	3.89

Table 19 reports the performance of the baseline and fine-tuned models across all evaluation layers, and the results are visualized in Figure 11. Overall, the results indicates that the fine-tuning approach generate a mixed but informative effect. On one hand, this technique improves the cultural alignment and Bahraini dialect specialization, since the model retrieve the knowledge from datasets and its clear in the responses. On the other hand, it preserves core bilingual behavior alignment and maintains competitive performance on standard benchmark.

On MMLU-EN, the performance in fine-tuned model FT-v1 is 59.5%, which is less than the base model, while the instructed FT-v2 it is improving to reach 70.8%, therefore it is a quick recovery. On Arabic MMLU, all the models remain steady between 63-58% ,however both fine-tuned model showed strong performance near the SELECTKD, while the instructed fine-tuned achieved slightly higher. Furthermore, Both optimized models achieve 20% in GSM8K,

which is less than the SELECTKD but still demonstrates stable mathematical reasoning retention. This implies that solving mathematical problems is not solving the primary goal of fine-tuning process, therefore, any changes made here should be seen as incidental rather than task driven. The strongest gains appear in **AraDiCE-Qatar**, where FT-v1 reaches **60.0%** and FT-V2 reaches **66.7%**. This shows that the fine-tuning stage improves Gulf cultural understanding, with FT-V2 achieving the best result in this layer. These results support the claim that curriculum-based fine-tuning helps the model adapt more effectively to GCC-related cultural knowledge.

The Habash corpus layer show a clear divergence between the two fine-tuned versions. The first version FT-v1 perform poorly on this layer with 4.7%, overall, 7% on Dialect ID, 1.0% on Normalize and 6% on code-switch. This indicates that FT-v1 tends to respond as a generative dialogue model rather than as a task-solving analytical model. FT-V2 improves substantially, reaching 40.7% overall, 84.0% on Dialect ID, 25.0% on Normalize, and 13.0% on Code-Switch, which suggests that the revised fine-tuning version recovers much of the analytical capability needed for Bahraini corpus tasks. In the bilingual behavioral evaluation, both FT-v1 and FT-V2 retain **86.7%** English accuracy. FT-v1 reaches **80.0%** Arabic accuracy with an **LPG of 6.7%**, while FT-V2 reaches **70.7%** Arabic accuracy with an **LPG of 16.7%**. FT-v1 therefore shows better bilingual balance, while FT-V2 shows a larger gap between English and Arabic performance. The **CLCS** results confirm this pattern: FT-v1 achieves **66.7%**, whereas FT-V2 reaches **60.0%**, indicating that FT-v1 preserves cross-lingual consistency more effectively.

The outcome of perplexity also varies in both fine-tuned versions. FT-v1 achieves a PPL of 3.79, while FT-V2 reaches 3.89. Both values are lower than the base model and KD baselines, which suggests that the fine-tuned models are better adapted to the training distribution. However, the lower perplexity should be interpreted alongside the behavioral metrics, since a lower score alone does not guarantee better dialectal or analytical performance.

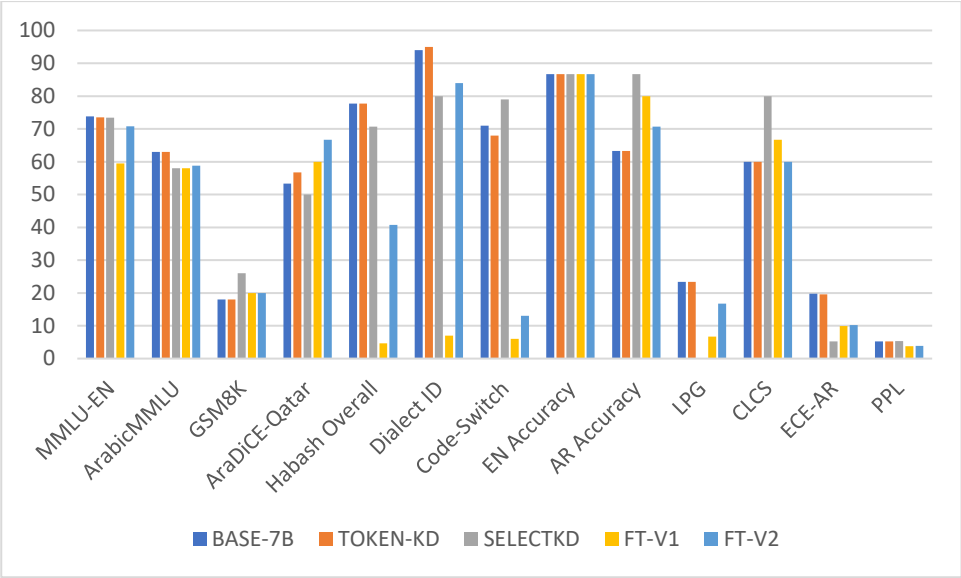


Figure 11: Comparative performance analysis of baseline, knowledge distillation, and fine-tuned models across multilingual reasoning, Arabic/GCC evaluation, behavioral metrics, and perplexity benchmarks.

