Paul Vardhan Bethapudi

Efficient Performance Recovery of Pruned LLMs for End Devices

Indian Statistical Institute

Document Details

Submission ID
trn:oid:::3618:142616581

Submission Date
Jun 11, 2026, 5:09 PM GMT+5:30

Download Date
Jun 11, 2026, 5:26 PM GMT+5:30

File Name
paul.pdf

File Size
4.6 MB

79 Pages

23,319 Words

130,230 Characters

4% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography

Custom Section Exclusions

{titlesCount} Section Titles, {keywordsCount} Keywords

Section title	No. of Section Starters	Section Starters
"isi"	2	Indian statistical institute, kolkata

Match Groups

98 Not Cited or Quoted 4%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

3% Internet sources
3% Publications
0% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **98** Not Cited or Quoted 4%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **0** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

3% Internet sources
3% Publications
0% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Internet	arxiv.org 1%
2	Internet	dspace.isical.ac.in:8080 <1%
3	Internet	aman.ai <1%
4	Internet	blog.yueqianlin.com <1%
5	Publication	"Intelligent Systems", Springer Science and Business Media LLC, 2025 <1%
6	Internet	assets-8291.accso.de <1%
7	Publication	Chouhan, Ayush. "Pixel Classification Using U-Net (Semantic Segmentation)", Indi... <1%
8	Internet	www.bestpfe.com <1%
9	Publication	Zhengqing Yuan, Huiwen Xue, Chao Zhang, Yongming Liu. "EvoText: Enhancing N... <1%
10	Internet	aclanthology.org <1%

11 Publication
Ceyu Xu, Yongji Wu, Xinyu Yang, Beidi Chen, Matthew Lentz, Danyang Zhuo, Lisa ... <1%

12 Publication
Valentina Franzoni, Silvia Tagliente, Alfredo Milani. "Generative Models for Sourc... <1%

13 Internet
123dok.net <1%

14 Publication
Savvas, Spyros. "Sentiment Analysis for Financial News", University of Piraeus (Gr... <1%

15 Internet
ecommons.usask.ca <1%

16 Internet
docobook.com <1%

17 Internet
d-nb.info <1%

18 Internet
trepo.tuni.fi <1%

19 Publication
Chunlei Li, Huanyu Li, Guangshuai Gao, Zhoufeng Liu, Pengcheng Liu. "An acceler... <1%

20 Publication
Yao Jing, Bin Guo, Nuo Li, Yasan Ding, Yan Liu, Zhiwen Yu. "Scalable order dispatc... <1%

21 Internet
serwiss.bib.hs-hannover.de <1%

22 Publication
Huiting Ma, Dengao Li, Guiji Zhao, Li Liu, Jian Fu, Xiaole Fan, Zhe Zhang, Yuchen Li... <1%

23 Internet
americaspg.com <1%

24 Publication
Arvind Dagur, Sohit Agarwal, Dhirendra Kumar Shukla, Shabir Ali, Sandhya Sharm... <1%

25 Publication

Goyal, Shubham. "Scalable Local Characters Using Fine-Tuned Small Language M... <1%

26 Internet

nips.cc <1%

27 Internet

scholarworks.alaska.edu <1%

28 Internet

swabhs.com <1%

29 Publication

"Artificial Intelligence", Springer Science and Business Media LLC, 2024 <1%

30 Publication

"Computer Vision – ECCV 2024", Springer Science and Business Media LLC, 2025 <1%

31 Publication

A. Punitha, S. Syedakbar, S. Jeyasudha. "Advanced Pathways in Electrical, Commu... <1%

32 Publication

Ansell, Alan John. "Efficient and Composable Adaptation for Cross-Lingual Transfe... <1%

33 Publication

Hou-I Liu, Marco Galindo, Hongxia Xie, Lai-Kuan Wong, Hong-Han Shuai, Yung-Hu... <1%

34 Publication

Obinwanne, Uchechukwu Emmanuel. "Optimized Large Language Model for Hate... <1%

35 Publication

Oswald Campesato. "Chapter 6: LLMs and Fine-Tuning (2)", Walter de Gruyter Gm... <1%

36 Publication

Paul Trust, Rosane Minghim. "A Study on Text Classification in the Age of Large L... <1%

37 Publication

R. N. V. Jagan Mohan, B. H. V. S. Rama Krishnam Raju, V. Chandra Sekhar, T. V. K. P... <1%

38 Publication

Sabot, Andrew. "Everything Is a Matrix: Minimizing Data Movement and Paramet... <1%

39 Publication
Sung, Yi-Lin. "Efficient Fine-Tuning for Language and Multimodal Models", The Un... <1%

40 Publication
Xunyu Zhu, Jian Li, Yong Liu, Can Ma, Weiping Wang. "A Survey on Model Compre... <1%

41 Internet
d197for5662m48.cloudfront.net <1%

42 Internet
hal.science <1%

43 Internet
ia801302.us.archive.org <1%

44 Internet
lirias.kuleuven.be <1%

45 Internet
prod-ms-be.lib.mcmaster.ca <1%

46 Internet
www.promptlayer.com <1%

47 Publication
Ghosh, Nikhil. "Theoretical Foundations of Deep Learning: Optimization, Generali... <1%

48 Publication
Shivam R Solanki, Drupad K Khublani. "Generative Artificial Intelligence", Springe... <1%

49 Publication
Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Ji-Rong Wen. "Large Language M... <1%

50 Publication
Xinyu Wang, Zhaoxin Fan, Faguo Wu, Hongwei Zheng, Yuanze Hu, Gen Li, Zhichao... <1%

51 Publication
He, Xuehai. "Bridging the Gap Between Multimodal Foundation Models and Worl... <1%

Efficient Performance Recovery of Pruned LLMs for End Devices

Paul Vardhan Bethapudi

Efficient Performance Recovery of Pruned LLMs for End Devices

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology

in

Computer Science

with specialization in Data Science

by

Paul Vardhan Bethapudi

[Roll No: CS2420]

under the supervision of

Prof. Ujjwal Bhattacharya

Professor

Computer Vision and Pattern Recognition Unit

Indian Statistical Institute

Kolkata 700108, India

May 2026

Certificate

This is to certify that the dissertation entitled **"Efficient Performance Recovery of Pruned LLMs for End Devices"** submitted by **Paul Vardhan (CS2420)** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science with specialization in Data Science**, is a bonafide record of work carried out by him under my supervision and guidance.

The dissertation has fulfilled the requirements as per the regulations of the institute and, in my opinion, has reached the standard required for submission.

Prof. Ujjwal Bhattacharya
Professor
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute
Kolkata 700108, India

Dedication

To my family, teachers, and everyone who supported me throughout this academic journey.

Acknowledgments

I would like to express my sincere gratitude to my supervisor, Prof. Ujjwal Bhattacharya, for his guidance and tremendous support throughout this dissertation. His feedback greatly helped me streamline the pruning, recovery, and evaluation pipeline.

I am also thankful to the Indian Statistical Institute, Kolkata, for providing the academic exposure and computational support required for this work. The experiments on Llama-3.2-3B Instruct model, Wanda pruning, QLoRA, DoRA, WikiText-2, and GSM8K helped me understand the practical challenges involving efficient LLM deployment.

I thank my teachers, classmates, and friends for their encouragement and useful discussions during different stages of the project. Those suggestions helped me improve both in technical aspects and presentation of this dissertation.

I am deeply grateful to my family for their constant support, patience, and encouragement. Their belief in me gave me strength during the difficult phases of experimentation, debugging, and writing.

Finally, I thank everyone who directly or indirectly supported me in completing this dissertation.

Paul Vardhan Bethapudi
Indian Statistical Institute
Kolkata 700108, India

Abstract

Large Language Models have shown strong performance across reasoning, language understanding, and generation tasks, but their computational and memory requirements make deployment on low resource devices difficult. This dissertation studies efficient performance recovery of pruned LLMs, focusing on whether a pruned model can regain useful task performance through parameter-efficient fine-tuning while preserving the benefits of compression.

The work uses Llama-3.2-3B-Instruct as the dense reference model and applies activation-aware Wanda pruning at multiple sparsity levels(20%, 30% and 50%). The pruned models are evaluated on language modeling and reasoning tasks using WikiText-2 perplexity and GSM8K (Grade School Math 8000 problems) accuracy. Adaptation is performed using efficient parameter adaptation methods like QLoRA and DoRA. Additionally, the thesis evaluates the balance between sparsity, precision, perplexity, speed, memory usage, and device compatibility.

Based on the results obtained from the experiments conducted, we can safely say that the adapter recovery technique improves the performance of sparse models. DoRA performs better in GSM8K recovery compared to QLoRA recovery at sparsity of 20%, 30%, and 50%. Additionally, the WikiText-2 perplexity for QLoRA is slightly lower than DoRA at 30% and 50% sparsity. The largest recovery gain occurs at 50% sparsity, where GSM8K accuracy improves from 0.1800 to 0.4508 using DoRA recovery. However, the best final recovered GSM8K accuracy is obtained at 20% sparsity with DoRA.

Keywords: Large Language Models(LLM), Model Pruning, Wanda, LoRA QLoRA, DoRA, Parameter-Efficient Fine-Tuning(PEFT), Model Compression, End-Device Deployment, GSM8K, WikiText-2.

Contents

List of Figures

List of Tables

List of Algorithms

iThenticate Page 19 of 85 - Integrity Submission Submission ID trn:oid:::3618:142616581

Abbreviations

Abbreviation	Full Form
LLM	Large Language Model
PEFT	Parameter-Efficient Fine-Tuning
QLoRA	Quantized Low-Rank Adaptation
DoRA	Weight-Decomposed Low-Rank Adaptation
LoRA	Low-Rank Adaptation
Wanda	Pruning by Weights and Activations
PPL	Perplexity
VRAM	Video Random Access Memory
GSM8K	Grade School Math 8K Dataset
HF	Hugging Face
GPU	Graphics Processing Unit
CPU	Central Processing Unit
FLOPs	Floating Point Operations
SFT	Supervised Fine-Tuning

iThenticate Page 20 of 85 - Integrity Submission Submission ID trn:oid:::3618:142616581

28

Chapter 1

Introduction

1.1 Background

Large Language Models have been crucial to recent advances in natural language processing. They are capable of generating meaningful text, following instructions, summarizing lengthy documents, dealing with reasoning challenges, and adapting to a wide variety of downstream tasks. Nevertheless, this comes with a downside. Large Language Models even with a relatively small number of billions of parameters require a lot of memory, computation power, and time inference when put into practice anywhere except large servers.

An apparent trade-off arises. More significant models usually have better language understanding and reasoning capabilities. Yet, their size makes their use on end-users' devices or moderate hardware a challenge. From the perspective of real applications, however, accuracy alone is insufficient. Models must function within memory bounds, provide timely responses, and produce tokens efficiently under these conditions. This factor becomes especially critical when inference is to be carried out near the end-users instead of in the cloud exclusively.

One way of dealing with this problem could be through model compression. Several techniques exist to this end. Pruning stands out in that it removes or deactivates unnecessary weights. In theory, pruning is supposed to result in model sparsity and thus reduce memory footprint and lower the computational burden. Reality, though, might differ. Weights removal from the pretrained large language model would distort its representations that were learned during the pre-training and instruction tuning process. As a result, quality of language modeling goes down or reasoning suffers, or both.

The present work focuses on what can be retrieved by pruning. Sparsity is considered here merely as means to an end and not an end in itself. Besides that, an attempt is made to investigate how much of the lost performance can be recovered through lightweight adaptation. The base model being used is Llama-3.2-3B-Instruct. Wanda pruning is applied at several sparsity levels. The checkpoints produced this way are then rescued by parameter-efficient fine-tuning methods (QLoRA and DoRA). Performance evaluation was done using WikiText-2

perplexity, GSM8K accuracy, as well as memory and throughput figures.

1.2 Motivation

The rationale behind the current research is inspired by observations made throughout our initial efforts to implement compression of language models. It does not make sense to compress the model if the resulting variant lacks the necessary capabilities to perform the intended task. sparser model loses its usefulness fast when pruning interferes with reasoning. On the other hand, if the process of recovery after pruning or the final model architecture itself becomes too bulky, the entire effort becomes impractical because of the need for excessive memory space. Therefore, the core question is not about the ability to prune a model, but rather whether the model can be pruned and recovered successfully with a preserved advantage in efficiency.

It becomes even more crucial to consider this problem in the context of end-device use cases. Cloud environments with large GPUs and enough memory space hide various inefficiencies. When working with smaller GPUs or real resource-constrained devices, such inefficiencies are readily apparent. Memory capacity defines whether or not a model can fit into memory, latency defines usability, and throughput influences efficiency of token generation. Privacy and offline usage become an issue closer to the user. Hence, deploying language models should be approached both as a modeling problem and as a systems problem, similarly to model training.

Pruning can become particularly appealing due to the idea of removing unnecessary parameters. At the same time, mere pruning may prove to be insufficient. A model that was pruned with low or medium sparsity will retain general language characteristics but will have poor performance in specific tasks. Such effects would be critical in case of reasoning, where even slight shifts of internal representation could result in incorrect answer. Another problem is that the effect of pruning on a language model varies depending on the task, which can cause a shift in perplexity score in one direction and decrease accuracy in mathematical reasoning in another direction.

Therefore, some way to recover the model after pruning needs to be considered. Full-fledged fine-tuning becomes prohibitively costly in terms of computation and time required. Instead, parameter-efficient fine-tuning can provide a more reasonable solution as a means to update only a part of model parameters. As a part of the current dissertation, the process of model recovery with QLoRA and DoRA adapters after applying Wanda pruning was tested.

The other reason for this choice is methodological. The reliance on one measure alone produces a partial account of both pruning and retraining. The GSM8K accuracy measure can be useful for measuring reasoning ability but does not capture the complete quality of language model. Similarly, WikiText-2 perplexity can be helpful for measuring next-token prediction skills but is unable to measure multi-step reasoning ability. Additionally, none of these measures can be used in isolation from measures of memory and throughput efficiency.

1.3 Problem Statement

This problem involves the optimal performance recovery of pruned Large Language Models (LLMs) in constrained computational setups. Pruning is used to decrease parameter efficiency in pretrained models, followed by adaptation in order to recover the lost performance due to the pruning process. The main idea is to increase the effectiveness of the task-specific recovered model and keep the benefits that come along with compression.

Formally, consider M to be the pre-trained model initially and consider s to be the desired sparsity. Pruning algorithm P results in the pruned model M_s as

$$M_s = P(M, s),$$

where M_s contains sparse parameters as per the described pruning policy. Since pruning has an adverse effect on the ability of the model to perform natural language processing and reasoning tasks, the recovering function R is then performed using the training data set D_{train}:

$$M_s^{\text{rec}} = R(M_s, D_{\text{train}}).$$

The objective is to achieve a model M_s^{rec} that performs increasingly similarly to the original model M while retaining advantages in memory usage, inferability, or compression. Ideally, the relationship can be stated as follows:

$$\text{Performance}(M_s^{\text{rec}}) \approx \text{Performance}(M),$$

subject to reduced deployment cost compared with the uncompressed model.

Reaching such an approximation poses certain difficulties. First, different sparsity degrees have different impacts on the model, some are enough to maintain its structure for recovering, while others take away too much capacity. Second, different recovery approaches may behave in diverse ways depending on pruning severity and specific evaluations. Thus, both pruning and recovery approaches are considered together and not separately. The study follows a consistent chain including initial evaluation, pruning, subsequent evaluation, recovery, and final evaluation.

1.4 Research Objectives

In particular, the first objective is to set up a thorough experimental pipeline for the effective pruning and recovery of a Llama-scale model. The starting point in the pipeline is the base Llama-3.2-3B-Instruct model, which must be benchmarked under the same protocol before compressing it in any way. It is crucial since all the other experiments will be conducted in

comparison with that baseline.

The second objective is related to experimenting with different degrees of sparsity for Wanda pruning. In our experiment, we treat various sparsity levels as separate compression protocols rather than as multiple iterations of one and the same experiment. That choice enables us to observe the response of our model to moderate, severe, and aggressive pruning.

The third objective is about comparing the effect of recovery performed via QLoRA and DoRA. In addition to being generally less costly to implement than fine-tuning, the two techniques offer quite different approaches to the introduction of trainable parameters. Thus, studying both of them allows for finding out whether or not the specific approach affects how well the adaptation procedure succeeds.

Additionally, we compare the models in terms of their performance in both language modeling and reasoning tasks. For the purpose, WikiText-2 perplexity measures the general capability of a language model, while GSM8K accuracy shows its skills in mathematical reasoning. As they are not equivalent, using both of them helps us have a more consistent picture of the model's pruning impact and recovery performance.

The next objective of the work is to conduct the tests of the models under identical conditions. This is significant as even minor differences in prompt construction, sample size, generation length, and answer extraction may influence the result in case of GSM8K. With a protocol fixed, the comparisons become more valid.

Finally, we measure and compare some practical metrics of efficiency of the models such as peak VRAM consumption, throughput, latency, and the ease of deployment. Taking into account the motivation of deployment to end devices with constrained hardware capacity, the necessity of these measurements becomes obvious.

1.5 Thesis Contributions

The first major contribution of this work is the formulation of a unified empirical framework for investigating LLM pruning and recovery under realistic assumptions. It should be noted that pruning is never treated as an isolated step, but the full experiment cycle – starting with a dense baseline experiment and going all the way through Wanda pruning, raw pruning evaluation, parameter-efficient fine-tuning recovery, and finally comparing with a dense baseline – is used. It allows seeing what kind of degradation results from pruning and what kind of improvements one may reasonably expect after recovery.

The second significant contribution is represented by the comparison of different levels of pruning severity performed by means of Wanda pruning for Llama-3.2-3B-Instruct. The benefit of Wanda pruning lies in its use of both weight magnitude and activation statistics, without requiring full retraining before pruning. The usage of Wanda pruning in different settings with varying degrees of pruning severity makes it possible to analyze the effects that

pruning has on language modeling and reasoning abilities.

A further point is the direct comparison of two recovery methods, namely QLoRA and DoRA, that are performed on Wanda-pruned checkpoints with equal sparsity. Although both of these techniques belong to the PEFT category, their implementation in the process of adaptation is different from one another. QLoRA is based on low-rank adaptation using the base model, loaded in a memory-saving way as a 4-bit representation of the model. On the contrary, DoRA modifies the approach to low-rank adaptation via breaking down weight representation into magnitude and direction. This dissertation investigates the reaction of both approaches to the effect of pruning on language modeling and mathematical reasoning tasks.

The third significant contribution is represented by the usage of both WikiText-2 and GSM8K datasets for evaluating language modeling and reasoning ability respectively. Using this combination of datasets allows disentangling language model degeneration and reasoning degeneration since they do not always coincide.

The fourth contribution is made by performing deployment-oriented analysis, where memory consumption, throughput, and latency are measured alongside the other metrics. This is an important part of the research because the end goal here is not just reaching a more compact model but creating a model that performs better under restricted inference.

Overall, this work makes a major contribution in the form of a structured study of the problem of compressing large language models in terms of the trade-off between compression and performance. One should not think that this paper introduces a new method of pruning; the value of this work lies elsewhere.

1.6 Scope and Assumptions

Pruning and recovery is only performed empirically under restrictive hardware settings in this paper. The base language model considered here is Llama-3.2-3B-Instruct, and the pruning scheme adopted in this research is the Wanda pruning. The methods used to perform recovery include QLoRA and DoRA parameter-efficient fine-tuning. The measures of evaluation used include perplexity on WikiText-2 dataset, accuracy on GSM8K dataset, as well as throughput, latency, and memory consumption.

The chosen benchmarks are believed to provide insights into certain aspects of quality language modeling and mathematics. WikiText-2 acts as a benchmark that measures the ability of language modeling. GSM8K is considered since arithmetic and multi-step reasoning tasks are often vulnerable to pruning operations. Nonetheless, the two datasets above cannot represent all of the abilities of LLMs. Evaluation that involves commonsense reasoning, programming instructions, coding, and question answering is outside the focus of the current work.

"End-device" in this case refers to a practical yet restrictive notion that refers to the

constrained GPU, CPU, or edge-like environment in which the amount of available resources is relatively less than that available in cloud computing servers. In other words, end-devices in this work refer to the environment used for experimental purposes and not production deployments in mobile devices or embedded devices where unstructured sparsity does not guarantee performance speedup unless exploited efficiently by the inference engine and hardware.

It is also assumed that adapter-based model recovery is a feasible and realistic approach to fine-tuning under resource-constrained settings. The recovery process would become too costly when full fine-tuning of all the parameters is applied in this experimental setting. Thus, in order to make the experiment more practical and relevant in the real world, recovery is carried out through lightweight fine-tuning.

1.7 Thesis Organization

Chapter 2 provides an overview of the necessary technical background. This section describes Large Language Models, autoregressive language modeling, transformers, transformer decoder structure, model compression, pruning, Wanda pruning, parameter-efficient fine-tuning, and evaluation metrics. Additionally, it restates the deployment requirements driving the current research.

Chapter 3 provides an overview of related work on efficient LLM inference, pruning techniques, post-pruning deterioration of model performance, recovering a compressed model, quantization techniques, low-rank model adaptation, and evaluation tailored towards deployment. Chapter 3 is intended to set out the context of the work carried out herein, with specific emphasis on the necessity for a baseline–pruning–recovery experiment under a unified framework.

Chapter 4 describes the methodology behind the work. It discusses model selection, dataset selection, baseline evaluation, Wanda pruning, post-pruning evaluation, QLoRA and DoRA recovery, handling adapters, and analysis on the end device level. Chapter 4 describes the approach taken in carrying out this research.

Chapter 5 contains a description of the algorithms, implementation specifics, and computational complexity of the project. Mainly, it deals with the description of the experimental procedure, evaluation algorithms used, and environments used to reproduce the experiments.

26

Chapter 6 contains the experimental results and the discussion thereof. It examines the performance of the dense baseline model, the raw pruned model, and recovered models against the metrics selected for comparison. Other factors considered include recovery ratio, efficiency trade-off, suitability for deployment, and behavioral differences in GSM8K predictions.

Chapter 7 summarizes the key findings from the research carried out in chapters 4 to 6, highlights the limitations, and discusses future plans.

Chapter 2

Preliminaries

2.1 Large Language Models

Large Language Models are neural models capable of predicting and generating text. In this dissertation, in addition to their language capabilities, the emphasis is placed on their computational complexity. Parameters, activations, memory footprint, and inference speed matter for any pruning and recovery task. Therefore, it is important to assess such models not only in terms of accuracy/perplexity, but also in terms of performance and memory footprint.

The vast majority of modern LLMs for text generation purposes are based on the architecture of a Transformer decoder. They generate text tokens in an autoregressive fashion, which means that predictions made at each step are conditioned on previously generated tokens. Although a model may be quite modest-sized in terms of number of parameters, like Llama-3.2-3B-Instruct, performing inference requires going through several model layers in multiple passes. It makes it quite challenging to deploy on resource-constrained hardware.

Llama-3.2-3B-Instruct will be taken as the dense reference model for further experiments. In other words, the purpose here is not to develop a new LLM, but to explore the process of pruning and recovery of a pretrained instruction model using efficient algorithms.

2.1.1 Autoregressive Language Modeling

The autoregressive language modeling formulation factorizes the probability distribution of a sequence of tokens into a chain of conditional distributions as follows:

$$P(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} P(x_t \mid x_{<t}),$$

where $x_{<t}$ denotes all tokens before position t.

At test time, when presented with a prompt, the model predicts one token at a time. The sequential property of generating tokens plays a key role in the deployment of these models

since predicting each token necessitates passing it through the model. This property becomes critical during training on reasoning datasets like GSM8K, where an incorrect prediction in any step can lead to an inaccurate numerical result. As such, pruning should be considered very cautiously, as the pruned model might generate fluent text but cannot reason completely.

2.1.2 Transformer Decoder Architecture

A Transformer decoder architecture is constructed using multiple identical layers composed of self-attention, feedforward network, residual connections, and normalization layers. In the causal decoder setting, the use of the attention mask restricts the token attention so that it sees only past tokens but never future ones. Therefore, decoder architectures alone are suited for autoregressive decoding.

Self-attention and feedforward projections represent a significant part of the model weights. Linear projection layers are equally important candidates for sparsification, since sparse projection layers would reduce the amount of active weights in the model. Nevertheless, weight-level pruning is not an easy task. While a particular weight might look like it carries little value in its magnitude, it should be considered significant when used often enough during the processing of the input.

2.2 Model Compression

The model compression is any algorithm reducing the memory consumption or computational resources for the neural network. It is a problem for LLMs since even 3B-size language model becomes hard to run repeatedly under GPU-memory limitations.

There are many methods of compression, such as pruning, quantization, distillation, and low-rank adaptation. The pruning zeroes some of the weights of a model. Quantization decreases numeric representation precision, like reducing floats to 8-bits or 4-bits. The distillation learns one model called student from another – teacher model. Low-rank adaptation lowers the number of trainable parameters of a model.

In this thesis, the research mostly relies on the approach combining pruning and adapter-based recovery. Pruning is used for obtaining sparse checkpoints, and then QLoRA and DoRA are used to recover some of the model capabilities that were lost in pruning. This workflow is critical since without further fine-tuning the pruned models could become worse at reasoning. Hence, in this research, compression is treated as a whole pruning–recovery workflow rather than an individual operation.

It is also worth mentioning that having sparse weights does not automatically lead to fast execution. In case the inference backend uses dense matrix multiplication operations, having some zero weights does not make the computations much faster. As a consequence, actual

throughput and memory consumption results are presented in experiments.

2.3 Neural Network Pruning

The process of neural network pruning involves either removing or deactivating certain components of the model. Such components include weight matrices, neurons, channels, attention heads, and other elements. The general idea behind pruning is that some of the parameters are less significant for generating a prediction compared to others. Thus, if such parameters can be removed from the network without damaging its functional behavior, this will make the model more compact.

It should be noted that pruning for larger neural networks is different from pruning for smaller networks. Since the representations inside the model are distributed among different layers, attention modules, and feed-forward blocks, pruning will have a detrimental effect on the language modeling, reasoning, or instruction-following capabilities of the model. Therefore, in order to understand if pruning damages the model's capabilities, it is crucial to test the pruned model itself.

In this dissertation, the pruning is done on a pretrained neural network. It is post-training pruning since the model is pretrained with the available Llama-3.2-3B-Instruct checkpoint.

2.3.1 Unstructured Pruning

In unstructured pruning, the values of individual weights are set to zero, while the matrix structure itself stays the same. In other words, when denoting weight matrices by W, pruning zeroes out certain elements of that matrix. Dimensions of the matrix will stay the same, but the total number of non-zero weights will decrease.

This method's benefit is its flexibility: it allows for reaching different levels of sparsity without pruning entire neurons, heads, or layers. In this case, Wanda pruning is applied to model weights using this unstructured approach. The pruned models have 20%, 30%, and 50% sparsity.

However, its drawback lies in hardware constraints. A matrix can be considered sparse; however, that does not mean it will be executed faster in the case of hardware limitations. In this dissertation, no assumptions about speed-up are made regarding unstructured pruning; instead, throughput is measured experimentally.

2.3.2 Structured Pruning

Structural pruning prunes entire structures like attention heads, entire channels, neurons, or even complete blocks. As this form of pruning is modifying the architecture of the computation,

it is more likely to be beneficial for hardware acceleration than pruning. For instance, pruning an entire channel reduces the density of the matrix multiplications.

Nevertheless, structural pruning is more radical since pruning an entire head or a block will result in the loss of functionality that is hard to recover. Structural pruning is not utilized in this dissertation, but it is mentioned here to distinguish it from Wanda pruning, which is more hardware-friendly.

2.3.3 Magnitude-Based Pruning

The magnitude based approach represents one of the most straightforward ways of pruning. In this case, it is assumed that weights with smaller magnitudes have less significance. The measure for such a weight W_{ij} is calculated as follows:

$$I_{ij} = |W_{ij}|.$$

Weights having the least magnitude are eliminated until the desired level of sparsity is achieved.

Although simple and efficient in terms of speed, this technique does not take activations into account. As a result, a small weight assigned to the feature that is actively used by the neural network may still be significant, whereas a big weight associated with an inactive feature will probably be less useful.

2.4 Activation-Aware Pruning

The idea of activation aware pruning is that, along with the weight value, the activation of the corresponding input channel should also be considered. In the linear layer, the output is dependent on both of them. Thus, the pruning score must depend on them as well.

Wanda considers the above fact. After going through a few examples (calibration data) and gathering activation statistics of the target linear layers, the pruning scores can be estimated. This data is never used for supervised learning but only serves to determine the activation of input channels at the time of normal operation of the model.

For the weight W_{ij}, the importance score equals

$$S_{ij} = |W_{ij}| \cdot \|X_j\|_2,$$

Where W_{ij} is the weight value, and X_j indicates the collected activations associated with input channel j. Those weights that have lesser score values are considered less important and thus are pruned based on the specified sparsity ratio.

The importance of the above score lies in its ease of calculation despite limited computing power compared to simple magnitude pruning methods. However, the accuracy of the process of pruning will be affected by the calibration data since the collected activations must reflect the actual inputs.

2.5 Wanda Pruning

The most widely used pruning method in this study is Wanda pruning. This approach has been chosen since it takes into account both the magnitudes of the weights as well as their activations when deciding what weights to prune, making it better suited for pruning LLMs compared to just pruning based on magnitudes alone.

The importance of a particular weight W_{ij} is computed using

$$S_{ij} = |W_{ij}| \cdot \|X_j\|_2,$$

Here W_{ij} denotes the weight value and X_j denotes the activation statistics gathered for the specific input channel. The weights which score low in magnitude are considered less significant and hence are removed until the required sparsity ratio is achieved.

An interesting characteristic about Wanda is that it does not necessitate the need for complete retraining prior to pruning. A limited set of data is allowed to pass through the model, which helps gather activation statistics. These are combined with weight values to generate pruning scores. In this case study, the Wanda approach was utilized to prune the Llama-3.2-3B-Instruct model at three different sparsity ratios: 20%, 30%, and 50%.

The table below details out some pruning methods considered while designing this case study.

Table 2.1: Comparison of pruning strategies

Method	**Importance Signal**	**Main Advantage**	**Limitation**
Magnitude pruning	Weight magnitude	Simple and fast	Ignores activations
Wanda	Weight and activation	No heavy retraining before pruning	Requires calibration data
Structured pruning	Blocks, heads, or layers	More hardware-friendly	May reduce model capacity sharply

For this particular problem, Wanda thus serves as an appropriate middle way because it is better informed than magnitude-based pruning while remaining efficient enough for computation in resource-constrained environments. Yet, it needs calibration information that appropriately reflects the inputs to the model.

2.6 Parameter-Efficient Fine-Tuning

Fully finetuning involves updating the entire set of model parameters, which is inefficient for big language models since gradients, optimizer states, and activations must be kept while training. It would not be efficient on the hardware setup that was used in this dissertation. One possible solution to this problem is PEFT, where all the model parameters except a few are frozen.

For this dissertation, PEFT is performed after the Wanda pruning procedure. In other words, unlike other cases, the adapter here is finetuned on the pruned model and not the dense model initially used. In such a case, the adaptation process requires adaptation to the pruned model.

2.6.1 LoRA

In LoRA, the weight matrix W is frozen and a low-rank modification is learned. The updated weight matrix can thus be expressed as follows:

$$W' = W + \Delta W = W + BA,$$

where both A and B are low-rank trainable matrices. Since only A and B are being trained, there are many fewer parameters involved compared to full fine-tuning. This approach constitutes the foundation of the adapter-based recovery techniques discussed in this thesis.

2.6.2 QLoRA

The QLoRA approach includes low-rank adaptation alongside memory-efficient quantized models loading. Here, the pretrained model is loaded at 4 bits, whereas the LoRA adapter parameters are trained. This approach decreases the memory footprint during the recovering process and makes training of adapters for various pruned checkpoints possible.

In the current workflow, the Wanda pruned model is considered as the static base, whereas QLoRA serves as the trainable adapter that is applied for the model recovery. Finally, the obtained model is assessed in relation to the baseline and pruned model.

2.6.3 DoRA

The proposed DoRA algorithm takes a different path in updating the low-rank decomposition than the conventional LoRA algorithm since DoRA decouples the learning of weight magnitude and weight direction updates. This will be an important technique to learn after considering Wanda since the application of Wanda will alter the structure of the weight matrices by turning

specific weights to zero.

It is essential to include the DoRA algorithm in this research because it will allow one to determine whether using a magnitude and direction-based recovery scheme will outperform other approaches.

2.7 Evaluation Metrics

For evaluating the proposed models, a combination of performance metrics and efficiency metrics will be employed. The reason for doing so is that sometimes a model may perform well with respect to one metric but poorly with regard to others. Thus, we may see a recovered model that does better in perplexity but fails in mathematical reasoning, or vice versa.

The key evaluation metrics chosen for this dissertation include WikiText-2 perplexity, GSM8K accuracy, throughput, latency, and peak VRAM consumption. WikiText-2 is used to gauge language modeling capability, whereas GSM8K is used to evaluate mathematical reasoning capacity.

2.7.1 Perplexity

Perplexity is used to quantify the quality of the predictions made by the model for a certain sequence of tokens. Perplexity can be defined as

$$\text{PPL} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log p(x_i)\right),$$

where $p(x_i)$ denotes the predicted probability of the token x_i.

Perplexity will be calculated using the WikiText-2 dataset. Perplexity can tell us whether pruning has harmed the next-token prediction capability of the model or not, and whether PEFT has helped in that regard.

2.7.2 Task Accuracy

Accuracy refers to the ratio of correct outputs against a set benchmark. In the case of this dissertation, accuracy of GSM8K is applied in measuring mathematical reasoning ability. The model produces an answer for each question presented, and this answer is compared to the gold standard answer.

Accuracy is calculated as

$$\text{Accuracy} = \frac{\text{Number of Correct Answers}}{\text{Total Number of Questions}}.$$

However, this method has strict standards since only the final answer that is extracted from the numbers is used. If the explanation is perfect but the final answer is wrong, the question is marked as incorrect. This is why GSM8K is suitable for spotting degraded reasoning due to pruning.

2.7.3 Efficiency Metrics

The efficiency metrics have been included in this paper due to the nature of work, where the topic of recovery has been discussed in the context of limited computing resources. Accuracy and perplexity alone cannot demonstrate if the recovery is efficient enough to use the recovered model practically.

The peak VRAM indicates the highest amount of GPU memory used while evaluating or recovering the model. Throughput indicates the rate of token generation per second. Latency represents the time taken by the system to generate responses.

2.8 End-Device Deployment Constraints

For the purposes of this dissertation, deployment of an end-device denotes the execution or testing of the models subject to restricted amounts of memory and processing power without the requirement that such devices be either a mobile phone or embedded computer.

The key limitations of memory, latency, throughput, storage, and accuracy will be considered since compression is valuable only if the resulting model still functions practically. Notably, unstructured pruning does not ensure runtime improvements unless the inference engine/hardware supports sparse computations.

Table 2.2: Deployment constraints considered in this dissertation

Constraint	**Meaning in this work**
Memory	Peak VRAM or RAM needed during inference or fine-tuning
Latency	Time required to generate output tokens
Throughput	Tokens generated per second
Storage	Disk size of model checkpoints and adapters
Accuracy	Retained task performance after compression and recovery

A sensible compression scheme would need to achieve an appropriate balance between these aspects, and not simply aim to maximize any of them in isolation. Extreme levels of sparsity become meaningless if there is a significant loss in accuracy, while the benefit of recovery ceases to exist if the gain in efficiency is negated.

Chapter 3

Literature Review

3.1 Efficient LLM Inference

Efficient inference is a core issue of LLM deployment beyond the realm of servers where billions of parameters are stored. Indeed, besides memory for the weights, a lot of memory is required to accommodate activations, key-value cache, as well as many repeats of inference to generate tokens one-by-one. The cost of inference thus becomes essential even when the model fit memory-wise.

A number of lines have been tried to achieve efficient inference for LLMs. One approach is quantization that decreases numerical precision, e.g., reducing the weights from FP16 or BF16 to 8-bit and 4-bit numbers. This technique allows for more efficient model storage and faster loading onto smaller GPUs. QLoRA operates on the same idea and keeps the pretrained model quantized while training low-rank adapters.

Pruning is also a well-known way to reduce model size, making it smaller or sparse by removing particular weights or other structures from the model architecture. Yet pruning is trickier when applied to LLMs, since their representations are distributed throughout layers to a great extent. Thus, removing arbitrary weights without taking into account their importance for calculations may lead to loss of language modeling and reasoning behavior.