4.6 Discussion

Figure 11 compares the baseline, knowledge distillation, and fine-tuned models across multilingual reasoning, Arabic/GCC evaluation, bilingual consistency, and perplexity. The figure shows that the KD models provide a strong initialization for bilingual Arabic-English performance, while the fine-tuned variants shift the model toward different strengths depending on the training strategy. SelectKD gives the strongest bilingual balance among the KD baselines, as reflected by its low LPG and high CLCS. Among the fine-tuned models, FT-v1 preserves bilingual consistency more effectively, whereas FT-V2 achieves stronger results on MMLU-EN, AraDiCE-Qatar, and Habash-based tasks. This indicates that staged fine-tuning improves dialect specialization, but the final behavior depends on whether the model is optimized for balance or for task-specific adaptation. In addition, the fine-tuning dataset was designed to support Bahraini specialization for deployment purposes. However, because it was derived largely from series and television show transcripts, it strongly shaped the model's behavior toward dialectal generation. As a result, the model gained Bahraini dialect knowledge but showed weaker analytical ability. To mitigate this limitation, instruction-format data was incorporated into FT-V2.

Additionally, interesting insights are extracted from the previously shown compression results when compared to each other in terms of their compression ratios, perplexities, task accuracies, and compression-accuracy tradeoffs.

Firstly, the **compression ratios** were computed for quantized ALLaM-7B using 5 different methods in Table 6, and additionally for four pruned models in Table 9. Comparing between them, it is found that on average quantization is capable of reducing the model size to a much more significant degree than pruning.

Additionally, the **perplexity** values reported in Table 7 for quantization, Table 10 for pruning, and Table 16 for knowledge distillation show that the perplexity gain was marginal for the 8-bit quantized models and the distilled models. Both the 4-bit quantized models and the pruned models showed higher perplexity gains comparatively. Thus, the simple metric of perplexity can tell us that 8-bit quantization generally produces more stable models compared to lower precisions and to pruning, and that pruning produces models equivalent to 4-bit quantized models in terms of the language predictive ability.

Next, the **task performance** benchmarks between the baseline and the compressed models are compared using the mean accuracy scores given in Table 8 for quantization and Table 11 for pruning. Since ALLaM-7B was used in both the compression techniques, it can be compared directly, and it is clear that quantization's performance was superior across almost all methods and tasks, with the one exception of QLoRA 4-bit performing worse on metabench than the pruned models. This further supports the conclusion that 50% pruned models, at least if not optimized, will generally produce comparable task performance to 4-bit quantized models, which was also similarly seen from the perplexity scores.

The **compression-accuracy tradeoff** of all methods is finally compared. For the models tested, the 8-bit quantization methods all achieved nearly the same or exceeded the accuracy as the original model with half of the original size. Impressively, GPTQ 4-bit also achieves a very high accuracy relative to the original model with only a quarter of the size.

Chapter 5
Conclusion and Future Work

This report investigated several compression and optimization methods, such as quantization, pruning, distillation, and fine-tuning and applied them on a variety of different large language models, including the Arabic model ALLaM and the latest multilingual models like Qwen, Llama, and Gemma. The compressed models were evaluated relative to their base counterparts, and several results were extracted, such as the compression ratio, perplexity, and task accuracy. The primary results show that the goal of achieving 50% reduction in model size while preserving 85% of the performance was achieved with many of the compression methods, such as ALLaM-7B when quantized on all the 8-bit methods (LLM.int8(), GPTQ with and without activations) and the weight-only GPTQ 4-bit method, and when pruned with SparseGPT and Wanda. Pruned Qwen3-8B and Gemma3-12B also achieved this goal but in the English benchmarks only.

Limitations. The compression results were conducted on a limited size range of models, mostly on models not exceeding the size of 7B. Additionally, while a varied range of model architectures were included, the report did not include experiments on Mixture-of-Experts (MoE) models due to their more complex architecture that requires special handling compared to standard dense models. Additionally, a limited number of Arabic models, metrics, and datasets were used due to the limited availability and issues faced with the existing resources, such as model architectural issues and the low quality of available metrics and datasets.

Project Implications. The results of this project provide insight on the rarely researched aspect of how a wide variety of LLM compression methods affect the accuracy and performance of multilingual models when dealing with the Arabic language, and in addition to how it affects models specifically trained for Arabic like ALLaM. Additionally, with the range of techniques tested, the comparative analysis works as a guide for choosing the most optimal compression method for practical limited-resource deployment.