Parameter-efficient adaptation techniques can be used to achieve efficient LLMs as well. LoRA technique freezes the base model, allowing to train only low-rank parameter adapters, while DoRA expands the idea by decoupling weight representation into magnitude and direction. Both of them do not compress the base model per se but make adaptation to new tasks cheaper.

This dissertation aims to address efficient inference through the combination of several aspects. Model size, sparsity level, adapters costs, memory footprint, throughput, and task-specific qualities should all be addressed simultaneously. No point in having a small-sized or sparse model which cannot perform reasoning or lose a lot of reasoning ability. Also, a sparse model alone does not mean efficient inference backend.

3.2 Pruning of Large Language Models

Pruning has been employed for compressing neural networks for quite some time now. However, the application of pruning in LLMs requires careful consideration since it involves deciding on the components of a model with millions/billions of parameters whose removal will not affect the output too much. The approach to pruning involves identifying weights/components that do not significantly contribute to the output.

The easiest form of pruning is magnitude pruning. As suggested by its name, magnitude pruning entails getting rid of components that have lower magnitudes compared to the rest. While this approach seems straightforward, it does not take into account the contribution of small components to the output when they are connected to input channels that are active often. For instance, in Transformer models, activation patterns differ from layer to layer, token to token and input data point to data point.

Unlike magnitude pruning, SparseGPT addresses the problem of pruning from the perspective of reconstructing the layer output using the available component. Second order information is utilized to determine whether removing a component will make the reconstructed output similar to the original one. Thus, pruning with second-order information provides a deeper understanding of pruning than simple magnitude pruning, albeit at an increased cost. SparseGPT proved that GPT-style models can be pruned in one shot[3].

Wanda adopts a more straightforward activation-aware approach. In contrast to SparseGPT, Wanda does not need any Hessian inversion, weight reconstruction, or re-training after the model pruning process. The importance of each weight is evaluated through the formula below

$$S_{ij} = |W_{ij}| \cdot \|X_j\|_2,$$

where W_{ij} refers to an individual weight, while X_j denotes the activation statistics of the respective input feature.

This importance calculation procedure is particularly pertinent to the current dissertation since Wanda is used at the pruning stage of our workflow. This pruning technique evaluates the importance of a weight according to its magnitude and activation statistics but without adding too much computational overhead for generating pruned models at different levels of sparsity, e.g., 20%, 30%, and 50%[6].

In structured pruning, the pruned unit will typically be bigger, like an entire attention head or neuron or even an entire block. While this approach might be more favorable for hardware implementation due to changing the dimensions of the dense matrix, it might also tend to discard useful capacity. The LLM-Pruner paper takes a cue from this concept, where it discovers dependent structural units and then prunes them followed by LoRA recovery [5]. However, in our case in this dissertation, Wanda-based unstructured pruning is chosen.

3.3 Post-Pruning Performance Degradation

The pruning process affects the computation of a model that has been pre-trained. If pruning eliminated only the redundant weights of the pre-trained model, the result will not differ significantly. However, useful weights might get eliminated too in the case of high sparsity. As a result, pruned LLMs might demonstrate degradation of their performance.

There are various types of model performance degradation resulting from pruning. First, the model may have problems with language modeling; this will be shown by an increased level of perplexity. Secondly, it might lose its ability to perform mathematical operations correctly. For example, it might not solve problems related to addition and subtraction in the case of GSM8K. Thirdly, it might not consider one of the conditions given in the prompt, and fourthly, it might give an answer that does not correspond to its reasoning.

Therefore, pruning should be measured before attempting any type of recovery from its adverse effects. Otherwise, it would not be known which part of the model behaviour remained after pruning and what has changed during the subsequent recovery process. Thus, in this dissertation, a three-step comparison is used: the dense baseline, the pruned model, and then the recovered pruned model.

The level of performance degradation also depends on the level of sparsity. Pruning that is too mild will leave the model intact, while aggressive pruning will eliminate useful model capacity. This is the reason why multiple sparsity levels were chosen for this experiment.

Finally, recent studies show that pruning quality should not be estimated based on the pruning mask alone. It might have a great impact on the recovery process [7], which makes further investigation necessary.

3.4 Recovery of Compressed Models

Recovery is necessary since unpruned fine-tuning may degrade both the language modeling performance and the reasoning capacity. The straightforward solution for recovery is full fine-tuning, but full fine-tuning is very costly in terms of memory when it comes to LLMs, as all the model weights, gradients, optimization states, and activations need to be stored. Such a solution is impractical for our case, where the target architecture of hardware is limited.

Parameter-efficient tuning provides a more efficient way for performance recovery. The idea behind parameter-efficient tuning is that, instead of fine-tuning the whole model, we tune only several parameters called adapters and keep the rest of the model almost frozen. In the context of the post-pruning fine-tuning, adapters are fine-tuned specifically for the pruned version of the model. Thus, they should make up not only for the task-specific adaptation, but also partially compensate for the loss introduced by pruning. Recent research in the restoration of pruned LLMs [2] confirms our assumptions about the usefulness of such recovery approaches.

LoRA is one of the widely used methods for parameter-efficient tuning. It approximates the change in the weight matrix using low-rank matrices:

$$W' = W + \Delta W = W + BA,$$

where A and B are learnable low-rank matrices. As only these matrices can be updated, the number of trainable parameters is far less than full fine-tuning, making LoRA-style recovery well-suited for large language models where memory is restricted.

The QLoRA technique goes one step further by utilizing 4-bit precision for loading the base model while training LoRA adapters, saving memory while recovering the model and allowing more iterations in training the adapter. While DoRA changes the way that LoRA works by decoupling magnitude from direction, DoRA might act differently because the base model weights would already be adjusted through pruning.

It cannot be taken for granted that recovery is always successful. For example, a recovery method may perform well on sparse models but poorly on highly-sparse models. Thus, this thesis will assess QLoRA and DoRA on various Wanda-pruned checkpoints and compare their recovered models using WikiText-2 perplexity and GSM8K accuracy. Furthermore, the recovered model must outperform the raw pruned model, retain meaning compared with the dense baseline, and even make sense in terms of efficiency.

3.5 Quantization and Low-Rank Adaptation

Both quantization and low-rank adaptation are useful for enabling recovery in light of constraints on GPU memory. The former enables reductions in the numerical precision of storing weights, while the latter enables reductions in the number of parameters that are trainable via fine-tuning. Both concepts are directly relevant to our work since recovery is based on fine-tuning Wanda-pruned Llama-3.2-3B-Instruct checkpoints, not on fine-tuning itself.

The importance of QLoRA lies in the fact that it leaves the base model in 4-bit quantized state, training only LoRA adapter weights. Various methods for reducing memory consumption are employed in fine-tuning processes; these include NF4 quantization, double quantization, and paged optimizers [1]. As a result, QLoRA makes recovery of multiple pruned checkpoints possible in experimental setups constrained by limited GPU memory.

A block-wise quantization approach can be defined as

$$X_{\text{Int8}} = \text{round}\left(\frac{127}{\text{absmax}(X_{\text{FP32}})} X_{\text{FP32}}\right),$$

The dequantization process, on the other hand, approximates the floating-point representation for calculations:

$$\text{dequant}(c, X_{\text{Int8}}) = \frac{X_{\text{Int8}}}{c}.$$

Quantization is an effective approach that helps save memory, but it may also create some approximations due to the use of integers for storing real values. This is especially true when we consider the language model application because outliers in the weights distribution could influence the scaling value. That is why QLoRA is not only useful in fine-tuning but also in memory savings.

LoRA works on a completely different principle as well. Instead of adjusting the entire pre-trained weight matrix, it adjusts it using low-rank matrices:

$$W' = W_0 + \Delta W = W_0 + BA,$$

where B is of dimension $d \times r$ and A is of dimension $r \times k$. The original set of weights is kept frozen, thus reducing optimizer memory space and gradient storage.

However, DoRA deviates from this idea and represents the pretrained weights as a combination of its magnitude and direction. This representation can be shown as

$$W = m\frac{V}{\|V\|_c} = \|W\|_c\frac{W}{\|W\|_c},$$

Here, m is the magnitude component, and V is the directional component. In DoRA, an LoRA-like update is used for the direction, whereas the magnitude component can also be updated. The updated weight can be expressed as

$$W' = m\frac{V + \Delta V}{\|V + \Delta V\|_c} = m\frac{W_0 + BA}{\|W_0 + BA\|_c}.$$

This expression becomes meaningful after trimming because Wanda modifies the weight distribution by fixing some of them to zero. An additive rank-k approximation can restore some lost behavior, but magnitude-direction approximation will generate another pattern of restoration. As a result, this dissertation experimentally compares QLoRA and DoRA methods instead of presuming that one approach is better than the other beforehand. DoRA is an extension of LoRA where the pretrained weights are decomposed into magnitude and direction [4].

3.6 Deployment-Oriented Evaluation

While most LLM evaluations center around their accuracy or perplexity, there are other factors to consider when deploying an LLM. The LLM must be loadable, testable, and operational under constraints on memory and time. In this dissertation, this is particularly important since the aim is not only to restore accuracy but also to establish whether the restored model has any

meaning under the constraint of available hardware.

Task accuracy is measured using GSM8K for mathematical reasoning, while WikiText-2 perplexity tests language modeling quality. In addition, peak VRAM, throughput, latency, and the overall model/adapter size are taken into account as well. By considering these criteria, we ensure that we do not reach a conclusion by solely looking at accuracy or perplexity alone.

The set of relevant performance measurements is outlined in the table below.

Table 3.1: Deployment-oriented evaluation indicators

Indicator	**Purpose in this dissertation**	**Preferred Direction**
Peak VRAM	Measures maximum GPU memory required during evaluation or recovery.	Lower
Throughput	Measures generated tokens per second during inference.	Higher
Latency	Measures time required for generation under a fixed setting.	Lower
Perplexity	Measures language modeling quality on WikiText-2.	Lower
GSM8K Accuracy	Measures final-answer correctness on mathematical reasoning problems.	Higher
Model/Adapter Size	Measures storage burden of checkpoints and recovery adapters.	Lower

The reason is that unstructured sparsity does not guarantee any improvement in wall-clock performance due to the possibility of a large number of zeros residing in dense matrices when performing inference with such sparsity. Structured/hardware-aware sparsity, however, might yield better performance results if hardware-supported.

Therefore, we should take into account more than just accuracy improvement. Recovery should allow for better performance than the base pruned model, yet keep the overhead reasonable enough to make deploying the model feasible in practice. The sweet spot is somewhere between recovering performance and paying for the deployment cost.

3.7 Research Gap

In terms of existing compression approaches, great progress has been achieved in such topics as pruning, quantization, and PEFT. However, they are mostly studied in isolation from each other while the actual usage case can involve all three aspects simultaneously. A model could be pruned well but still require recovery, and an approach which succeeds on a dense model might work quite differently on a pruned checkpoint.

Firstly, a unified post-pruning recovery pipeline needs to be provided in this dissertation. It is common to study pruning quality or adapter-based tuning in isolation; the dissertation, in turn, takes a predetermined sequence of actions – evaluation on the dense baseline, Wanda pruning at different levels of sparsity, evaluation on a raw pruned model, QLoRA/DoRA recovery,

and final evaluation.

Secondly, behavior of evaluation metrics needs to be considered separately. Improvement in WikiText-2 perplexity does not automatically translate into better results on GSM8K and vice versa. Therefore, both WikiText-2 and GSM8K should be used to avoid conflating language modeling quality with mathematical reasoning ability.

Thirdly, deployment performance needs to be assessed properly rather than reporting just the degree of sparsity and/or parameter reduction. For example, it will take special care for unstructured sparsity to provide speedups which directly reflect the sparsity ratio.

Finally, different approaches for adapter recovery after pruning need to be compared to each other. In particular, QLoRA is interesting for reducing memory consumption during the recovery process while DoRA provides decomposition by magnitudes and directions which can help with recovery from pruning.

This dissertation considers activation-aware LLM pruning [6], memory-efficient adapter recovery [1], and weight-decomposed low-rank adaptation [4] under a consistent evaluation framework.

3.8 Positioning of This Work

This research is situated at the crossroads of LLM pruning, parameter efficient recovery, and constraint deployment approaches. The proposed solution does not suggest a novel approach to either pruning or designing an adapter architecture. Instead, the paper explores the behavior of known techniques when being integrated into one experimental pipeline.

1 LLama-3.2-3B-Instruct is chosen as the underlying model. Wanda pruning has been selected due to the fact that it allows applying activation-aware post-training pruning without pre-pruning full retraining. QLoRA and DoRA have been chosen as recovery techniques due to their parameter-efficiency yet different forms of adaptation.

The proposed experimental pipeline can be expressed as

$$M_s = P_{\text{Wanda}}(M, s),$$

where M is the original LLM, s is the sparsity rate, and P_{Wanda} is the Wanda pruning process. The recovery process is carried out according to

$$M_s^{\text{rec}} = R_{\text{PEFT}}(M_s, D_{\text{train}}),$$

where D_{train} is the dataset used for recovery training.

This will help analyze three possible states of models – the dense one, the pruned Wanda model, and the recovered PEFT model.

Chapter 4

Proposed Methodology

4.1 Overview of the Proposed Framework

The methodology used for this work involves a pruning-recovery pipeline instead of the compression method. This method starts by taking a highly parameterized LLM that undergoes evaluation through an immutable framework. Then, Wanda pruning is implemented for particular levels of sparsity, followed by evaluation of the pruned checkpoint before recovering parameters to establish the degree of useful performance possible. The procedure adopted is of significant importance because recovery performance becomes relevant only if the performance of the original baseline and the pruned LLM have been quantified.

There are three main steps in the pipeline. Suppose M represents the original pretrained LLM and s represents the sparsity level desired. Wanda pruning results in a sparse model denoted as

$$M_s = P_{\text{Wanda}}(M, s),$$

where P_{Wanda} is the activation-aware pruning operation. The pruned model M_s is evaluated immediately to record the damage caused by pruning. Recovery then follows using a parameter-efficient method:

$$M_s^{\text{rec}} = R_{\text{PEFT}}(M_s, D_{\text{train}}),$$

where R_{PEFT} is either QLoRA or DoRA recovery and D_{train} is the recovery training dataset. Finally the original model M, the raw pruned model M_s, and the recovered model M_s^{rec} are compared under the same evaluation settings.

Figure 4.1 summarizes the complete experimental pipeline used in this dissertation. The baseline dense model Llama-3.2-3B-Instruct is tested initially, followed by pruning at three different sparsities: 20%, 30%, and 50% with Wanda. Then the pruned but uncorrected checkpoints are tested before any recovery operation, and finally, the QLoRA and DoRA adapters are fine-tuned and combined with the respective pruned base models.

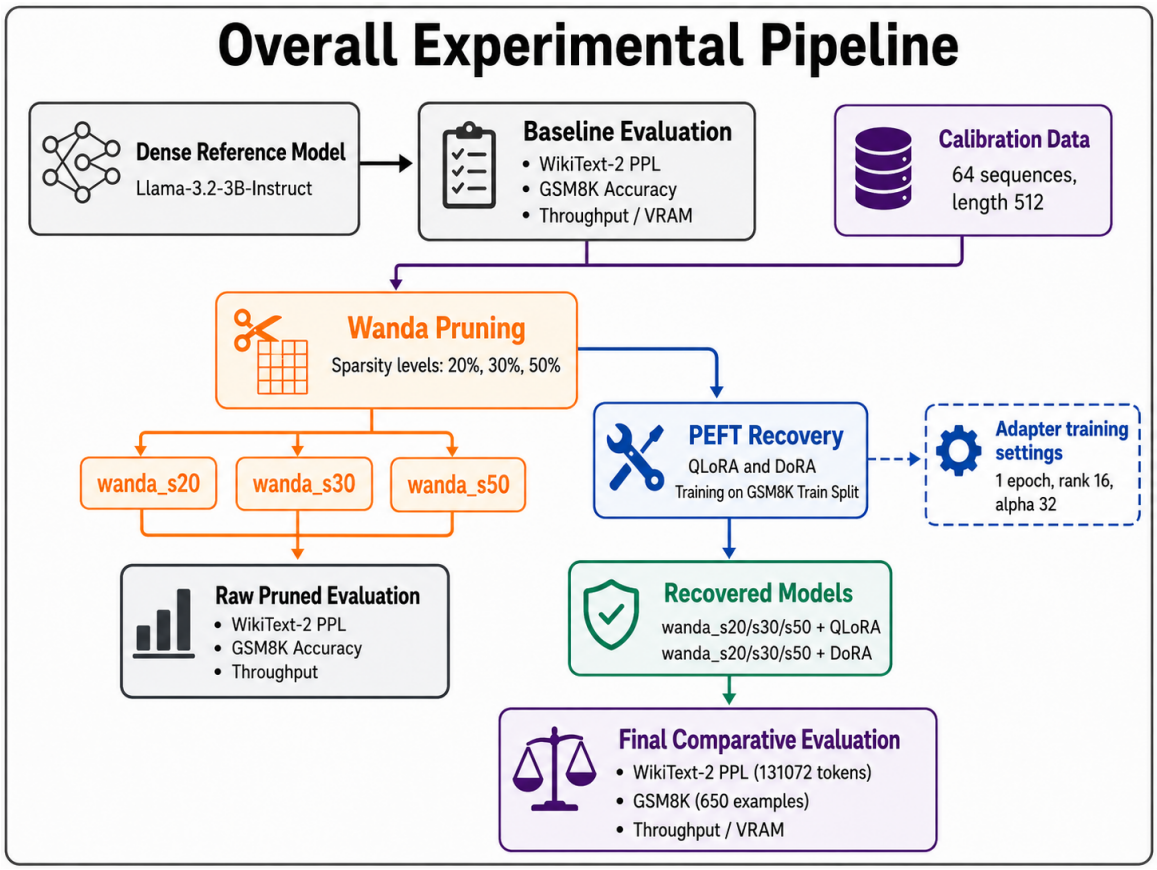

Figure 4.1: Overall experimental pipeline followed for pruning, recovery, and evaluation of Llama-3.2-3B-Instruct.

The unaltered, pruned model takes a significant place in this scheme. Without Ms measurement, it would be impossible to distinguish between deterioration caused by pruning and any potential improvement through subsequent recovery. If a restored model demonstrates high quality of performance, the interpretation of this result should take into account how much the initial deterioration was. The poor quality of the restoration process does not automatically imply the ineffectiveness of adaptation; on the other hand, it might be because the sparsity chosen is simply too large for the model.

Another aspect that shapes the design is related to implementation. Precision alone is not enough – if a pruned model is meant for limited hardware, it must be tested on parameters such as memory usage, speed, and runtime behaviour. Hence, the proposed approach combines an evaluation of the performance of the model with the analysis of its efficiency. The critical question to be asked is whether it is possible to bring the usefulness of the LLM back without undermining the reasons for pruning.

4.2 Model Selection

The experiments used the Llama-3.2-3B-Instruct base model. 3B-scale models were chosen as a good tradeoff between having enough size to learn interesting things about pruning and recovery while being feasible to fit into GPU memory and evaluate in a reasonable amount of time. Bigger models would make it impractical to perform repeated rounds of pruning and recovery at different sparsity ratios and using two different methods. Meanwhile, much sparser models will fail to demonstrate the deployment problems typical for LLMs.

At the same time, a model of this size fits well into the scope of the research focused on end-devices. While larger models may be a more likely candidate for deployment on a device with limitations, models of this size already require proper memory management without being overly large. They are too big to be run effortlessly, but they can fit a small GPU. Thus, Llama-3.2-3B-Instruct is a good middle ground for this study.

In this study, the model is viewed as a pretrained base model. The goal is not to train an LLM from scratch. The focus is on achieving certain results when you have a pretrained checkpoint available. This makes it useful for exploring real research cases where it is impossible to train an LLM from scratch, but pruning and adaptation are still viable.

To start, the dense model was evaluated before any pruning took place. This is done in order to create a basis for comparison. Once pruning was completed, the model was assessed both in relation to the raw pruned checkpoint and to the dense version of the model.

Table 4.1: Base model configuration

Component	Description
Dense reference model	Llama-3.2-3B-Instruct
Model family	Decoder-only transformer language model
Approximate scale	3 billion parameters
Model usage in this work	Dense reference model for Wanda pruning, PEFT recovery, and deployment-oriented evaluation
Pruning method applied	Wanda activation-aware pruning
Recovery methods applied	QLoRA and DoRA
Primary evaluation datasets	WikiText-2 and GSM8K
Main deployment indicators	Peak VRAM, throughput, latency, and model/adapter storage

4.3 Dataset Selection

The selection of the two datasets is motivated by their ability to characterize two different properties of the model. WikiText-2 is selected as the evaluation dataset for evaluating the performance of the model on language modeling using perplexity. GSM8K is used as the dataset for evaluating the mathematical reasoning capability of the model using final-answer accuracy. The reason behind choosing the two datasets is that one is useful for characterizing

the model's capability of language prediction, whereas the other is useful for the reasoning capability of the model.

This is necessary due to the fact that pruning does not necessarily affect every property of the model equally. A model can continue predicting text fluently while having a degraded reasoning capability. It may show relatively small changes in perplexity but much larger drops in the accuracy of reasoning. Having both datasets gives a balanced view regarding how much damage has been caused by pruning and whether recovery can be achieved.

Besides evaluation datasets, Wanda also needs some calibration datasets. This is because the Wanda pruning algorithm uses the activation values at the inputs to calculate the importance of each weight. The activation values are calculated by using text sequences. However, this data is not used for supervised learning in pruning.

4.3.1 WikiText-2

WikiText-2 is used for testing the performance of language models. The model operates on text sequences, and the accuracy of its predictions can be measured via perplexity, the lower the value of which, the more probable it is that the model predicts the right tokens from the test text.

For our experimental setup, WikiText-2 will primarily be used for estimating the influence of pruning on the overall prediction capability of the model. The model loses some weights, which could make its inner representation unstable and, hence, result in increased perplexity. After recovery, a decrease in perplexity would show that the model recovers its language modeling abilities to a certain extent.

Perplexity is calculated using the formula below.

$$\mathrm{PPL} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log p(x_i)\right),$$

where N is the number of evaluated tokens and $p(x_i)$ is the probability assigned to the correct token x_i. In the experiments the same evaluation script and token processing setup are kept fixed across dense, pruned, and recovered models so that the comparison remains fair.

4.3.2 GSM8K

GSM8K acts as a benchmark to test the mathematical reasoning ability of the model. This benchmark dataset consists of math word problems that require understanding of the problem, some intermediate reasoning process, and providing an end numerical answer. In the current research paper, GSM8K is especially useful since reasoning problems are known to be sensitive to pruning. Thus, although a solution generated by the model may be coherent, any arithmetic

mistake or confusion about a quantity will result in an inaccurate final solution.

The model's accuracy is measured by the correctness of its final solution. The model provides a solution for a specific problem, and the final answer is evaluated against the true value. Accuracy is calculated as:

$$\text{Accuracy} = \frac{\text{Number of Correct Answers}}{\text{Total Number of Questions}}.$$

A fixed sample size, seed, prompt structure, generation length, and extraction mechanism of the answers is used whenever possible. This is crucial due to the potential influence of small variations in the implementation process on the evaluation of GSM8K. For example, slight changes in the prompt format and maximum generation length could lead to different numbers of correct answers. Since the current study involves comparing three models, the priority should be given to consistency rather than personalization of the prompts.

4.3.3 Calibration Data for Pruning

Wanda pruning requires calibration data to estimate activation statistics. During calibration a small number of input sequences are passed through the model, and the input activations of linear layers are collected. These activations are then used in the Wanda importance score

$$S_{ij} = |W_{ij}| \cdot \|X_j\|_2,$$

where W_{ij} is a weight and X_j represents the activation values corresponding to the input channel j. Weights with lower scores are pruned according to the target sparsity level.

This information is significant since Wanda does not judge the importance of a weight based on its size alone, but also considers how active its corresponding input is. This relationship makes the process of pruning become more consistent with how the model computes. At the same time, calibration is based on the presumption that the selected examples are representative of the kind of text the model deals with. If this information turns out to be too specific, then the statistics about activations will prove to be misleading.

4.4 Baseline Evaluation

Baselining formed the first step in the experiment procedure. The densely connected Llama-3.2-3B-Instruct model was evaluated against the benchmark datasets without applying any form of pruning and recovery. This formed the basis of comparison throughout the experiment. Without baselining, it would not have been easy to ascertain whether pruning significantly degraded the model's performance or whether recovery improved its performance back to significant levels.

Baseline performance testing involved the use of different performance measures. WikiText-2 perplexity was used to measure language modeling capabilities, while GSM8K accuracy was used to evaluate reasoning capabilities. Practical performance measures such as generated tokens, throughputs, latencies, and peak VRAM were also collected when possible.

Table 4.2: Baseline evaluation template

Metric	**Purpose**
WikiText-2 Perplexity	Measures general language modeling quality.
GSM8K Accuracy	Measures mathematical reasoning through final-answer correctness.
Correct Answers	Counts the number of correctly solved GSM8K problems.
Total Generated Tokens	Records the total number of generated output tokens during evaluation.
Throughput	Measures generated tokens per second.
Peak VRAM	Measures maximum GPU memory usage during evaluation.
Latency	Measures response generation time under a fixed setting.

In another one, the same evaluation procedure was used throughout the entire work. Namely, the same scripts were applied to evaluate dense, pruned, and recovered models under the same conditions, whenever possible. Such a choice helps address a common problem in experiments, which is that differences in the result obtained at various stages can be attributed not to real improvement or worsening of model performance but merely to changes in the experiment setting.

Concerning GSM8K, the baseline experiment was conducted with a set amount of examples and the same random seed. The algorithm of answers' extraction was consistent among all models. Similarly, for WikiText-2, the process of text splitting and tokenization was the same. While these considerations may seem trivial at first glance, they matter within the context of a comparative study since the difference in a few answers in the GSM8K data set can affect the conclusions, especially considering the limited size of the subset.

Finally, the baseline stage also gave an idea of the original deployment cost of the model before the experiment. Specifically, the memory footprint and the speed of generation were determined, and then these metrics could be used as a basis to understand whether the process of pruning and recovering has positively affected the compatibility of the model with low-end hardware.

4.5 Wanda Pruning Methodology

After baseline evaluation, Wanda pruning was applied to the dense model. I selected Wanda because it is a post-training pruning method that uses both weight magnitude and activation information. This fit the project setting, where full retraining before pruning was not practical

under the available compute budget. Wanda instead allowed pruning through a calibration set and a relatively lightweight activation collection process.

The pruning was applied mainly to the linear layers of the Transformer model. These layers contain a large portion of the model parameters, so they were natural targets for sparsification. The first embedding layer and final output-related components were treated carefully, since aggressive pruning in such parts may directly disturb token representation or output prediction. The pruning flow was layer-wise: collect activations, compute importance scores, select low-scoring weights, and set those weights to zero according to the target sparsity.

In this study, pruning was not treated as the final solution. It was the compression stage that created sparse models for later recovery. This distinction is important. A raw pruned model may lose language modeling quality or reasoning ability. Because of that, every pruned checkpoint was evaluated before recovery, so the damage caused by pruning remained visible and measurable.

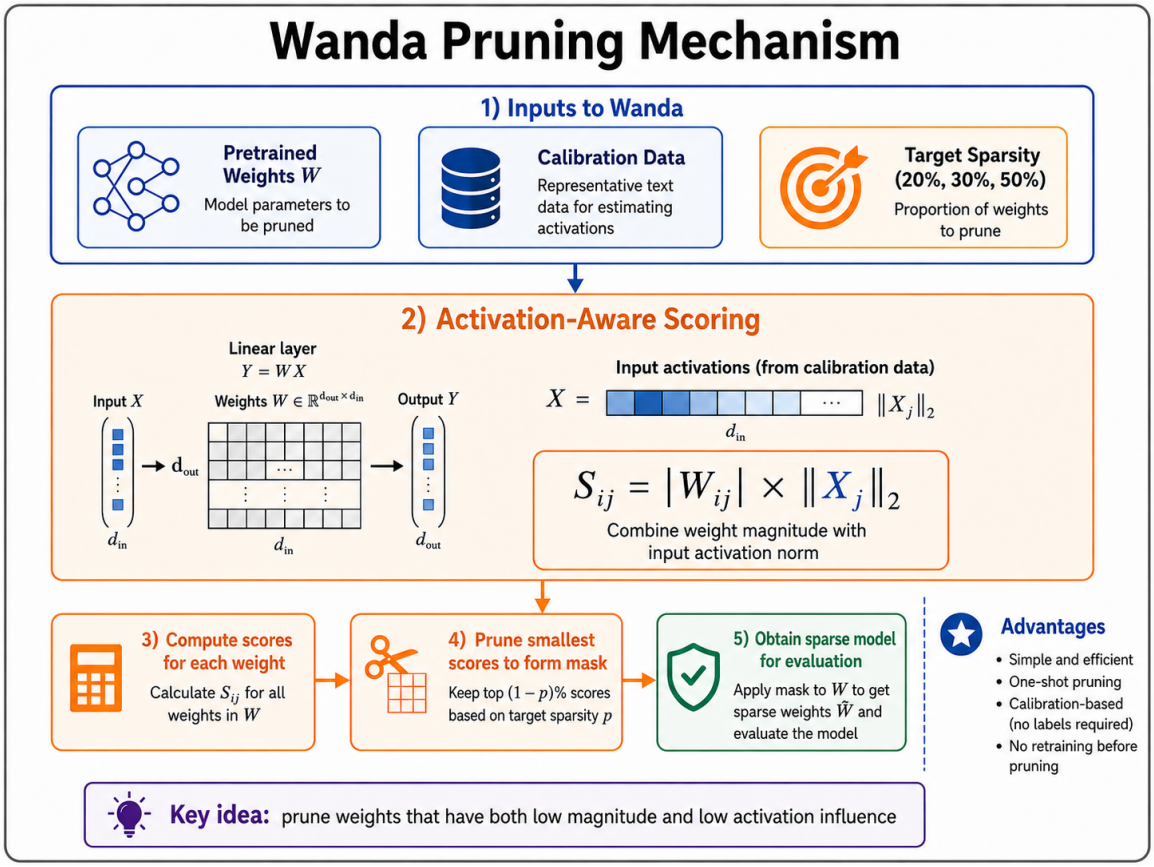

Figure 4.2: Wanda pruning mechanism used to compute activation-aware importance scores and generate sparse pruned checkpoints.

Figure 4.2 illustrates the Wanda pruning mechanism used in this work. The algorithm first uses a limited dataset for propagation within the network to obtain the activation statistics. Wanda then computes the importance of each connection based on the product of its weight magnitude and activation norm. Connection weights that are less important are pruned depend-

ing on the desired sparsity, while more important connections are preserved.

4.5.1 Activation Collection

The reason why activation statistics are necessary for Wanda is that, apart from weights themselves, the pruning algorithm needs information about how frequently the respective input channels are activated. When collecting activations, the model was fed several text sequences and the activations on input for the target linear layers were collected.

In this particular implementation, it is important to note that the size of the activation dataset was small enough so that activation collection was feasible under memory limitations of the GPU. For reproducibility purposes, the number of sequences and their length were defined in advance, and they remained constant for all variations of Wanda pruning.

It is important to note that these sequences were not used as supervised fine-tuning data. The learning process involving label information was not yet implemented. These examples were collected only to estimate the active hidden input space during regular inference passes.

4.5.2 Importance Score Computation

For the target linear layers, Wanda assigns an importance value to each weight considering both weight magnitude and input activation norms. The formula for the importance is given by

$$S_{ij} = |W_{ij}| \cdot \|X_j\|_2,$$

where W_{ij} is the weight connecting input channel j to output channel i, and X_j represents the collected activation values for the corresponding input channel. The term $\|X_j\|_2$ captures how strongly that input feature is used across the calibration samples.

The importance calculated in this manner holds great significance since only the weight magnitude may not correctly evaluate the importance. For instance, a weight which is smaller in magnitude but has very high input activation values and frequent activity might play a significant role, whereas a bigger weight having less activity might have lesser importance during actual inference.

4.5.3 Sparsity Levels

The investigation uses several sparsity levels instead of using only one. It is justified due to the fact that the degree of pruning greatly affects performance loss and the complexity of recovery. At low levels of sparsity, pruning allows retaining more behavior of the model, while at high levels of sparsity, pruning results in more compactness. However, the extent of the damage caused to the model is harder to address.

The pruning configurations used in this project are summarized in Table 4.3.

Table 4.3: Pruning configurations

Configuration	Sparsity	Pruning Method	Recovery Applied
wanda_s20	20%	Wanda	QLoRA / DoRA
wanda_s30	30%	Wanda	QLoRA / DoRA
wanda_s50	50%	Wanda	QLoRA / DoRA

Three scenarios provide opportunities for comparison between moderate, moderate-to-aggressive, and aggressive pruning. In case of 20%, pruning should allow keeping more capabilities of the original model. Scenario of 30% serves as the median one to understand better whether pruning damage and its recovery have been possible. Finally, scenario of 50% can be characterized as the most radical scenario that will help to investigate the possibilities of recovery.

4.6 Post-Pruning Evaluation

After each pruning of a Wanda-trained model, a pre-recovery evaluation took place before any sort of recovery attempt. This preliminary process ensures isolation of the effect caused by the pruning. Ignoring this step could lead to an inability to interpret the results, as it might be difficult to know whether the adapter managed to recover enough performance or had simply optimized the model to approach the baseline.

For the evaluation after pruning, identical datasets and process were used to those in the baseline phase. For measuring the effect of pruning on language modeling, the perplexity score was calculated on the WikiText-2 dataset. The accuracy on the GSM8K dataset was measured to determine the effect on math reasoning. Throughput and maximum VRAM usage are reported whenever possible.

The central comparison at this point is

$$M \to M_s,$$

where M is the dense model and M_s is the pruned model at sparsity level s. The difference between their results shows the performance degradation caused by pruning. Later, after recovery, the comparison becomes

$$M \to M_s \to M_s^{\text{rec}}.$$

The following three-stage approach outlines the primary experimentation process used in this paper: first, identify how well the dense model performs; second, quantify the level of impairment caused by pruning; and lastly, determine the extent to which that impairment can

be recovered using PEFT methods.

However, this pruning-induced damage may vary significantly among different evaluation criteria. For instance, while perplexity would reduce slowly, GSM8K accuracy may fluctuate more irregularly. Reasoning-oriented tasks depend on multistage generation of outputs and consistent final answers. The pruned model might still generate coherent text but be incapable of performing arithmetic, neglecting some conditions, or calculating the wrong answer. Therefore, post-pruning evaluation does not serve as an ordinary benchmark; instead, it demonstrates what abilities have been impaired by the compression process.

4.7 Recovery Using PEFT

Upon the assessment of the unrefined pruned models, recovery is carried out through parameter-efficient fine-tuning. It is worth noting that full fine-tuning is not used because it requires updates for all the parameters of the model, and it is simply too costly, especially within the confines of the current hardware resources. More importantly, this approach goes against the practical aim of the study, in which efficiency in terms of recovery is prioritized over re-training.

In this case, parameter-efficient fine-tuning approaches prove to be more applicable since they involve minimal changes to the base model by keeping it mostly frozen while training just a few new parameters. In particular, the recovery methods used here are QLoRA and DoRA. Both approaches belong to the adapter-type techniques; however, they differ in how they represent their update process. The comparison of these two will reveal whether the recovery outcome depends on the choice of adapter type, aside from pruning, that is.

For each pruned model M_s, recovery is performed as

$$M_s^{\text{rec}} = R_{\text{PEFT}}(M_s, D_r),$$

where D_r denotes the recovery training data and R_{PEFT} denotes the selected recovery method. The recovered model is then evaluated using the same protocol as the baseline and raw pruned models.

4.7.1 QLoRA Recovery

QLoRA recovery involves low-rank adaptation where the base model remains frozen and memory usage is optimized. QLoRA recovery comes in handy in this research because it allows one to conduct the recovery experiment without having to pay for the associated memory expenses that come with fine-tuning. The pruned model acts as the fixed backbone while the adaptable weights get modified.

The LoRA-style update can be written as

$$W' = W + \Delta W = W + BA,$$

where W is the frozen base weight, and A and B are low-rank trainable matrices. Since only these adapter matrices are trained, the number of trainable parameters is much smaller than the full model size.