Future Research. Since this report mainly experimented on the Arabic model ALLaM and used ArabicMMLU as the main Arabic evaluation dataset, further research can also be done on other Arabic models using a more diverse range of Arabic evaluation metrics. This would also drive the development of more Arabic models with bigger sizes that can be compressed with

higher quality. Additionally, more compression and optimization methods can be tested both on their own and in combinations to gain more comprehensive insight, especially with the additional context of Arabic performance. Finally, further developing the proposed ACAI platform and fine-tuning a high-performance Arabic-Bahraini model is in the plan for future work.

Overall, the study confirms that compression-aware adaptation is a practical strategy for Arabic and multilingual large language models, and that the best deployment approach depends on the balance between model size, accuracy, bilingual behavior, and task specialization. The findings provide a useful reference for selecting between quantization, pruning, distillation, and fine-tuning in limited-resource settings. Future work should extend this analysis to larger models, broader Arabic benchmarks, and more diverse dialectal and instruction-following datasets.

References

Abdulrahim, D., Inoue, G., Shamsan, L., Khalifa, S., Habash, N., 2022. The Bahrain Corpus: A Multi-genre Corpus of Bahraini Arabic, in: Calzolari, N., Béchet, F., Blache, P., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Odijk, J., Piperidis, S. (Eds.), Proceedings of the Thirteenth Language Resources and Evaluation Conference. Presented at the LREC 2022, European Language Resources Association, Marseille, France, pp. 2345–2352.

Al-Khalifa, S., Durrani, N., Al-Khalifa, H., Alam, F., 2025. The Landscape of Arabic Large Language Models (ALLMs): A New Era for Arabic Language Technology. https://doi.org/10.48550/arXiv.2506.01340

Alyafeai, Z., Masoud, M., Ghaleb, M., Al-shaibani, M.S., 2021. Masader: Metadata Sourcing for Arabic Text and Speech Data Resources. https://doi.org/10.48550/arXiv.2110.06744

Asgari, E., Kheir, Y.E., Javaheri, M.A.S., 2025. MorphBPE: A Morpho-Aware Tokenizer Bridging Linguistic Complexity for Efficient LLM Training Across Morphologies. https://doi.org/10.48550/arXiv.2502.00894

Barakat, H., 2026. Bahraini Speech Dataset [WWW Document]. URL https://huggingface.co/datasets/Hishambarakat/Bahraini_Speech_Dataset (accessed 5.12.26).

Bari, M.S., Alnumay, Y., Alzahrani, N.A., Alotaibi, N.M., Alyahya, H.A., AlRashed, S., Mirza, F.A., Alsubaie, S.Z., Alahmed, H.A., Alabduljabbar, G., Alkhathran, R., Almushayqih, Y., Alnajim, R., Alsubaihi, S., Mansour, M.A., Alrubaian, M., Alammari, A., Alawami, Z., Al-Thubaity, A., Abdelali, A., Kuriakose, J., Abujabal, A., Al-Twairesh, N., Alowisheq, A., Khan, H., 2024. ALLaM: Large Language Models for Arabic and English [WWW Document]. arXiv.org. URL https://arxiv.org/abs/2407.15390v1 (accessed 2.18.26).

Cao, Z., Gu, B., Zhang, C., Gimenes, P., Lu, J., Cheng, J., Gao, X., Zhao, Y., 2025. Scaling Laws For Mixed Quantization. https://doi.org/10.48550/arXiv.2410.06722

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., Tafjord, O., 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. https://doi.org/10.48550/arXiv.1803.05457

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., Schulman, J., 2021. Training Verifiers to Solve Math Word Problems. https://doi.org/10.48550/arXiv.2110.14168

Dettmers, T., Lewis, M., Belkada, Y., Zettlemoyer, L., 2022a. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale [WWW Document]. arXiv.org. URL https://arxiv.org/abs/2208.07339v2 (accessed 4.23.26).

Dettmers, T., Lewis, M., Belkada, Y., Zettlemoyer, L., 2022b. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale. https://doi.org/10.48550/arXiv.2208.07339

Dettmers, T., Pagnoni, A., Holtzman, A., Zettlemoyer, L., 2023. QLoRA: Efficient Finetuning of Quantized LLMs. https://doi.org/10.48550/arXiv.2305.14314

El Filali, A., ALOUI, M., Husaain, T., Alzubaidi, A., Boussaha, B.E.A., Cojocaru, R., Fourrier, C., Habib, N., Hacid, H., 2025. Open Arabic LLM Leaderboard - a Hugging Face Space

by OALL [WWW Document]. URL https://huggingface.co/spaces/OALL/Open-Arabic-LLM-Leaderboard (accessed 2.17.26).

Elmadany, A., Nagoudi, E.M.B., Abdul-Mageed, M., 2022. ORCA: A Challenging Benchmark for Arabic Language Understanding [WWW Document]. arXiv.org. URL https://arxiv.org/abs/2212.10758v2 (accessed 2.17.26).

Fanar Team, Abbas, U., Ahmad, M.S., Alam, F., Altinisik, E., Asgari, E., Boshmaf, Y., Boughorbel, S., Chawla, S., Chowdhury, S., Dalvi, F., Darwish, K., Durrani, N., Elfeky, M., Elmagarmid, A., Eltabakh, M., Fatehkia, M., Fragkopoulos, A., Hasanain, M., Hawasly, M., Husaini, M., Jung, S.-G., Lucas, J.K., Magdy, W., Messaoud, S., Mohamed, A., Mohiuddin, T., Mousi, B., Mubarak, H., Musleh, A., Naeem, Z., Ouzzani, M., Popovic, D., Sadeghi, A., Sencar, H.T., Shinoy, M., Sinan, O., Zhang, Y., Ali, A., Kheir, Y.E., Ma, X., Ruan, C., 2025. Fanar: An Arabic-Centric Multimodal Generative AI Platform. https://doi.org/10.48550/arXiv.2501.13944

Farghaly, A., Shaalan, K., 2009. Arabic Natural Language Processing: Challenges and Solutions. ACM Transactions on Asian Language Information Processing 8, 14:1-14:22. https://doi.org/10.1145/1644879.1644881

Frantar, E., Alistarh, D., 2023. SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot. https://doi.org/10.48550/arXiv.2301.00774

Gemma Team, Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., Rouillard, L., Mesnard, T., Cideron, G., Grill, J., Ramos, S., Yvinec, E., Casbon, M., Pot, E., Penchev, I., Liu, G., Visin, F., Kenealy, K., Beyer, L., Zhai, X., Tsitsulin, A., Busa-Fekete, R., Feng, A., Sachdeva, N., Coleman, B., Gao, Y., Mustafa, B., Barr, I., Parisotto, E., Tian, D., Eyal, M., Cherry, C., Peter, J.-T., Sinopalnikov, D., Bhupatiraju, S., Agarwal, R., Kazemi, M., Malkin, D., Kumar, R., Vilar, D., Brusilovsky, I., Luo, J., Steiner, A., Friesen, A., Sharma, Abhanshu, Sharma, Abheesht, Gilady, A.M., Goedeckemeyer, A., Saade, A., Feng, A., Kolesnikov, A., Bendebury, A., Abdagic, A., Vadi, A., György, A., Pinto, A.S., Das, A., Bapna, A., Miech, A., Yang, A., Paterson, A., Shenoy, A., Chakrabarti, A., Piot, B., Wu, B., Shahriari, B., Petrini, B., Chen, C., Lan, C.L., Choquette-Choo, C.A., Carey, C.J., Brick, C., Deutsch, D., Eisenbud, D., Cattle, D., Cheng, D., Paparas, D., Sreepathihalli, D.S., Reid, D., Tran, D., Zelle, D., Noland, E., Huizenga, E., Kharitonov, E., Liu, F., Amirkhanyan, G., Cameron, G., Hashemi, H., Klimczak-Plucińska, H., Singh, H., Mehta, H., Lehri, H.T., Hazimeh, H., Ballantyne, I., Szpektor, I., Nardini, I., Pouget-Abadie, J., Chan, J., Stanton, J., Wieting, J., Lai, J., Orbay, J., Fernandez, J., Newlan, J., Ji, J., Singh, J., Black, K., Yu, K., Hui, K., Vodrahalli, K., Greff, K., Qiu, L., Valentine, M., Coelho, M., Ritter, M., Hoffman, M., Watson, M., Chaturvedi, M., Moynihan, M., Ma, M., Babar, N., Noy, N., Byrd, N., Roy, N., Momchev, N., Chauhan, N., Sachdeva, N., Bunyan, O., Botarda, P., Caron, P., Rubenstein, P.K., Culliton, P., Schmid, P., Sessa, P.G., Xu, P., Stanczyk, P., Tafti, P., Shivanna, R., Wu, R., Pan, R., Rokni, R., Willoughby, R., Vallu, R., Mullins, R., Jerome, S., Smoot, S., Girgin, S., Iqbal, S., Reddy, S., Sheth, S., Põder, S., Bhatnagar, S., Panyam, S.R., Eiger, S., Zhang, S., Liu, T., Yacovone, T., Liechty, T., Kalra, U., Evci, U., Misra, V., Roseberry, V., Feinberg, V., Kolesnikov, V., Han, W., Kwon, W., Chen, X., Chow, Y., Zhu, Y., Wei, Z., Egyed, Z., Cotruta, V., Giang, M., Kirk, P., Rao, A., Black, K., Babar, N., Lo, J., Moreira, E., Martins, L.G., Sanseviero, O., Gonzalez, L., Gleicher, Z., Warkentin, T., Mirrokni, V., Senter, E., Collins, E., Barral, J., Ghahramani, Z., Hadsell, R., Matias, Y., Sculley, D., Petrov, S., Fiedel, N., Shazeer, N., Vinyals, O., Dean, J., Hassabis, D., Kavukcuoglu, K., Farabet, C., Buchatskaya, E., Alayrac, J.-B., Anil, R., Dmitry, Lepikhin, Borgeaud, S., Bachem, O., Joulin, A., Andreev, A., Hardin, C., Dadashi, R.,