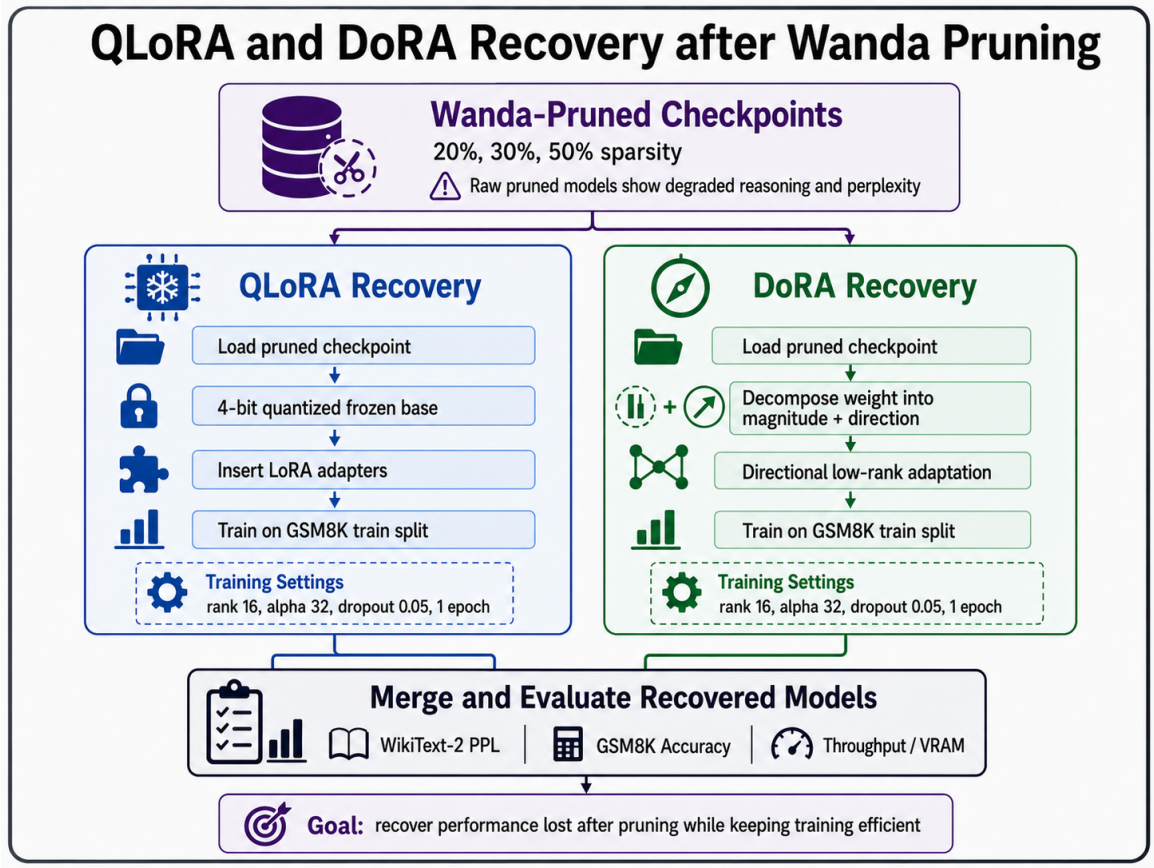

Figure 4.3: PEFT-based recovery pipeline for Wanda-pruned models using QLoRA and DoRA adapters.

Figure 4.3 shows the recovery process applied after Wanda pruning. PEFT adapters are added to the attention and projection layers in each pruned model. Memory-efficient recovery using low-rank adapters and quantization is done using QLoRA while DoRA performs magnitude–direction decomposition of the model parameters for recovery. After training on the GSM8K training subset, the trained adapters are added to the base pruned model before performing the evaluation step.

In the context of model recovery, QLoRA serves two purposes: first, as an adapter that helps adapt the model to a new task, and second, as a tool to correct degradation caused by Wanda pruning. The idea is that the adapter will be able to learn to correct the errors of the pruned model so as to perform better on the given dataset.

The setup for QLoRA model recovery includes parameters such as the adapter rank, alpha, dropout, learning rate, batch size, gradient accumulation, number of training steps, number of epochs, and maximum sequence length. All the mentioned values are fixed for each degree of sparsity of the model.

4.7.2 DoRA Recovery

DoRA recovery is included because it changes the standard low-rank adaptation mechanism. Instead of treating adaptation only as an additive low-rank update, DoRA decomposes the pre-trained weight into magnitude and direction components. The adapted weight can be written as

$$W' = m\frac{W_0 + BA}{\|W_0 + BA\|_c},$$

where m represents the magnitude component and BA gives the low-rank directional update. This is what makes DoRA unique compared to traditional LoRA-based recovery. It allows the model to control the scale and direction of the adapted weight representation.

This difference gains importance after pruning. Wanda changes the structure of the weights by removing some of them. Recovery that can control both the scale and direction of weight might behave differently from a simple additive adapter. But this does not necessarily mean DoRA will always be superior to QLoRA. The comparison takes place under equal pruning and testing conditions.

In this experiment, DoRA is run on the same set of pruned checkpoints where QLoRA was used for recovery. This will allow for a comparative study between the two recovery algorithms at levels of sparsity of 20%, 30%, and 50%.

4.7.3 Adapter Merging and Inference

After adapter recovery training, adapters could be loaded individually along with the pruned base model or incorporated in the model itself depending on the experiment setup and compatibility of the used framework. The individual loading of adapters has some advantages over the adapter merge in the context of experimentation. Firstly, it maintains the modularity of the base pruned checkpoint and the recovery adapter. Additionally, such approach allows comparing the effect of the QLoRA and DoRA adapters trained using the same pruned model.

While the adapter merge can simplify the inference process through incorporation of adapter weights in the base ones, the right choice depends on available memory space, desired precision, and necessity of switching between several adapters. In the current study, the focus is on consistent evaluations rather than optimization of deployments based on adapter merge alone.

An important practical note should be made, however. PEFT recovery does not make

the pruned base model unnecessary for the deployment. While being smaller in comparison with the base model, the adapter continues functioning on top of it. Thus, when analyzing the deployment, one should take into account the size of both pruned model and recovery adapter. It explains the consideration of model and adapter sizes, VRAM consumption, and inference throughput in the next section.

The last recovered-model evaluation stage included merging the trained adapter with the respective pruned base model prior to evaluating the performance on GSM8K and WikiText-2 data. Such action was primarily taken to optimize the inference process. If the adapter was retained as a separate component during inference, the token generation process would have been slower due to the necessity to perform the computation via the adapter at each forward pass. As a result, the inference process became simpler in this case with similar speeds of evaluation among the recovered models.

4.8 End-Device-Oriented Evaluation

Evaluation oriented toward end-device restores the connection to the actual target of our experiment. In this case, by end-device we do not mean that the produced model would actually run on some mobile phone or a real embedded machine. Instead, we use this terminology from the research perspective – to refer to inference environments that suffer from memory, computing power, and speed limitations in comparison to large cloud servers. It could be, for example, a smaller GPU, a CPU setup, or consumer machine.

The recovered pruned models are evaluated according to both task-related metrics and efficiency criteria. Among the key metrics are the following: model size after pruning, adapter size, VRAM used in inference, token throughput, latency per request, and task performance. This combination is important since no model can be considered deployable simply because it is sparse; it should also be able to work under certain hardware conditions and produce sensible outputs.

At this step, we should understand trade-offs as well as consequences. Higher sparsity allows lowering non-zero weight counts but might negatively affect GSM8K accuracy or raise perplexity. Recovering task performance increases the number of adapter parameters and might affect inference memory consumption depending on the way the adapter was trained. This means that the best model may not be the most compact but the one that achieves a reasonable trade-off.

The following metrics are collected during end-device-oriented evaluation:

Efficiency Profile = {Peak VRAM, Throughput, Latency, Model Size, Adapter Size}.

These values are interpreted together with

Performance Profile = {WikiText-2 PPL, GSM8K Accuracy}.

However, there exists one aspect that should not be overlooked. Sparsity is not proportional to the wall-clock speed-up by nature unless used along with sparse kernel computations or hardware-aware libraries for inference. Thus, I do not make any exaggerations in regards to the speed-ups achieved through the process of pruning. The analysis provided relies on real-time metrics and behavior observed during the experiment.

4.9 Complete Experimental Flow

It should be noted that there was no separation between the two phases, meaning both pruning and recovery were in one pipeline because they depended on each other sequentially. Initially, the dense model sets the standard for comparison. Subsequently, pruning based on Wanda is performed and sparse checkpoints of the selected sparsities are obtained. Then the evaluation of the raw pruned checkpoints takes place without the recovery step.

Let M denote the original dense model and let s denote a target sparsity level. Wanda pruning produces a sparse model

$$M_s = P_{\text{Wanda}}(M, s).$$

The pruned model M_s is then recovered using a PEFT method R_r, where r is either QLoRA or DoRA:

$$M_s^{\text{rec}} = R_r(M_s, D_{\text{train}}).$$

This is where we compare the three model states, namely the densely-connected base model M, the sparse pruned model M_s, and the recovered modelM_s^{rec}. There are two main hypotheses being tested here - the degree of performance loss resulting from pruning, and the recoverability of such losses.

The final stage base model in this work is Llama-3.2-3B-Instruct. Wanda activation-aware pruning was done at three different levels of sparsity: 20%, 30%, and 50%. The respective checkpoints are named as `wanda_s20`, `wanda_s30`, and `wanda_s50`. Both QLoRA and DoRA recovery were performed on each checkpoint separately, giving a total of six configurations.

It is crucial to specify what the data used in Wanda pruning means. The activation data that was used in Wanda is the calibration data, but not the validation data in the supervised learning paradigm. Namely, a relatively small batch of input sequences is passed through the model to gather activation statistics from linear layers of interest. Then this information, along with the absolute value of the weights, is used to calculate the Wanda importance score. In the pruning stage, the amount of activation collection data was equal to 64 calibration sequences with the length of 512 tokens. This data did not serve for the model training, its purpose is to provide some estimates on which weights should be pruned.

The GSM8K test was done with fixed number of samples of 650 problems and random seed of 42. Also, the maximum number of generated tokens was set to 512 per problem. The same prompt schema, the same answer extraction rules, and the same evaluation script were used in the evaluation of dense baseline model, raw pruned models, and recovered models. The decoding configuration was kept unchanged between the models. Top-p sampling parameter or other generation parameters were fine-tuned individually for each checkpoint.

Algorithm 1 Efficient Recovery Pipeline for Pruned LLMs

Input: Original pretrained model M, sparsity levels $S = \{20\%, 30\%, 50\%\}$, recovery methods $R = \{\text{QLoRA}, \text{DoRA}\}$, evaluation datasets D_{eval}, recovery training data D_{train}

Output: Evaluation results for dense, raw pruned, and recovered models

Evaluate dense baseline model M on D_{eval}
for each sparsity level $s \in S$ **do**
    Apply Wanda pruning:

$$M_s \leftarrow P_{\text{Wanda}}(M, s)$$

    Evaluate raw pruned model M_s on D_{eval}
    for each recovery method $r \in R$ **do**
        Perform PEFT recovery:

$$M_{s,r}^{\text{rec}} \leftarrow R_r(M_s, D_{\text{train}})$$

        Merge the trained adapter with the corresponding pruned base model
        Evaluate recovered model $M_{s,r}^{\text{rec}}$ on D_{eval}
    end for
end for
Compare all configurations using perplexity, GSM8K accuracy, throughput, and deployment-oriented indicators
Select the most suitable sparsity–recovery configurations under the chosen evaluation constraints

For WikiText-2 perplexity metric, the test data contains 131,072 tokens. Sequence length was set to 1,024 tokens with stride equal to 512 tokens. It ensures that all dense, pruned, and recovered models will be evaluated on the same text.

During the recovery process, we used the GSM8K training split for this task. A total of 7,473 training samples were used for adapter recovery, where each recovery task underwent one epoch of training. The adapter rank was kept constant at 16, while the LoRA alpha was tuned to 32, and the dropout and learning rate were set to 0.05 and 2×10^{-4}, respectively. The target modules used in both QLoRA and DoRA recovery are the same:`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`. The advantage of using static

adapter positions allowed for more control when comparing QLoRA to DoRA, since the degree of pruning, amount of recovery data, and adapter insertion points stayed consistent.

For the final tests, the trained adapters were incorporated into the pruned base models before performing any evaluation on the GSM8K and WikiText-2 datasets. The main goal here was to optimize the inference process by reducing the number of operations. Without merging the adapters, calculations will need to be performed each time during generation, which could cause slowdowns. Incorporating the adapters into the model makes the inference process easier and provides a more traditional way of model loading. Therefore, the results obtained from the experiments in the next chapter relate to the adapter-merged models.

Chapter 5

Algorithms, Implementation, and Complexity

5.1 System Pipeline Overview

The implementation itself followed the pipeline principle, with each step built upon the preceding one. Such a structure was preserved since each subsequent step relied on the results of the previous stage. The dense model had to be evaluated first, after which the pruned checkpoints were to be created, and the recovery adapters were to be fine-tuned on the pruned checkpoints. Only then could the comparison of all the model variants be done according to the same conditions. Otherwise, the final experiment would be hard to validate.

Generally speaking, the process includes five main blocks: baseline evaluation, Wanda pruning, recovery training, evaluation, and logging/artifact management. The baseline evaluation scripts are to evaluate the original Llama-3.2-3B-Instruct model prior to the compression process. The pruning script uses the Wanda approach on the required sparsity. The recovery block involves training both QLoRA and DoRA adapters on the pruned checkpoints. Evaluation scripts are used multiple times for baseline, pruned, and recovery models to allow for comparison. Metrics, predictions, performance, and memory logs are stored with the help of logging scripts.

It may be assumed that reproducibility relies not only on the algorithms involved but also on the manner in which outputs are stored. Thus, for GSM8K, output answers will also have to be saved alongside the metrics to better understand the nature of the mistake. For the perplexity calculation, the exact configuration for evaluation will have to be preserved to interpret any differences in PPL as model-related and not script-dependent. This approach makes the experiments much easier in the case of varying sparsities and multiple recoveries.

Thus, the implementation serves as an experimental control framework. Due to the fact that multiple states of the model are compared, the evaluation process has to be carefully repeated for each state. In particular, in the case of GSM8K, even slight discrepancies in answer

extraction and length influence the evaluation result significantly.

Table 5.1: Major components of the implementation

Component	**Purpose**
Baseline evaluation scripts	Evaluate the original dense model on WikiText-2 perplexity and GSM8K accuracy.
Pruning script	Apply Wanda pruning at selected sparsity levels and save the pruned checkpoints.
Recovery scripts	Train QLoRA and DoRA adapters on the Wanda-pruned models.
Evaluation scripts	Evaluate dense, raw pruned, and recovered models under the same protocol.
Logging utilities	Save metrics, JSONL predictions, throughput, latency, and hardware usage.
Upload utilities	Save model checkpoints and adapters locally or upload them to Hugging Face Hub when required.

5.2 Baseline Evaluation Algorithm

The baseline evaluation algorithm evaluates the dense model to establish its performance before pruning. In this way, it establishes the metrics that will be used for comparison in future experiments. As much as possible, the evaluation process follows the same principles throughout, so the algorithm serves to define the common method of evaluation for the entire experiment.

The baseline evaluation algorithm loads the tokenizer and base model, calculates the perplexity value on WikiText-2, provides answers to GSM8K questions, checks the results against the correct answer and saves the metrics of interest. In addition to the accuracy metric and perplexity, token generation and throughput, as well as the maximum VRAM consumption, are saved. These numbers will come in handy at the point where deployment feasibility will be considered.

It may seem that the step is easy and not important, but actually, it plays an essential role. This step defines the evaluation standard. Loose results of baseline evaluation could have made it hard to prove anything further. For that reason, the baseline results can be considered the foundation of this experimental study.

Algorithm 2 Baseline Evaluation

Input: Dense model M, WikiText-2 evaluation set D_{ppl}, GSM8K evaluation set D_{gsm}

Output: Baseline perplexity, GSM8K accuracy, generated outputs, and efficiency metrics

Load tokenizer and dense model M
Tokenize D_{ppl} using the model tokenizer
Compute next-token prediction loss on D_{ppl}
Calculate perplexity from the average loss
for each question $q_i \in D_{\text{gsm}}$ **do**
    Generate model response $\hat{y}_i$ using the fixed prompt template
    Extract the final numerical answer from $\hat{y}_i$
    Compare the extracted answer with the gold answer
end for
Compute GSM8K accuracy
Record generated tokens, throughput, latency, and peak VRAM
Save metrics and prediction outputs

5.3 Wanda Pruning Algorithm

The Wanda pruning technique is implemented after baseline analysis. In this pruning technique, the magnitude of the weight is used along with the activation norm of its input channel. Therefore, this technique can be considered more appropriate than basic pruning because the value of LLM weights cannot always be interpreted correctly through their magnitude.

To compute the Wanda score, the activation for the selected target layer will be extracted using calibration data, and the Wanda score will be calculated as

$$S = |W_l| \cdot \|X_l\|_2,$$

where W_l denotes the weight matrix of layer l and X_l denotes the collected input activation statistics for that layer. The lowest-scoring weights are selected according to the required sparsity level and set to zero. The resulting sparse model is saved as a pruned checkpoint.

This process is carried out for all the selected sparsity rates. During pruning, there is no recovery process as well. The pruning stage will only yield the pruned model without recovery. Before adapting the pruned model, it will be assessed individually to separate the impact of pruning from adaptering process.

Algorithm 3 Wanda Pruning

Input: Dense model M, calibration data D_c, target sparsity level s

Output: Wanda-pruned model M_s

1: Load dense model M
2: **for** each target linear layer l in M **do**
3: Pass calibration samples from D_c through the model
4: Collect input activation statistics X_l for layer l
5: Obtain the layer weight matrix W_l
6: Compute Wanda importance score:

$$S_l = |W_l| \cdot \|X_l\|_2$$

7: Identify the lowest-scoring weights according to sparsity level s
8: Set the selected weights to zero
9: **end for**
10: Save the pruned checkpoint as M_s
11: Return M_s

5.4 Recovery Training Algorithm

Post pruning, there is always some degradation of language modeling/reasoning capabilities of the model.

Recovery aims to mitigate this degradation through parameter-efficient fine-tuning. As opposed to fine-tuning, all model parameters are frozen, and the training is done for adapter parameters only, which preserves the integrity of the recovery process within the limited-resource assumption. On a conceptual level, the algorithm used during the recovery procedure is universal to both QLoRA and DoRA.

In the chosen PEFT approach, trainable adapter parameters are introduced, and the model is fine-tuned with train batches selected from the recovery dataset, computing the language modeling loss, and updating adapter param- eters, while keeping all other weights constant.

The adapter architecture for QLoRA is the standard low-rank update approach in conjunction with efficient memory usage and model loading, while for DoRA, weight decomposition into magnitude and direction is used. While their inner workings are different, the high-level pipeline stays the same for both cases.