Hussenot, L., 2025. Gemma 3 Technical Report. https://doi.org/10.48550/arXiv.2503.19786
Habib, N., Fourrier, C., Kydlíček, H., Wolf, T., Tunstall, L., 2023. LightEval: A lightweight framework for LLM evaluation.
Hassibi, B., Stork, D.G., 1992. Second order derivatives for network pruning: optimal brain surgeon, in: Proceedings of the 6th International Conference on Neural Information Processing Systems, NIPS'92. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 164–171.
Hassibi, B., Stork, D.G., Wolff, G.J., 1993. Optimal Brain Surgeon and general network pruning, in: IEEE International Conference on Neural Networks. Presented at the IEEE International Conference on Neural Networks, pp. 293–299 vol.1. https://doi.org/10.1109/ICNN.1993.298572
Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., Steinhardt, J., 2021. Measuring Massive Multitask Language Understanding. https://doi.org/10.48550/arXiv.2009.03300
Hu, T., Zhou, X.-H., 2024. Unveiling LLM Evaluation Focused on Metrics: Challenges and Solutions. https://doi.org/10.48550/arXiv.2404.09135
Huang, H., Yu, F., Zhu, J., Sun, X., Cheng, H., Song, D., Chen, Z., Alharthi, A., An, B., He, J., Liu, Z., Zhang, Z., Chen, J., Li, J., Wang, B., Zhang, L., Sun, R., Wan, X., Li, H., Xu, J., 2023. AceGPT, Localizing Large Language Models in Arabic [WWW Document]. arXiv.org. URL https://arxiv.org/abs/2309.12053v5 (accessed 2.18.26).
HuggingFace, 2025. Perplexity of fixed-length models [WWW Document]. URL https://huggingface.co/docs/transformers/en/perplexity (accessed 5.3.26).
Ip, J., Vongthongsri, K., 2026. deepeval.
Koubaa, A., Ammar, A., Ghouti, L., Najar, O., Sibaee, S., 2024. ArabianGPT: Native Arabic GPT-based Large Language Model [WWW Document]. arXiv.org. URL https://arxiv.org/abs/2402.15313v2 (accessed 2.18.26).
LeCun, Y., Denker, J., Solla, S., 1989. Optimal Brain Damage, in: Advances in Neural Information Processing Systems. Morgan-Kaufmann.
Lin, S., Hilton, J., Evans, O., 2022. TruthfulQA: Measuring How Models Mimic Human Falsehoods. https://doi.org/10.48550/arXiv.2109.07958
Ma, X., Fang, G., Wang, X., 2023. LLM-Pruner: On the Structural Pruning of Large Language Models. Advances in Neural Information Processing Systems 36, 21702–21720.
Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., Paul, S., Bossan, B., Tietz, M., 2026. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods.
Mikolov, T., Karafiát, M., Burget, L., Černocký, J., Khudanpur, S., 2010. Recurrent neural network based language model. Presented at the Proc. Interspeech 2010, pp. 1045–1048. https://doi.org/10.21437/Interspeech.2010-343
Movva, R., Lei, J., Longpre, S., Gupta, A., DuBois, C., 2022. Combining Compressions for Multiplicative Size Scaling on Natural Language Tasks, in: Calzolari, N., Huang, C.-R., Kim, H., Pustejovsky, J., Wanner, L., Choi, K.-S., Ryu, P.-M., Chen, H.-H., Donatelli, L., Ji, H., Kurohashi, S., Paggio, P., Xue, N., Kim, S., Hahm, Y., He, Z., Lee, T.K., Santus, E., Bond, F., Na, S.-H. (Eds.), Proceedings of the 29th International Conference on Computational Linguistics. Presented at the COLING 2022, International Committee on Computational Linguistics, Gyeongju, Republic of Korea, pp. 2861–2872.
Omran, T.M., Sharef, B.T., Grosan, C., Li, Y., 2023. Transfer learning and sentiment analysis of Bahraini dialects sequential text data using multilingual deep learning approach. Data & Knowledge Engineering 143, 102106. https://doi.org/10.1016/j.datak.2022.102106

Polino, A., Pascanu, R., Alistarh, D., 2018. Model compression via distillation and quantization. https://doi.org/10.48550/arXiv.1802.05668

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J., 2023. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. https://doi.org/10.48550/arXiv.1910.10683

Red Hat AI, vLLM Project, 2024. LLM Compressor.