Algorithm 4 PEFT-Based Recovery

Input: Pruned model M_s, recovery dataset D_r, PEFT method $A \in \{\text{QLoRA}, \text{DoRA}\}$

Output: Adapter checkpoint and recovered model $M_{s,A}^{\text{rec}}$

Load pruned model M_s
Freeze base model parameters
Insert trainable adapter parameters according to method A
for each training step **do**
    Sample a mini-batch from D_r
    Run the forward pass through the pruned model and adapter
    Compute language modeling loss
    Backpropagate only through adapter parameters
    Update adapter parameters using the selected optimizer
end for
Save the trained adapter checkpoint
Merge adapter weights with the corresponding pruned base model for final evaluation
Return recovered model $M_{s,A}^{\text{rec}}$

Recovery is an important part of our experiments as pruning by itself only answers how much the model can be compressed without significant degradation. However, what we care about is whether the lightweight finetuning procedure can restore enough capability of a pruned model to make its use practical.

5.5 GSM8K Evaluation Procedure

The GSM8K evaluation methodology is performed on the baseline, pruned, and restored models to evaluate their mathematical reasoning ability. I don't consider GSM8K as a standard text-generation evaluation benchmark. In the current research, it is employed to assess the ability of the model to understand the word problem, go through the required arithmetic reasoning, and return the accurate numeric result.

The evaluation begins with the preloading of the fixed subset of examples in GSM8K with a fixed random seed. A fixed seed is critical here since multiple model variants are evaluated. The variation of the question set leads to variance in results due to the sample selection rather than model performance. Also, the prompt templates are consistent across all models to provide reliable comparison results.

For each question, the model produces a solution. The resulting text could contain the arithmetic reasoning process, calculations, and the final result of calculation. As the correct answer should be numeric, the script extracts the last numeric value from the produced solution

and compares it to the ground truth. Accuracy is calculated as

$$\text{Accuracy} = \frac{\text{Number of Correct Answers}}{\text{Total Number of Evaluated Questions}}.$$

As for the settings in the experiments, the number of generated tokens in each case is fixed to the same maximum number. This is important since too low generation length might cut off the reasoning process before reaching the conclusion, while too large one may result in useless reasoning continuation.

Additionally, the decoding setting remains constant so that the variations introduced by sampling procedure will not be a part of the comparison.

It should be noted that the accuracy of GSM8K is highly dependent on the answer extraction. A model can output a solution, yet use a different form of writing it down. At the same time, another one may get the correct number, but provide a weak reasoning trail. Thus, the predictions in JSONL form will be kept for qualitative analysis, as it will help to establish whether pruning helps to improve the reasoning process or just the form of the final solution.

Thus, the purpose of using GSM8K in this study is double – first, it helps to measure reasoning skills of each model numerically, and second, it helps to get examples of generation that could shed light on the effect of pruning and recovery on reasoning process.

5.6 Perplexity Evaluation Procedure

Perplexity analysis is applied to analyze the quality of language modeling. Although the GSM8K data set is designed to test math reasoning, the perplexity test is performed to evaluate the general ability of the model to predict natural language. In the current research, WikiText-2 will be utilized to perform perplexity testing.

Firstly, the evaluation text is tokenized using the tokenizer of the base model. Then, the tokenized sequence is split into blocks within the limits of context size and memory capacity. For each block, the next-token prediction loss is calculated. The losses are summarized for the evaluated tokens, and the perplexity is calculated as

$$\text{PPL} = \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\log p(x_i)\right),$$

where N is the number of evaluated tokens and $p(x_i)$ is the probability assigned by the model to the correct token x_i. Lower perplexity indicates better language modeling quality.

In the experiment, the identical text splitting, tokenizer, maximum token or char limit, and method of averaging loss are used for all model variants, as perplexity depends on the evaluation text, splitting strategy, and tokenizer used. It was important in the experiments as

we compare the performance of dense, pruned, and recovered models, and thus, the evaluation strate- gy has to be kept constant throughout.

Perplexity is beneficial because it is a more stable metric compared to exact match accuracy. The fact that a model makes a mistake while generating a GSM8K response does not automatically mean that language modeling performance is lost, although perplexity would help us determine this fact. On the other hand, perplexity is not an indicator of reasoning perform- ance, meaning that even a good one might produce incorrect results.

5.7 Hardware and Software Environment

The tests are conducted within certain constraints of the GPU environment. This is not a matter of implementation but the essence of the work since the research is based on the idea of efficient recovery and deployability.

These constraints have an impact on all aspects of the experiment, including model loading, pruning, recovery training, batch sizes, selections of precision, and the duration of evaluations. The initial exploratory experiments were run on a scaled-down Tesla P6 setup; however, the final recovery training and evaluation experiments reported in this thesis were done on a RunPod NVIDIA A40 GPU.

In case of necessary recovery or other related processes, tests can be conducted in notebook or Kagglelike environments. The implementation utilizes mainly the following tools: PyTorch, Hugging Face Transformers, PEFT, Datasets, and Accelerate. Depending on the task, the model is loaded in one of three formats: FP16, BF16, 4-bit.

The final recovery training and evaluation was performed using a paid RunPod GPU setting. Although initial tests were carried out using small GPU settings, the final results were obtained using an NVIDIA A40 GPU. The latter had enough memory to recover and test all six models, which included recovery using three sparsity values with QLoRA and three sparsity values with DoRA.

The hardware setup influences a number of design decisions. The ability to do full fine-tuning of the parameters of the models used is out of the question in the proposed setup, hence the choice of PEFT-based model recovery. Other recovery setup parameters such as batch size, gradient accumulation, sequence length, and quantized loading are also set to accommodate memory constraints so that experiments will still align with the motivation for the study.

The hardware setup is not merely a background factor in the study. It influences the research question. With unlimited computing power, full fine-tuning and model comparisons are possible tasks. In the face of constrained GPU memory, the pertinent question is whether pruning and model recovery could deliver usable results.

Table 5.2: Final hardware and software environment used for recovery and evaluation

Item	**Final Configuration**
Evaluation platform	RunPod paid GPU environment
GPU	NVIDIA A40
GPU memory	Approximately 46 GB VRAM
Base model	Llama-3.2-3B-Instruct
Pruning method	Wanda activation-aware pruning
Pruning sparsity levels	20%, 30%, and 50%
Recovery methods	QLoRA and DoRA
Framework	PyTorch
Libraries	Transformers, PEFT, Datasets, Accelerate, BitsAndBytes
Artifact backup	Hugging Face Hub

5.8 Complexity Analysis

This paper's discussion on complexity is confined to the three primary stages of computation of the pipeline: Wanda pruning, recovery training through PEFT method, and inference. This analysis is not an attempt to come up with the entire cost of each step of computation for the Transformer network; rather, it seeks to establish the major costs incurred during the process.

5.8.1 Pruning Complexity

Wanda pruning incurs its biggest overhead during activation collection and computation of the scores. Calibration data is passed through the network to collect input activations to the target linear layers. No backward propagation takes place in this process, and only forward propagation takes place.

Consider a target linear layer whose weight matrix is W_l. Let the collected activation statistics be represented by X_l. Wanda then computes the score

$$S = |W_l| \cdot \|X_l\|_2.$$

33 This score is computed by multiplying the absolute value of weights and their respective norms. The weights that have lower scores are retained based on a particular sparsity value and become zero. Wanda is less complicated than other second-order pruning algorithms because of the absence of the need to invert the Hessian matrix, reconstruct weights, and retrain models. Pruning is performed individually for 20%, 30%, and 50% sparsity values; therefore, the pruning costs increase with the number of sparsity settings.

5.8.2 Recovery Training Complexity

Training for the recovery task employs parameter-efficient fine-tuning. As such, only adapter parameters are fine-tuned whereas the pruned base model is kept fixed. The number of trainable parameters and memory needed for storing gradients and optimizers are therefore reduced when compared to full fine-tuning.

In the LoRA-based approach, the weight updating is represented by

$$\Delta W = BA,$$

where A and B are low-rank trainable matrices. The trainable parameter count for such an adapted matrix, given the original weight matrix shape to be $d \times k$ and adapter rank to be r, can be expressed approximately as

$$r(d + k),$$

which is much smaller compared to dk for $r \ll \min(d, k)$. The memory savings due to this factor form the primary rationale behind why PEFT is applicable in situations where pruning of the model is to be done. While QLoRA saves on memory by quantizing the base-model, DoRA alters the procedure by making use of magnitude–direction decomposition. However, even in such situations, the computationally expensive process remains that of the forward and backward passes.

5.8.3 Inference Complexity

Inference within autoregressive LLMs is still costly since the generation of tokens is sequential. Every subsequent token requires a forward pass over the entire network, with the complexity depending on the size of the model, the sequence length, the precision of computations, the batch size, and the efficiency of the inference engine.

It is not necessarily true that the use of Wanda pruning yields proportional improvements in inference speed, as the operation remains dependent on the type of matrix multiplication kernels utilized. Dense matrix multiplication kernels, for instance, would require non-zero weights to be stored and multiplied within dense tensors. As a result, a model having 50% sparsity is unlikely to be twice as fast without sparse kernels or hardware support.

This means that the interpretation of inference performance in this dissertation relies solely on experimentally obtained measurements of throughput and latency as opposed to the theory alone. Adapter merging helps make the recovered model simpler, but the pruning algorithm would have to load and execute the pruned base model regardless.

5.9 Reproducibility Details

Reproducibility becomes especially important in this study because of the presence of multiple models and evaluations. Minor changes in the random seed, prompts, lengths of generation, and answer extraction may impact results, particularly for GSM8K. In order to achieve this, the design of the experiment takes into account all the important parameters associated with each of the outputs produced by the study.

Important parameters associated with reproducibility include random seed value, path to the checkpoint, version of the data set, number of samples used in the evaluation, maximal length of generation, hyperparameters used in adapters, and file paths where outputs are saved. Generated responses from GSM8K are saved in JSONL format along with answers extracted and correctness labels. Summary metrics are also separately saved in JSON format to reproduce tables even if not all experiments are reproduced.

The following information is recorded for each major run:

- model name or checkpoint path;
- pruning sparsity level, where applicable;
- recovery method, such as QLoRA or DoRA;
- dataset name and split;
- evaluation sample size and random seed;
- maximum generation length for GSM8K;
- adapter hyperparameters such as rank, alpha, dropout, learning rate, batch size, and training steps;
- output paths for JSONL prediction files and metric files;
- hardware-related values such as peak VRAM and throughput.

There were many learnings involved in creating this system. Just documenting accuracy alone isn't sufficient because there's much more information conveyed through the outputs of the model than through that one number. Outputs allow for detecting incorrect computations, problems with the generation process, following the prompt, and errors with extraction. The reproducibility pipeline is also key for writing up the dissertation.

The reproducibility pipeline ensures support for writing up the final dissertation, which involves making sure the model analysis takes place under specific conditions where the outputs are saved.

15

Chapter 6

Experimental Results and Discussion

6.1 Experimental Design

These experiments investigate the amount of performance loss suffered due to Wanda pruning and what portion of the losses can be reversed via PEFT. The Llama-3.2-3B-Instruct model will undergo the analysis in the following three forms: dense baseline, raw Wanda-pruned checkpoint, and recovered checkpoint.

In the case of Wanda pruning, the sparsities of 20%, 30%, and 50% will be considered. The former corresponds to a weak form of pruning, the latter two represent an intermediate configuration and an aggressive one, respectively. Each raw pruned checkpoint is then recovered via QLoRA and DoRA.

The groups of models subject to evaluation include dense baselines, raw Wanda-pruned checkpoints, QLoRA-recovered checkpoints, and DoRA-recovered checkpoints. All models follow the same protocol. The language modeling quality metric will be WikiText-2 perplexity, GSM8K accuracy will measure mathematical reasoning capability, and throughput will serve as the efficiency metric. Raw pruned models are evaluated prior to recovery in order to quantify the damage done by pruning and the improvement achieved during recovery.

6.2 Baseline Results

The dense version of Llama-3.2-3B-Instruct model is initially benchmarked to obtain the ground truth comparison metric for other experiments. Baseline scores are reported in Table 6.1. Specifically, for the GSM8K evaluation, 650 questions are employed, using random seed 42 with max length of 512. For WikiText-2, the score metric is perplexity measured on 131,072 tokens.

The dense baseline yields the best score overall in this scenario. It has the lowest perplexity for WikiText-2 and the highest accuracy for GSM8K amongst all the experimental setups considered here. Hence, the recovered models should always be compared against both

Model	PPL	Correct	Accuracy	Throughput
Llama-3.2-3B-Instruct	10.4627	462 / 650	0.7108	46.33 tok/s

Table 6.1: Baseline performance of the dense Llama-3.2-3B-Instruct model

the dense baseline and the raw pruned checkpoint models.

6.3 Effect of Wanda Pruning

Once the dense checkpoint has been prepared, Wanda pruning is conducted on the checkpoints with 20%, 30%, and 50% sparsity, respectively. Before evaluating QLoRA and DoRA recovery, we assess the impact of the pruning process itself on performance, as demonstrated in Table 6.2.

Model	Sparsity	PPL	Correct	Accuracy	Throughput
wanda_s20	20%	23.3884	328 / 650	0.5046	48.06 tok/s
wanda_s30	30%	25.4533	314 / 650	0.4831	47.56 tok/s
wanda_s50	50%	45.1633	117 / 650	0.1800	48.02 tok/s

Table 6.2: Performance of raw Wanda-pruned models before recovery

The experiments reveal an evident decline in performance following pruning. The raw pruned checkpoints exhibit lower accuracy and higher perplexity than the baseline. At 20% and 30%, the reduction in performance is relatively small; however, it is severe at 50% sparsity, with GSM8K accuracy dropping to 0.1800.

As expected, pruning has affected both next-token prediction and math reasoning ability. With 20% and 30% sparsity, the models can still perform well, whereas 50% sparsity removes a significant amount of mathematical reasoning performance from the model. These findings support our assumption that there is a need for a recovery step with QLoRA and DoRA.

6.4 Recovery Results Using QLoRA

After pruning of the checkpoints, the recovery with QLoRA was performed for each of the sparsity levels. For QLoRA recovery, the pruned checkpoints were considered frozen models, and the adaptation parameters were trained on the GSM8K training dataset. The same recovery settings were utilized for each of the sparsity levels, including 20%, 30%, and 50%.

The QLoRA recovery results are shown in Table 6.3 below.

QLoRA recovery provides better performance for all three raw pruned checkpoints. This performance gain can be observed for both metrics, GSM8K accuracy and WikiText-2 perplexity. QLoRA recovery shows the maximum effect for 50% sparse models. Nonetheless,

Model	Sparsity	PPL	Correct	Accuracy	Throughput
wanda_s20 + QLoRA	20%	21.7185	344 / 650	0.5292	47.49 tok/s
wanda_s30 + QLoRA	30%	23.2286	329 / 650	0.5062	46.80 tok/s
wanda_s50 + QLoRA	50%	33.8288	289 / 650	0.4446	47.77 tok/s

Table 6.3: Recovery results using QLoRA

despite applying the recovery method, the obtained results for QLoRA models are still worse than those for a dense baseline model.

6.5 Recovery Results Using DoRA

The application of DoRA recovery technique was done on the same checkpoints for Wanda pruning used to recover the QLoRA model. All factors, including the base model, sparsity ratio, training dataset, adapter size, and evaluation process, remain identical for both methods. However, the primary difference in DoRA is based on using magnitude-direction decomposition instead of standard LoRA update.

The DoRA recovery results are shown in Table 6.4 below.

Model	Sparsity	PPL	Correct	Accuracy	Throughput
wanda_s20 + DoRA	20%	21.5220	352 / 650	0.5415	47.37 tok/s
wanda_s30 + DoRA	30%	23.8493	343 / 650	0.5277	45.30 tok/s
wanda_s50 + DoRA	50%	34.4044	293 / 650	0.4508	47.64 tok/s

Table 6.4: Recovery results using DoRA

In comparison, DoRA performs better in all cases. The models obtained through DoRA recovery produce more accurate results on GSM8K tasks compared to their raw counterparts. The model with a 20% sparsity ratio produces the highest accuracy in its class, while the 50% model makes the most progress compared to the raw pruned model. Moreover, perplexity becomes lower as well, which means that the model improves its language modeling abilities.

6.6 Comparison of QLoRA and DoRA

Models trained from raw, Wanda pruned weights, and those obtained via QLoRA and DoRA recovery methods are compared in table Table 6.5. Given that the conditions of training, evaluation and pruning are identical for the QLoRA and DoRA recovery methods, the differences in the results can be attributed to recovery methods' peculiarities only.

Sparsity	Raw Acc.	QLoRA Acc.	DoRA Acc.	Raw PPL	QLoRA PPL	DoRA PPL
20%	0.5046	0.5292	0.5415	23.3884	21.7185	21.5220
30%	0.4831	0.5062	0.5277	25.4533	23.2286	23.8493
50%	0.1800	0.4446	0.4508	45.1633	33.8288	34.4044

Table 6.5: Comparison of raw pruned, QLoRA-recovered, and DoRA-recovered models

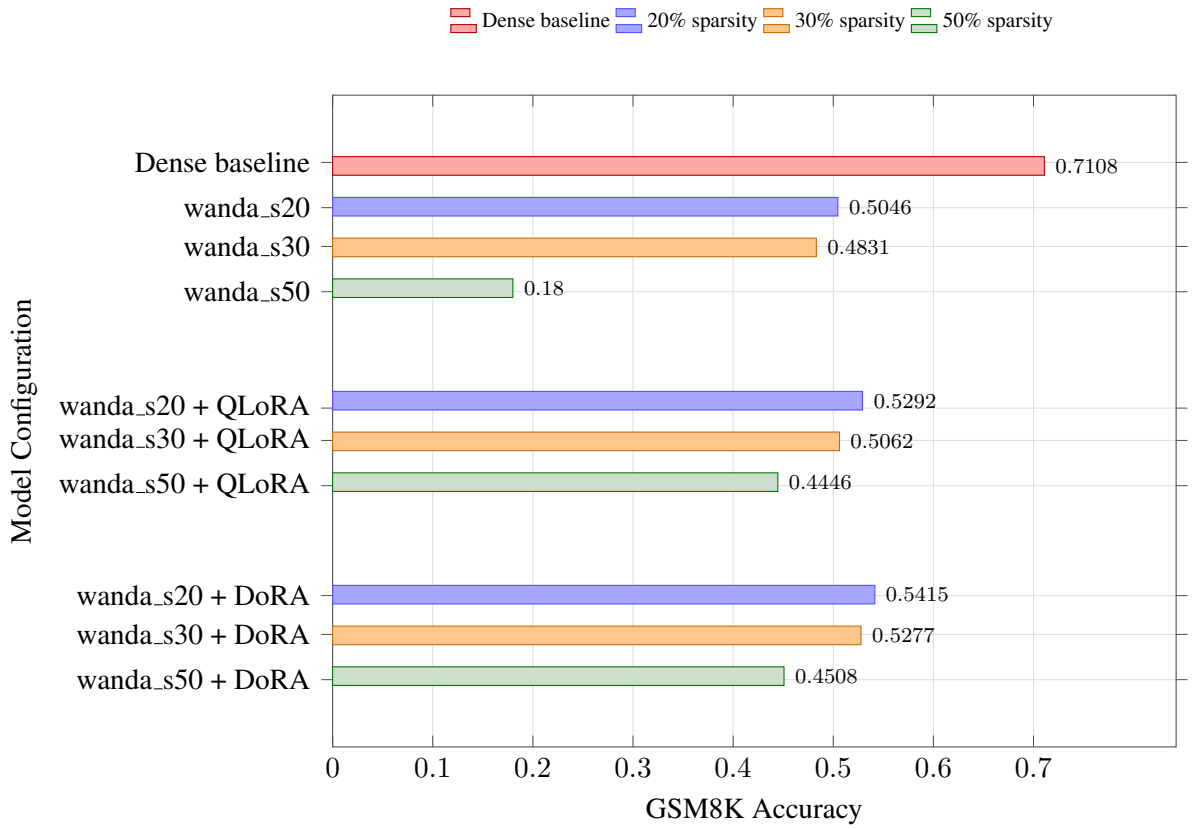

Figure 6.1: GSM8K accuracy comparison across dense baseline, raw Wanda-pruned models, and PEFT-recovered models.