Sakaguchi, K., Bras, R.L., Bhagavatula, C., Choi, Y., 2019. WinoGrande: An Adversarial Winograd Schema Challenge at Scale. https://doi.org/10.48550/arXiv.1907.10641

Sun, M., Liu, Z., Bair, A., Kolter, J.Z., 2024. A Simple and Effective Pruning Approach for Large Language Models. https://doi.org/10.48550/arXiv.2306.11695

Sutawika, L., Schoelkopf, H., Gao, L., Abbasi, B., Biderman, S., Tow, J., fattori, ben, Lovering, C., farzanehnakhaee70, Phang, J., Thite, A., Fazz, Aflah, Muennighoff, N., Wang, T., sdtblck, nopperl, gakada, tttyuntian, researcher2, Etxaniz, J., Chris, Lee, H.A., Kasner, Z., Khalid, LSinev, Hsu, J., Kanekar, A., KonradSzafer, AndyZwei, 2024. EleutherAI/lm-evaluation-harness: v0.4.3. https://doi.org/10.5281/zenodo.12608602

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention Is All You Need [WWW Document]. arXiv.org. URL https://arxiv.org/abs/1706.03762v7 (accessed 2.16.26).

von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., Huang, S., Rasul, K., Gallouédec, Q., 2020. TRL: Transformers Reinforcement Learning.

W3Techs, 2025. Usage Statistics and Market Share of Content Languages for Websites, February 2026 [WWW Document]. W3Techs. URL https://w3techs.com/technologies/overview/content_language (accessed 2.16.26).

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., Platen, P. von, Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T.L., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M., 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. https://doi.org/10.48550/arXiv.1910.03771

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, Jian, Tu, J., Zhang, J., Yang, Jianxin, Yang, Jiaxi, Zhou, Jing, Zhou, Jingren, Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, Mei, Xue, M., Li, Mingze, Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, Xingzhang, Wang, X., Zhang, X., Ren, Xuancheng, Fan, Y., Su, Y., Zhang, Yichang, Zhang, Yinger, Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., Qiu, Z., 2025. Qwen3 Technical Report. https://doi.org/10.48550/arXiv.2505.09388

Yang, G., He, C., Guo, J., Wu, J., Ding, Y., Liu, A., Qin, H., Ji, P., Liu, X., 2024. LLMCBench: Benchmarking Large Language Model Compression for Efficient Deployment. https://doi.org/10.48550/arXiv.2410.21352

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., Choi, Y., 2019. HellaSwag: Can a Machine Really Finish Your Sentence? https://doi.org/10.48550/arXiv.1905.07830

Bucilă, C., Caruana, R. and Niculescu-Mizil, A., 2006. Model compression. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535-541. Han, S., Mao, H. and Dally, W.J., 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv preprint arXiv:1510.00149. Hinton, G., Vinyals, O. and Dean, J., 2015. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531. Kim, Y. and Rush, A.M., 2016. Sequence-

level knowledge distillation. arXiv preprint arXiv:1606.07947. Xu, Z. et al., 2024. A survey on knowledge distillation of large language models. arXiv preprint arXiv:2402.13116.

Abdul-Mageed, M., Zhang, C., Elmadany, A., Bouamor, H. and Habash, N. (2021). *NADI 2021: The Second Nuanced Arabic Dialect Identification Shared Task*. [online] pp.244–259. Available at: https://aclanthology.org/2021.wanlp-1.28.pdf [Accessed 5 April 2026].

Qwen, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J. and Dang, K. (2024). *Qwen2.5 Technical Report*. [online] arXiv.org. Available at: https://arxiv.org/abs/2412.15115.

Bura, B. (n.d.) *Sentiment Analysis of Multilingual Dataset of Bahraini Dialects, Arabic, and English*. Available at: https://bura.brunel.ac.uk/handle/2438/26245 (Accessed: 7 April 2026).

Elmadany, A., Habash, N. and Bouamor, H. (2022) *Masader: Metadata Sourcing for Arabic Text and Speech Data Resources*. Proceedings of LREC 2022. Available at: https://aclanthology.org/2022.lrec-1.681/ (Accessed: 8 April 2026).

Hishambarakat (2026) *Bahraini_Speech_Dataset*. Available at: https://huggingface.co/datasets/Hishambarakat/Bahraini_Speech_Dataset (Accessed: 8 April 2026).

Al-Ajmi, A., Khalifa, S., Habash, N. and Eskander, R. (2022) 'The Bahrain Corpus: A Dialectal Arabic Corpus from Multiple Genres', *Proceedings of the Language Resources and Evaluation Conference (LREC 2022)*, pp. 4236–4243.

Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L. and Chen, W. (2022) 'LoRA: Low-Rank Adaptation of Large Language Models', *International Conference on Learning Representations (ICLR)*. Available at: https://openreview.net/forum?id=nZeVKeeFYf9 (Accessed: 10 April 2026).

Mousi, B., Durrani, N., Ahmad, F., Hasan, M.A., Hasanain, M., Kabbani, T., Dalvi, F., Chowdhury, S.A. and Alam, F. (2025) 'AraDiCE: Benchmarks for Dialectal and Cultural Capabilities in LLMs', *Proceedings of the 31st International Conference on Computational Linguistics (COLING 2025)*, pp. 4186–4218. Available at: https://aclanthology.org/2025.coling-main.283/ (Accessed: 11 April 2026).

Qwen Team (2024) 'Qwen2.5 Technical Report', *arXiv preprint arXiv:2412.15115*. Available at: https://arxiv.org/abs/2412.15115 (Accessed: 11 April 2026).

Ruis, L., Bhoopchand, A., Polu, S., Bowman, S.R. and Rocktäschel, T. (2025) ‘Evaluating LLMs is Harder Than You Think: Benchmark Validity’, OpenReview [Preprint]. Paper ID: mdA5lVvNcU.

Wu, X., Zhang, Y., Chen, Y. and Liu, Q. (2025) ‘SelecTKD: Selective Token Knowledge Distillation for Large Language Models’, *arXiv preprint*. Available at: https://arxiv.org/abs/2510.24021 (Accessed: 13 April 2026).

Abdulrahim, D., Inoue, G., Shamsan, L., Khalifa, S. and Habash, N. (2022) ‘The Bahrain Corpus: A Multi-genre Corpus of Bahraini Arabic’, *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pp. 2345–2352.

Egiazarian, V. et al. (2024) 'Extreme Compression of Large Language Models via Additive Quantization', in *Proceedings of the 41st International Conference on Machine Learning (ICML)*. [Online]. Available at: https://arxiv.org/abs/2401.06118 (Accessed: 12 May 2026).

Esser, S.K. et al. (2020) 'Learned Step Size Quantization', in *Proceedings of the 8th International Conference on Learning Representations (ICLR)*. [Online]. Available at: https://arxiv.org/abs/1902.08153 (Accessed: 12 May 2026).

Wang, R. (2025) *Model Quantization: Concepts, Methods, and Why It Matters*, *NVIDIA Technical Blog*. Available at: https://developer.nvidia.com/blog/model-quantization-concepts-methods-and-why-it-matters/.

Jacob, B. et al. (2018) 'Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [Online]. Available at: https://arxiv.org/abs/1712.05877 (Accessed: 12 May 2026).

Kim, S. et al. (2021) 'I-BERT: Integer-only BERT Quantization', in *Proceedings of the 38th International Conference on Machine Learning (ICML)*. [Online]. Available at: https://arxiv.org/abs/2101.01321 (Accessed: 12 May 2026).

Li, Y. et al. (2021) 'BRECQ: Pushing the Limit of Post-Training Quantization by Block-wise Reconstruction', in *Proceedings of the 9th International Conference on Learning Representations (ICLR)*. [Online]. Available at: https://arxiv.org/abs/2102.05426 (Accessed: 12 May 2026).

Nagel, M. et al. (2020) 'Up or Down? Adaptive Rounding for Post-Training Quantization', in *Proceedings of the 37th International Conference on Machine Learning (ICML)*. [Online]. Available at: https://arxiv.org/abs/2004.10568 (Accessed: 12 May 2026).

Xiao, G. et al. (2023) 'SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models', in *Proceedings of the 40th International Conference on Machine Learning (ICML)*. [Online]. Available at: https://arxiv.org/abs/2211.10438 (Accessed: 12 May 2026).

Shao, W. et al. (2024) 'OmniQuant: Omnidirectionally Calibrated Quantization for Large Language Models', in *Proceedings of the 12th International Conference on Learning Representations (ICLR)*. [Online]. Available at: https://arxiv.org/abs/2308.13137 (Accessed: 12 May 2026).

Appendix A
Additional Results

To complement the main results in Chapter 4, additional results are included here.

Quantization

Table 20: Quantization - Per-task scores of ArabicMMLU and metabench.

FP16 denotes the base precision, Int8 denotes LLM.int8(), Int4 denotes QLoRA's double quantization, and W8A8, W8A16, and W4A16 denote GPTQ quantization with the precision of weights (W) and activations (A).

Task/Model	F16	Int8	W8A8	W8A16	Int4	W4A16
ArabicMMLU	70.08	69.84	69.73	**70.16**	65.67	**66.78**
Humanities	74.94	74.31	74.12	**74.67**	70.84	**71.64**
Language	73.63	72.84	73.82	**74.00**	70.29	**70.78**
Other	73.87	74.07	**74.40**	74.11	68.60	**70.29**
Social Studies	66.55	66.32	66.41	**66.67**	63.07	**63.38**
STEM	63.64	63.80	62.67	**63.83**	57.97	**60.19**
metabench	55.10	53.34	55.70	55.24	40.73	**50.06**
ARC	59.31	60.69	**64.14**	58.62	32.41	**60.00**
GSM8K	62.45	64.14	**65.40**	63.71	28.69	**56.12**
HellaSwag	65.59	**64.52**	**64.52**	**64.52**	23.66	**58.06**
MMLU	70.83	70.83	70.83	**71.88**	58.33	**71.88**
TruthfulQA	26.62	24.68	**27.27**	**27.27**	20.78	**24.03**
Winogrande	79.70	75.94	78.95	**79.70**	**69.92**	**69.92**

Pruning

Table 21: Pruning - Model parameter counts after pruning with SparseGPT/Wanda.