As shown in figure 6.1, both recovery methods allow improvements in GSM8K accuracy post-Wanda pruning. Moreover, DoRA obtains higher GSM8K accuracy than QLoRA for the 20%, 30% and 50% sparsity rates.

However, as one may see from figure 6.2, there is no similarity in the behavior of models with respect to perplexity and GSM8K accuracy. While DoRA obtains the lowest perplexity value at 20% sparsity rate, QLoRA achieves a smaller perplexity at 30% and 50% sparsity rates. Hence, language model recovery and reasoning recovery have different behaviors. Thus, DoRA outperforms QLoRA in GSM8K recovery, while the latter is better for perplexity recovery at a larger sparsity ratio.

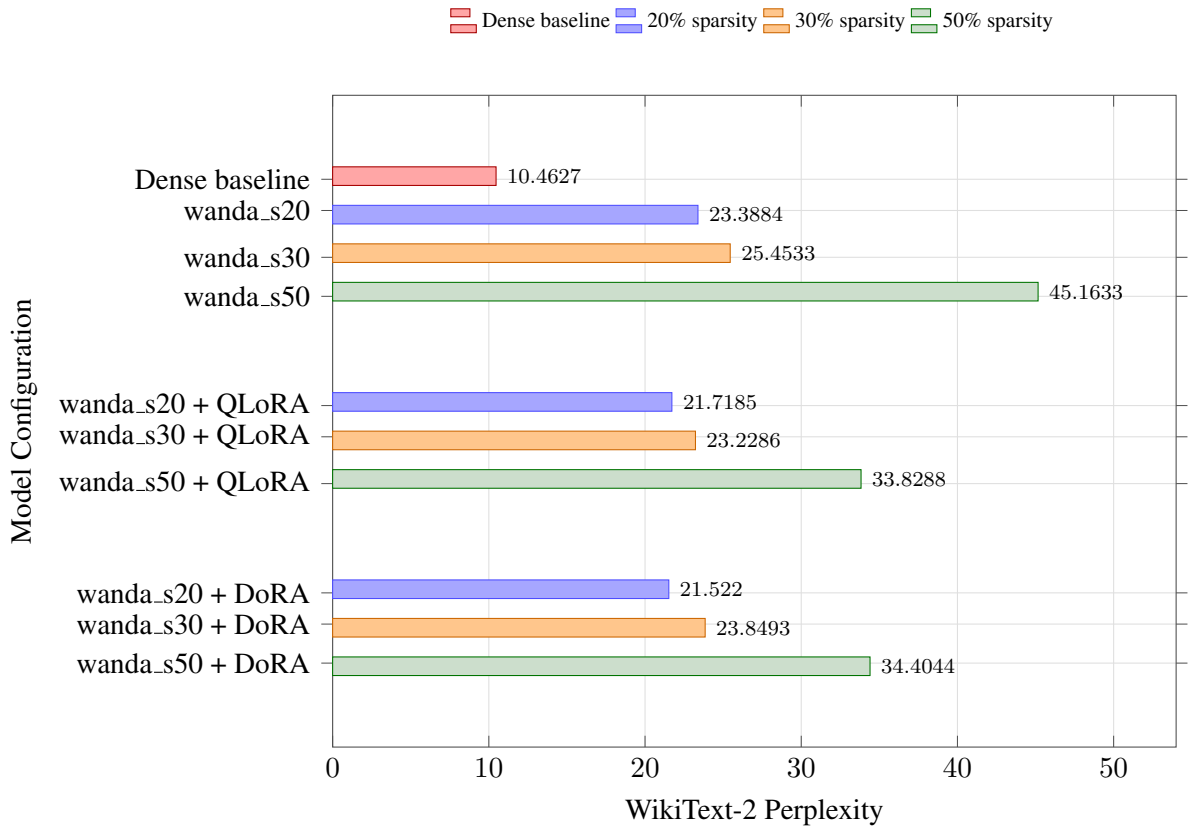

Figure 6.2: WikiText-2 perplexity comparison across dense baseline, raw Wanda-pruned models, and PEFT-recovered models. Lower perplexity indicates better language modeling quality.

6.7 Recovery Gain Analysis

While accuracy alone cannot provide a full explanation for recovery behavior, since every sparsity degree begins with different raw accuracy, GSM8K recovery gain is utilized to quantify the amount of recovered accuracy achieved with the application of recovery methods.

The GSM8K recovery gain is defined as

$$\text{Recovery Gain} = \text{Recovered Accuracy} - \text{Raw Pruned Accuracy}.$$

Since DoRA has the best performance on GSM8K under all sparsity rates, Table 6.6 lists the best recovered model and their gains for all sparsity rates.

Sparsity	Raw Acc.	Best Method	Best Recovered Acc.	Absolute Gain
20%	0.5046	DoRA	0.5415	+0.0369
30%	0.4831	DoRA	0.5277	+0.0446
50%	0.1800	DoRA	0.4508	+0.2708

Table 6.6: GSM8K recovery gain over raw Wanda-pruned models

The largest gain is achieved by recovery when the sparsity rate is 50%. In this case, DoRA can improve the GSM8K's accuracy from 0.1800 to 0.4508. It is clear that the improvement becomes obvious only when significant damage is done. Nevertheless, it cannot be said that the 50% sparsity recovered model is the best model among the three.

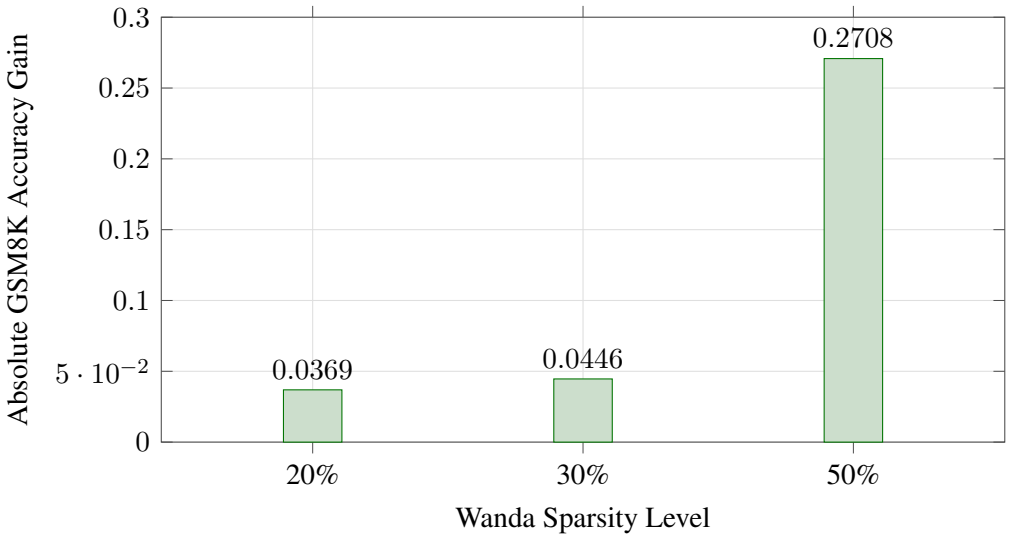

Figure 6.3: Best GSM8K recovery gain over raw Wanda-pruned models at each sparsity level. The gain is computed as recovered accuracy minus raw pruned accuracy.

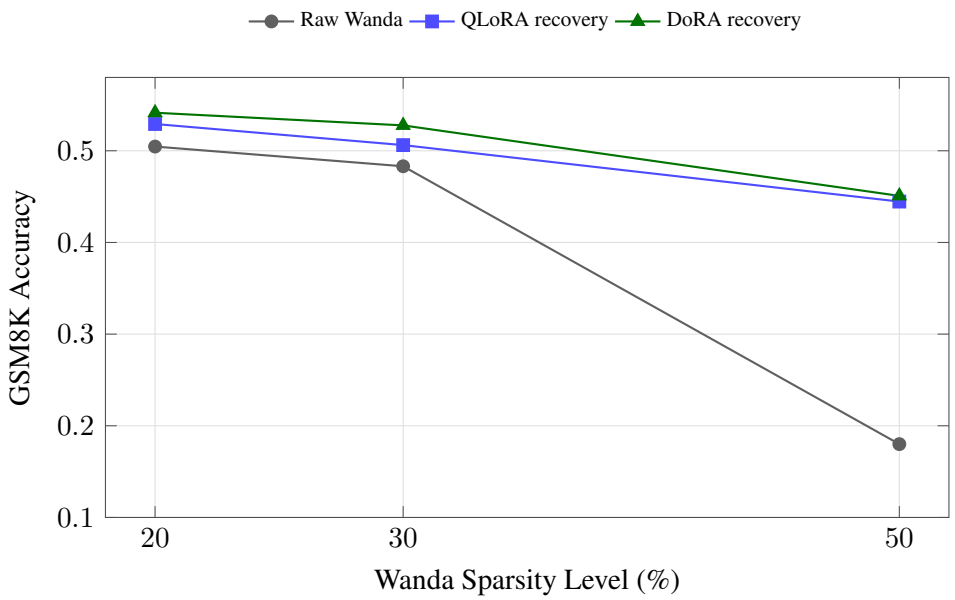

Figure 6.4: GSM8K accuracy trend across sparsity levels for raw Wanda-pruned, QLoRA-recovered, and DoRA-recovered models.

Figure 6.3 and Figure 6.4 present two results which are somewhat connected but still different. While the 20% DoRA is the one which achieves the highest recovered accuracy, it is not the 20% DoRA that provides the maximum gain

6.8 Efficiency Analysis

Efficiency is analyzed in terms of throughput in GSM8K generation. As Wanda introduces unstructured sparsity, we cannot claim any speed-up unless the sparse compute capability is utilized in the inference back end. Table 6.7 presents the results obtained for the dense model, pruned model, and recovered model.

Model	Sparsity	GSM8K Accuracy	Throughput
Dense baseline	0%	0.7108	46.33 tok/s
`wanda_s20`	20%	0.5046	48.06 tok/s
`wanda_s20 + QLoRA`	20%	0.5292	47.49 tok/s
`wanda_s20 + DoRA`	20%	0.5415	47.37 tok/s
`wanda_s30`	30%	0.4831	47.56 tok/s
`wanda_s30 + QLoRA`	30%	0.5062	46.80 tok/s
`wanda_s30 + DoRA`	30%	0.5277	45.30 tok/s
`wanda_s50`	50%	0.1800	48.02 tok/s
`wanda_s50 + QLoRA`	50%	0.4446	47.77 tok/s
`wanda_s50 + DoRA`	50%	0.4508	47.64 tok/s

Table 6.7: GSM8K accuracy and throughput comparison across dense, pruned, and recovered models

Throughput values are similar for all considered settings. The dense setting achieves throughput of 46.33 tok/s, and both raw and recovered models have throughput values in the similar range. It means that there was no significant wall-clock speed-up due to Wanda sparsification.

Such behavior is expected since sparsity on its own does not ensure that there will be any speed-ups during inference. Namely, if backend performs only dense multiplication, some zero-weight values can be still stored within the dense tensor during calculations. Hence, reaching 50% sparsity rate does not mean that we achieve $2\times$ speed-up.

Figure 6.5 presents additional evidence of similar behavior. Recovered models show throughput values close to those observed in the dense setting. Thus, the effect of model recovery for us was more related to restoring performance rather than achieving any speed-up.

As for our recovered models, `wanda_s20 + DoRA` achieved the highest accuracy on GSM8K while having throughput close to the dense setting. The `wanda_s50 + DoRA` recovered model demonstrated the biggest recovery gains compared to other models, although its overall accuracy remains lower than that of 20% and 30% recovery rates.

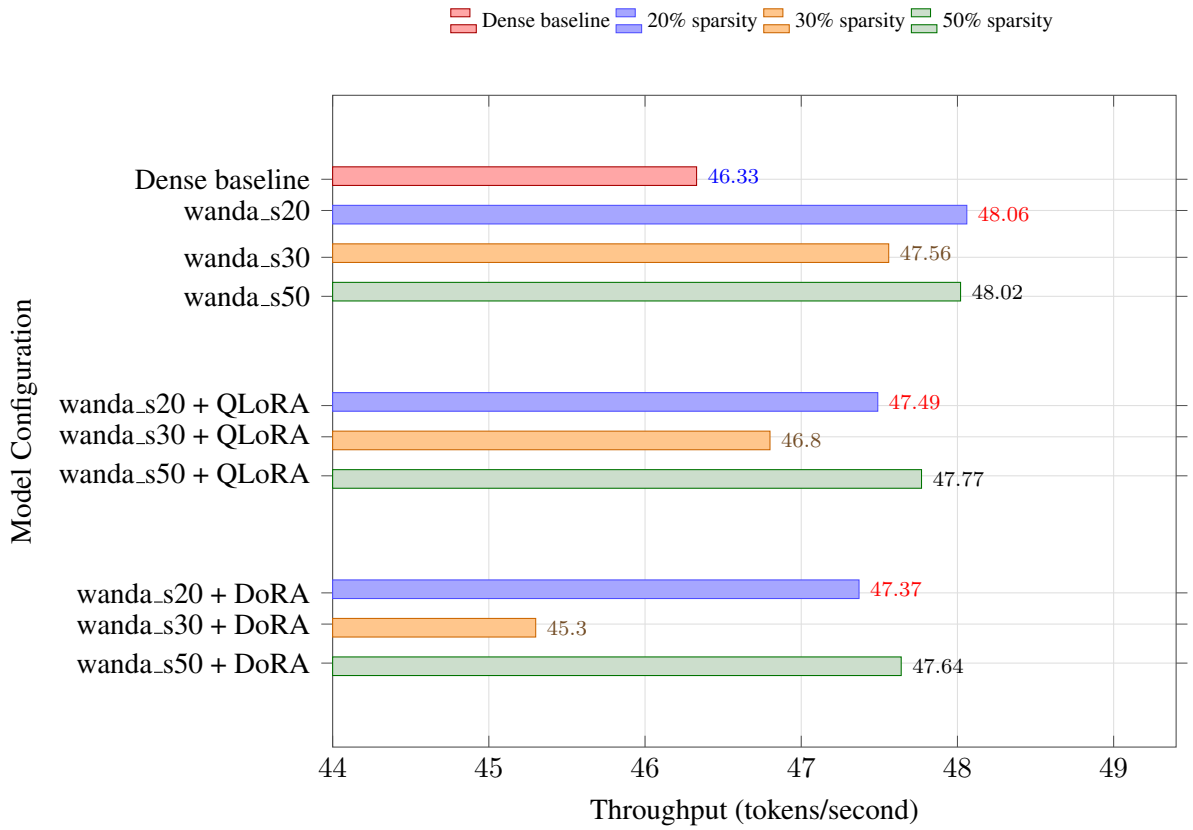

Figure 6.5: Throughput comparison across dense baseline, raw Wanda-pruned models, and PEFT-recovered models during GSM8K evaluation.

6.9 End-Device Feasibility Study

In terms of the end-device feasibility, the experimental results are discussed in light of the deployment motivation behind the dissertation. End-device does not imply any concrete deployment onto a mobile device or embedded system at this stage. Instead, the issue concerns a more abstract idea of running LLMs in conditions of limited memory, storage, and generation speed.

Because Wanda Pruning creates structural sparsity, the results obtained should not be taken as proof of acceleration on hardware devices. For a sparse checkpoint to be helpful for faster performance, it must be used by either the software framework or the hardware that has been able to exploit sparsity.

Taking into account the above information, the safe recovery configuration in terms of the reasoning accuracy is `wanda_s20 + DoRA`. Meanwhile, the `wanda_s30 + DoRA` configuration provides greater sparsity levels with still reasonable GSM8K accuracy values. Finally, the 50%-level models can serve well for research on extreme-level pruning recovery but not for the ultimate configuration for reasoning accuracy.

Thus, feasibility cannot be defined by sparsity alone. The useful compressed model

Configuration	Suitability Interpretation
`wanda_s20 + DoRA`	Best recovered GSM8K accuracy among the recovered models. This is the safest recovered configuration when reasoning accuracy is the main priority.
`wanda_s30 + DoRA`	Stronger sparsity than the 20% model while still retaining useful recovered accuracy. This is a reasonable middle-ground configuration.
`wanda_s50 + DoRA`	Largest recovery gain over the raw pruned checkpoint, but final accuracy remains clearly below the 20% and 30% recovered models. Useful for studying aggressive pruning recovery.
`wanda_s30/s50 + QLoRA`	Competitive perplexity compared with DoRA at higher sparsity. Useful when language modeling quality is prioritized over GSM8K accuracy.

Table 6.8: End-device-oriented interpretation of recovered model configurations

is characterized by multiple aspects including pruning level, reasoning accuracy, perplexity, throughput, and available sparse inference capabilities.