	ALLaM-7B	Qwen3-8B	Llama3.1-8B	Gemma3-12B
Dense	7,000,559,616	8,190,735,360	8,030,261,248	12,187,325,040
SparseGPT 50%	3,762,065,637	4,717,787,572	4,540,587,260	6,808,112,210
Wanda 50%	3,762,077,677	4,717,799,804	4,540,599,034	6,808,129,709

Table 22: Pruning – Per-task evaluation scores of ArabicMMLU and metabench.

D denotes the baseline dense models, S denotes the SparseGPT compressed variant, and W denotes the Wanda compressed variant. The bold scores represent the highest score for each task between the sparse variants.

	ALLaM 7B			Qwen3 8B			Llama-3.1 8B			Gemma 3 12B		
Task/Model	**D**	**S**	**W**	**D**	**S**	**W**	**D**	**S**	**W**	**D**	**S**	**W**
ArabicMMLU	70.08	62.62	**63.58**	62.38	**53.70**	50.52	56.69	**40.05**	36.52	65.04	**58.17**	47.00
Humanities	74.94	68.00	**68.58**	59.84	**51.93**	50.66	56.50	**38.73**	33.24	62.90	**56.62**	46.31
Language	73.63	64.82	**65.07**	60.15	**51.09**	47.51	56.8	**38.4**	32.32	65.31	**57.41**	43.74
Other	73.87	65.78	**67.51**	66.26	**56.44**	51.37	63.08	**44.52**	42.19	70.37	**62.48**	53.46
Social Studies	66.55	60.36	**61.02**	61.33	**53.77**	49.80	55.59	**40.9**	37.90	64.18	**58.28**	45.43
STEM	63.64	55.37	**56.87**	64.55	**54.84**	52.05	53.08	**37.99**	36.49	64.11	**56.87**	46.16
metabench	55.10	**45.60**	44.33	57.38	**50.42**	45.60	56.61	**45.45**	41.91	64.14	**54.32**	39.06
ARC	59.31	42.76	**45.52**	72.41	**55.17**	54.48	64.83	**46.9**	42.76	78.62	**64.83**	35.17
GSM8K	62.45	**21.94**	19.41	89.03	61.18	**79.75**	81.86	**38.82**	22.78	90.72	**67.93**	27.43
HellaSwag	65.59	**41.94**	31.18	55.91	**35.48**	27.96	60.22	**34.41**	25.81	70.97	**46.24**	21.51
MMLU	70.83	63.54	**66.67**	83.33	**73.96**	71.88	75.00	**65.62**	58.33	84.38	**77.08**	60.42
TruthfulQA	26.62	**19.48**	16.23	31.82	**26.62**	22.08	29.22	14.94	**17.53**	37.66	**25.32**	20.78
Winogrande	79.7	**71.43**	68.42	66.92	**66.92**	62.41	81.2	**71.43**	65.41	79.70	**69.17**	62.41

Knowledge Distillation

Table 23: Multi-Layer Evaluation Results for Knowledge Distillation and Fine-Tuned Models

LAYER	METRIC	BASE-7B	SFT-KD-7B	SEQ-KD-7B	TOKEN-KD	SELECTKD	FT-V1	FT-V2
L1	**MMLU-EN**	73.8 ± 0.6	—	—	73.5 ± 0.6	73.4 ± 0.6	59.5	70.8
L1	**ArabicMMLU**	63.0	61.0	60.0	63.0	58.0	58.0	58.8
L1	**GSM8K**	18.0	14	18.0	18.0	26.0	20	20
L2A	**AraDiCE-Qatar**	53.3	53.3	36.7	56.7	50.0	60	66.7
L2B	**Habash Overall**	77.7	55.0	60.7	77.7	70.7	4.7	40.7
L2B	**Dialect ID**	94.0	27.0	40.0	95.0	80.0	7.0	84.0
L2B	**Normalize**	68.0	61.0	61.0	70.0	53.0	1.0	25
L2B	**Code-Switch**	71.0	77.0	81.0	68.0	**79.0**	**6.0**	**13.0**
L3	**EN Accuracy**	86.7	86.7	83.3	86.7	**86.7**	**86.7**	**86.7**
L3	**AR Accuracy**	63.3	80.0	76.7	63.3	**86.7**	**80**	**70.7**
L3	**LPG**	23.4	6.7	6.6	23.4	0	6.7	16.7
L3	**CLCS**	60.0	73.3	66.7	60.0	**80.0**	**66.7**	**60.0**
L3	**ECE-AR**	19.78	3.60	16.16	19.53	5.24	9.93	10.23
PPL	**PPL**	5.19	1.32	—	5.20	5.34	3.79	3.89