6.10 Ablation Studies

The analysis of the impact of ablation carried out within the context of this dissertation is concerned with two factors which can be controlled: the degree of sparseness of Wanda and the choice of the restoration process. All other aspects of the experiments, such as adapter rank, LoRA alpha, dropout, learning rate, the number of epochs, and data for recovery and testing, were kept unchanged.

The first one is sparsity. With higher sparsity ranging from 20% to 50%, the raw pruned model shows worse performance on GSM8K. The second one is the recovery technique. Table 6.9 illustrates the optimal recovery technique at different sparsity levels in terms of GSM8K accuracy and WikiText-2 perplexity.

Sparsity	Best GSM8K Method	Best GSM8K Acc.	Best PPL Method	Best PPL
20%	DoRA	0.5415	DoRA	21.5220
30%	DoRA	0.5277	QLoRA	23.2286
50%	DoRA	0.4508	QLoRA	33.8288

Table 6.9: Ablation over sparsity level and recovery method

As illustrated in the ablation results, DoRA performs better at GSM8K accuracy under any sparsity level. Nonetheless, perplexity does not follow the same trend. For example, DoRA shows the optimal perplexity under 20% sparsity, whereas QLoRA slightly outperforms it when the sparsity level is 30% and 50%. This verifies the importance of evaluating recovery by multiple metrics. Further ablation experiments in other settings are postponed to future work.

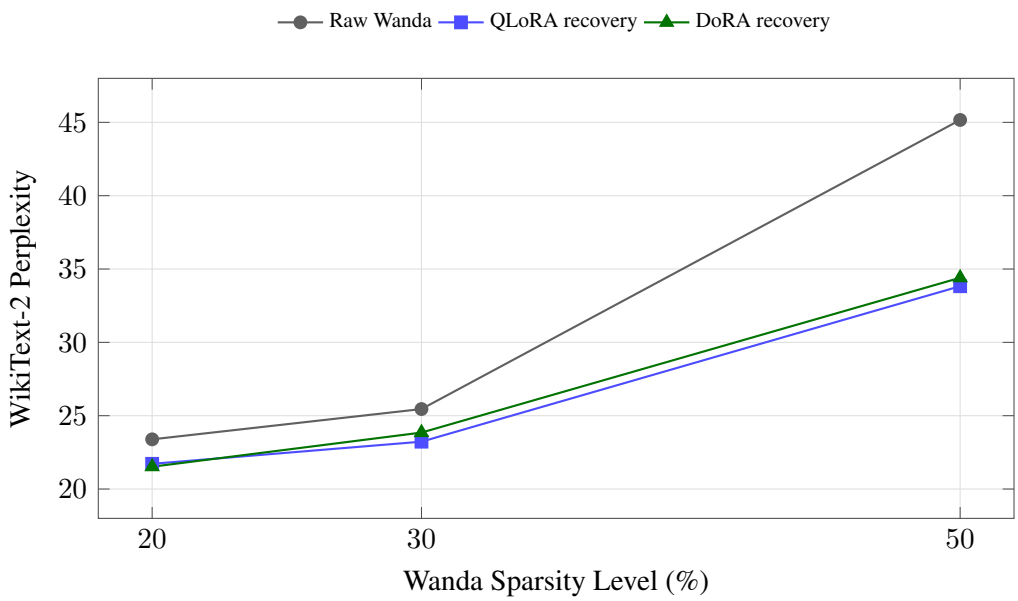

Figure 6.6: WikiText-2 perplexity trend across sparsity levels for raw Wanda-pruned, QLoRA-recovered, and DoRA-recovered models. Lower perplexity indicates better language modeling quality.

6.11 Qualitative Analysis of GSM8K

Case Type	Behaviour Observed	Interpretation
Correct after recovery	The raw pruned model gives an incorrect final answer, while the recovered model gives the correct final number.	Adapter recovery helps restore task-specific reasoning or answer formatting.
Still incorrect after recovery	Both raw pruned and recovered models give incorrect final answers.	The pruning damage may be too strong, or the adapter training is not enough for that reasoning pattern.
Partial reasoning error	The model follows part of the arithmetic correctly but makes an error in one intermediate step.	GSM8K accuracy is sensitive to small reasoning mistakes, especially after pruning.
Format or extraction issue	The model produces a response where the final answer is unclear or not written in a clean numerical form.	Evaluation depends not only on reasoning but also on stable final-answer formatting.
Preserved behaviour	Both the raw pruned and recovered models answer correctly.	Some questions remain unaffected by pruning, especially when the required reasoning is simpler.

Table 6.10: Common qualitative behaviours observed in GSM8K predictions

The accuracy of GSM8K is a quantitative comparison that can be made; nevertheless, this metric does not explain how the models succeeded or failed. To solve this problem, the pre-

diction files produced by the model were also analyzed. The model output, extracted numeric value, gold value, and correctness label are included in the file.

The common behaviours observed during this inspection are summarized in Table 6.10 above.

The qualitative analysis reveals that pruning does not make the model instantly lose its fluency. While pruned models can produce grammatically correct answers, their reasoning process tends to be unstable, with reasoning errors such as arithmetic mistakes, confusion about quantities, and ambiguity in final answer formats occurring commonly.

Recovery of the pruned models makes them more stable, particularly at sparsity rates of 20% and 30%. While recovery is still beneficial at 50%, reasoning errors tend to occur at a higher rate than at lower levels of sparsity.

6.12 Discussion

Based on the results from these experiments, Wanda pruning is harmful to both language modeling and mathematics. The best perplexity value of WikiText-2 for the dense model Llama-3.2-3B-Instruct is 10.46. The highest accuracy on the GSM8K dataset is 0.71

However, degradation correlates with sparsity. Models with 20% and 30% sparsity retain their mathematical reasoning abilities. In turn, at 50% sparsity, GSM8K score drops to 0.1800, while perplexity reaches 45.1633—this demonstrates that aggressive pruning deprives models of multi-step reasoning capabilities.

Both QLoRA and DoRA algorithms restore some loss. While DoRA achieves better GSM8K accuracy, QLoRA provides comparable perplexity results, especially at 30% and 50% sparsity. Thus, the choice between algorithms depends on the evaluation metric.

Gain in terms of recovery provides us with additional insights. The maximum gain of GSM8K score is obtained during 50% sparse case where DoRA improves the model from 0.1800 to 0.4508. However, the resulting model is not the best one since 20% and 30% sparse models have a better final GSM8K score.

Concerning throughput, we can state that pruning does not increase inference time proportionally in this task. Unstructured pruning makes it impossible for dense kernels to benefit from zeros, thus providing no speed-up. This stage primarily aims at performance restoration.

Summarizing the above information, we conclude that pruning and recovery are two stages inseparably connected to each other. First, pruning alone causes deterioration in model performance, and only QLoRA and DoRA partially restore it. Considering configurations, `wanda_s20 + DoRA` provides the safest choice regarding GSM8K accuracy. `wanda_s30 + DoRA` is an acceptable intermediate solution, and models recovered by 50% pruning are interesting cases of aggressive pruning.

17

Chapter 7

Conclusion and Future Work

7.1 Summary of Work

The presented study addressed the topic of performance recovery for pruned large language models with limited hardware resources. Llama-3.2-3B-Instruct was chosen as a dense reference model. The work proceeded according to the following scheme: dense baseline evaluation, Wanda pruning at certain ratios, raw pruned evaluation, PEFT-based performance recovery, and evaluation of the final recovered model. The order was significant since the results of recovery depended on evaluating dense and pruned baselines using a consistent workflow.

The studied model was pruned using Wanda activation-aware pruning at 20%, 30%, and 50%. Raw pruned checkpoints were analyzed first, and they were used to estimate the impact of pruning directly. As the result, pruning had led to an increase in WikiText-2 perplexity and the decrease in GSM8K accuracy, and it was maximal for 50% sparsity, where GSM8K accuracy was estimated as 0.1800 and perplexity – as 45.1633.

Then QLoRA and DoRA recovery was performed based on each raw pruned checkpoint. The recovered models were assessed for WikiText-2 perplexity and GSM8K accuracy to compare their linguistic qualities and reasoning ability. Both methods provided improvements compared to the baseline, but DoRA outperformed QLoRA on GSM8K accuracy across all levels of sparsity, whereas QLoRA achieved comparable results with dense baselines on WikiText-2 perplexity, especially at 30% and 50% sparsity.

The experiment revealed that Wanda pruning did not necessarily improve the runtime. In the considered experimental conditions, throughput estimates for dense, pruned, and recovered models remained practically equal to one another. Thus, unstructured sparsity by itself does not ensure acceleration, unless the inference framework allows exploiting it. Hence, the primary contribution of the work lies in analysis of pruning damage, PEFT-based recovery, and the trade-off between sparsity, perplexity, reasoning accuracy, and practicability.

7.2 Key Findings

However, the baseline model Llama-3.2-3B-Instruct performed better in all benchmarks with the highest WikiText-2 perplexity of 10.4627 and GSM8K accuracy of 0.7108. None of the recovered models could beat this value, but they were certainly superior to the raw pruned checkpoints.

At a higher sparsity rate, Wanda pruning led to a significant drop in performance. At 20% sparsity, the raw pruned model achieved 0.5046 accuracy and 23.3884 perplexity in GSM8K and WikiText-2, respectively. For the raw pruned model at 30% sparsity, GSM8K accuracy dropped to 0.4831 and perplexity to 25.4533. The damage done by aggressive pruning became especially evident for 50% sparsity when GSM8K accuracy fell to 0.1800 and perplexity rose to 45.1633.

DoRA and QLoRA performed equally well as post-training recovery techniques. In each case of the sparsity rate, the models recovered using them outperformed the raw pruned models. Furthermore, at 20%, 30%, and 50% sparsity levels, DoRA yielded higher GSM8K accuracy compared to QLoRA. However, the best GSM8K accuracy for a recovered model belonged to `wanda_s20 + DoRA` model with the value of 0.5415.

Nevertheless, QLoRA remains an effective technique for language modeling recovery. Although DoRA yielded the best accuracy score for the reasoning task, the recovered models with QLoRA provided marginally higher WikiText-2 perplexity at 30% and 50% sparsity rates. As such, depending on the desired metric, different techniques can be more or less efficient. Reasoning accuracy benefits the most from DoRA, while at a higher sparsity rate, QLoRA is preferable.

The highest relative recovery performance occurred at 50% sparsity rate, where DoRA improved GSM8K accuracy from 0.1800 to 0.4508. Still, despite this large performance boost, the resulting accuracy remained lower than that achieved with QLoRA or DoRA at other sparsity levels.

Finally, throughputs revealed that increasing sparsity rate does not necessarily lead to a proportional speedup. In Wanda, unstructured sparsity is obtained, and, therefore, sparse inference engines are needed to achieve any speedup since the dense engines do not benefit from zeros.

7.3 Limitations

Several limitations should be mentioned as well. For instance, all experiments were performed with just one base model Llama-3.2-3B-Instruct, which allowed controlling everything and making the process manageable. However, the obtained results cannot be generalized for other models such as Mistral, Gemma, Phi, or Llama variants.

Furthermore, evaluations were performed based on WikiText-2 and GSM8K datasets. Although these benchmarks allowed analyzing language and math reasoning capabilities, other important capabilities such as commonsense reasoning, factual question answering, code generation, long-context capability, safe behavior, and overall instruction following were not studied.

In addition, the sparsity created by the pruning algorithm was unstructured. While this type of sparsity helped to study the effects of pruning damage and recovery, it did not lead to tangible hardware optimizations in the selected experimental setup. Using structured pruning or hardware-aware sparsity techniques might yield better results if the focus is on runtime speed improvements.

Most hyperparameters of QLoRA and DoRA were selected and left unchanged in the work. These parameters include adapter rank, LoRA alpha, dropout probability, learning rate, number of training epochs, max sequence length, and target modules. However, selecting different values might provide better recovery performance.

Finally, no real deployments have been made for consumer devices including mobile phones, embedded boards, or consumer laptops. Here, the term ”deployment” refers to the process done on an experimental device with constraints. Thus, a deployment study would require testing the process on CPU-only devices, consumer GPUs, mobile NPUs, and sparse inference accelerators.

One more issue is related to using GSM8K for recovery purposes since it enables studying mathematical reasoning recovery but can bias adapters towards this specific kind of reasoning. A more general approach to recovery would result in better language capabilities, whereas task-specific recovery would increase the performance of the specific task.

7.4 Future Work

43 Possible directions for future work include the following:

Investigate the impact of applying the proposed pruning–recovery pipeline to other classes of LLMs and models of other sizes, e.g., Mistral, Gemma, Phi, or other variants of Llama. It could show how the discovered trends are Llama-specific or are observed generally for large language models.

Extend the scope of benchmarks beyond Wikitext-2 and GSM8K, using a wider range of evaluation tasks such as multiple-choice questions, common-sense reasoning, mathematical tasks from various domains, or even code generation to ensure a broader understanding of what was lost after pruning and recovered after fine-tuning.

Investigate structured pruning schemes, such as head pruning, pruning of feedforward channels, block-level pruning etc. Such schemes can potentially achieve higher speed-up compared to unstructured Wanda-based pruning if supported by the backend of a particular imple-

mentation of the architecture.

Apply the proposed method in conjunction with quantization techniques. For example, one could study the potential of using QLoRA technique for post-pruning quantized loading instead of just running a fine-tuned model with quantization. Alternatively, the proposed method could be followed by quantization during inference.

Do additional experimentation for recovery fine-tuning. Improvements can be gained by using adapter rank, learning rate, dropout, number of epochs, modules targeted, and training dataset size.

Run a complete end-to-end study of deployment scenarios for the obtained model, testing its performance on different hardware platforms, including CPU-only setups, consumer-grade GPU, mobile devices with an accelerator or a dedicated framework supporting sparse operations.

Use more diverse training data for fine-tuning the recovered model. For example, instead of training only on GSM8K data we could also add general instruction data, natural language modeling data, reasoning datasets, or any combination thereof to cover a broader set of skills instead of focusing mainly on math reasoning

7.5 Final Remarks

Pruning and recovery are treated together here as one process. The work confirms that pruning alone deteriorates language modeling and reasoning. Pruned checkpoints are not useless since there is an ability to recover a significant share of lost quality using recovery without full fine-tuning techniques such as QLoRA and DoRA.

Since the dense version of Llama-3.2-3B-Instruct provides the best results, the proposed recovery approach can hardly substitute the dense model completely. Rather, recovery helps restore damaged checkpoints without additional full fine-tuning cost.

When comparing recovered models, the best reasoning model is obtained using Wanda-pruning for the 20%-sparse checkpoint. The 30%-sparse model recovered with DoRA appears to be an optimal trade-off between efficiency and accuracy, whereas the partially recovered models with 50% sparsity provide limited recovery and pose high risks for reasoning.

Efficiency should be taken with care, as there was no apparent throughput gain when applying unstructured Wanda-pruning. Real efficiency gain comes only with sparse inference, structured pruning, quantization, and architecture-aware techniques.

The primary message from this work is that there are multiple aspects to consider when trying to make efficient LLMs. An efficient model cannot be characterized by its sparsity, and a recovered checkpoint cannot be characterized by its accuracy alone. The valuable point to keep in mind is how different approaches influence one another.

Bibliography

[1] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

[2] Zijian Feng, Hanzhang Zhou, Zixiao Zhu, Tianjiao Li, Jia Jim Deryl Chua, Lee Onn Mak, Gee Wah Ng, and Kezhi Mao. Restoring pruned large language models via lost component compensation. *arXiv preprint arXiv:2510.21834*, 2025.

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

[4] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[5] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. In *Advances in Neural Information Processing Systems*, 2023.

[6] Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, 2024.

[7] Moritz Wagner, Christophe Roux, Max Zimmer, and Sebastian Pokutta. A free lunch in llm compression: Revisiting retraining after pruning. *arXiv preprint arXiv:2510.14444*, 2026.

Appendix

A.1 Hyperparameters and Configuration

Table A.1: Final pruning, recovery, and evaluation configuration

Hyperparameter / Setting	Value
Base model	Llama-3.2-3B-Instruct
Pruning method	Wanda activation-aware pruning
Sparsity levels	20%, 30%, 50%
Wanda calibration sequences	64
Wanda calibration sequence length	512
Recovery methods	QLoRA and DoRA
Recovery dataset	GSM8K training split
Recovery training samples	7473
Recovery epochs	1
Training steps	934
Adapter rank	16
LoRA alpha	32
Dropout	0.05
Learning rate	2×10^{-4}
Maximum sequence length during recovery	512
Per-device batch size	2
Gradient accumulation steps	4
Effective batch size	8
QLoRA optimizer	paged adamw 8bit
DoRA optimizer	adamw torch
QLoRA quantization	4-bit NF4 with double quantization
Target modules	q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj
GSM8K evaluation sample size	650
GSM8K random seed	42
GSM8K maximum new tokens	512
WikiText-2 evaluated tokens	131072
WikiText-2 sequence length	1024
WikiText-2 stride	512
Final evaluation platform	RunPod NVIDIA A40 GPU
Adapter handling during evaluation	Merged into pruned base model before inference

A.2 Model and Adapter Artifacts

For reproducibility, the models, their checkpoints, recovery adapters, and backup repositories used in this dissertation are available in the following Hugging Face repositories. They include Wanda pruned checkpoints, QLoRA/LoRA recovery adapters, and backup repositories. Links to their Hugging Face repositories are provided below.

Artifact Type	**Configuration**	**Hugging Face Repository Link**
Wanda-pruned model	20% sparsity	paul258/llama32-3b-wanda-s20
Wanda-pruned model	30% sparsity	paul258/llama32-3b-wanda-s30
Wanda-pruned model	50% sparsity	paul258/llama32-3b-wanda-s50
QLoRA adapter	20% sparsity pilot recovery	paul258/wanda-s20-qlora-gsm8k-pilot
QLoRA adapter	20% sparsity full recovery	paul258/wanda-s20-qlora-gsm8k-full
QLoRA adapter	20% sparsity gentle recovery	paul258/wanda-s20-qlora-gsm8k-gentle-v1
LoRA adapter	20% sparsity clean recovery without bitsandbytes	paul258/wanda-s20-lora-clean-gsm8k-v2-no-bnb
Recovery adapters	Fast recovery adapter collection	paul258/llama32-3b-wanda-recovery-adapters-fast
Backup repository	Pruning and recovery backup	paul258/llama-pruning-recovery-backup

Table A.2: Hugging Face Hub repositories containing the Wanda-pruned checkpoints, recovery adapters, and backup artifacts used in this dissertation.