Anchor-Guided Repair: A Defense Mechanism for Enhancing Stability of Compromised Pretrained Language Models Against Low-Precision and Weight Noise Attacks

by

Abrar Mahir Rohan
22101267
Nafiz Khan
22101045
Tahsin Tajwar Tanni
22101744
Fuad Fardin
22101027
Anika Bushra
21201068

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
BRAC University
January 2026

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Abrar Mahir Rohan

22101267

Anika Bushra

21201068

Fuad Fardin

22101027

Nafiz Khan

22101045

Tahsin Tajwar Tanni

22101744

Approval

The thesis/project titled "Anchor-Guided Repair: A Defense Mechanism for Enhancing Stability of Compromised Pretrained Language Models Against Low-Precision and Weight Noise Attacks" submitted by

1. Abrar Mahir Rohan (22101267)
2. Anika Bushra (21201068)
3. Fuad Fardin (22101027)
4. Nafiz Khan (22101045)
5. Tahsin Tajwar Tanni (22101744)

of Fall 2025 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science in 31st January, 2026.

Examining Committee:

Supervisor:
(Member)

Dr. Muhammad Iqbal Hossain

Associate Professor
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Dr. Sadia Hamid Kazi

Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

Large Language Models (LLMs) are increasingly released as open-source and are susceptible to post-release attacks, including weight noise injection and low-precision quantization. Such attacks often have a significant negative effect on model stability and performance, augmenting perplexity and creating unreliable behavior. In practice, users might not have access to original clean weights and may unknowingly download compromised models. This thesis presents a defense mechanism called Anchor-Guided Repair, which aims to improve the stability of weakened LLMs against weight noise and low-precision attacks. The proposed method optimizes a fine-tuned attacked model on clean textual data while mitigating parameter changes via an anchor loss, which penalizes the difference from a clean, task-adapted baseline. This approach reduces instability by optimizing a composite task involving language modeling loss and anchor regularization, without distorting the knowledge acquired by the original model. The proposed method was put to the test across a range of model architectures and attack setups, including low-precision and Gaussian weight noise. The experimental results clearly show that Anchor-Guided Repair outperforms the attacked models consistently, keeping a favorable state of desirable conditions. While the defended model performs slightly below the clean baseline, it is significantly better than the compromised model. This emphasizes the power of anchoring in stabilizing large models without accessing proprietary training data. Altogether, Anchor-Guided Repair is a viable post-deployment defense for weight-level attacks, promoting safer and more trustworthy model reuse in open-source settings.

Keywords: Quantization attacks, Large Language Models security, weight noise injection, adversarial robustness, model compression, backdoor detection.

Acknowledgement

It is our great pleasure to extend our sincere appreciation to our advisor Dr. Muhammad Iqbal Hossain who gave us the most excellent guidance and patience during the course of this thesis and offered us excellent comments as well. His support and experience have proved to be priceless through all the steps of this work. The present study has been conducted with the material available at the University Library to which we are thankful.

Personally, we would like to express our appreciation to our parents, siblings who stood behind, showed love and support, and patience throughout the process. It is not possible without their support.

Thank you all.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Background

The rapid proliferation of Large Language Models (LLMs) has fundamentally transformed the landscape of natural language processing, enabling widespread adoption across domains ranging from code generation to creative writing. However, the immense computational and memory costs associated with modern LLMs often exceeding billions of parameters pose significant barriers to deployment on commodity hardware [6]. To address this, the open-source ecosystem has increasingly relied on Post-Training Quantization (PTQ) techniques, such as LLM.int8() and NF4, which compress model weights from standard floating-point formats (FP16/FP32) into lower-precision representations (INT8 or 4-bit) [2], [17]. This trend has made the powerful AI more reachable, but has brought about new complications in terms of dealing with the activation outliers and loss of weight accuracy [7], [18].

Although quantization is very effective in increasing efficiency, it also creates a very critical point of weakness, sensitivity of model weights. Recent studies have indicated that popular methods of quantization are manipulable, and this can present unhealthy results. A small adjustment of weight parameters can lead to a severe deterioration in the performance or safety of the model [5]. The smaller the bitweight of these models, the smaller the "safety margin" of these weights, and the more likely they will become unstable due to noise [15]. As a result, the integrity of the open-source supply chain heavily depends on the belief that downloaded weights are untouched and reliable. However, this assumption is increasingly being put to the test with the rise of quantization backdoors and weight-level attacks [9], [13].

1.2 Rationale of the Study

The primary motivation for this research stems from a critical gap in the current defense landscape for open-source LLMs: the lack of *user-side* repair mechanisms. Existing robustness techniques, such as noise injection-based training schemes or aggressive quantization-aware training, are predominantly *creator-side* defenses [1], [4]. These methods require access to the full proprietary training dataset and massive computational resources to train a robust model from scratch, which is infeasible for the average end-user.

However, in the real-world open-source ecosystem, users typically download pretrained models that may already be compromised by weight noise or structural instability. When a user encounters a degraded model (e.g., one suffering from high perplexity due to quantization noise), they lack the resources to retrain it and often lack access to the original "clean" weights for comparison [14]. This creates a dilemma where users must either discard the model or deploy unreliable systems that may exhibit degraded code quality or reasoning capabilities [11].

Therefore, there is an urgent need for a post-deployment defense mechanism that allows end-users to "repair" compromised models using limited data and compute. By developing a method that can stabilize weights against noise without requiring access to the original training pipeline, this study aims to secure the deployment of quantized LLMs [20].

1.3 Problem Statement

Large language models (LLMs) are distributed more and more often in open-source repositories and third-party platforms, which allows their wide adoption, but also creates additional vulnerabilities in the model supply chain. In reality, models often get deployed by end users that have been subjected to aggressive post-training changes like low-precision quantization or unintentional corruption of the parameters in the course of storage, transmission or conversion. More importantly, these models could be purposefully weakened by intentional weight fluctuations, which could lead to inadequate performance, unstable generation, or silent failure modes that are challenging to identify during deployment.

Conventional recovery techniques like full retraining or reference-based rectification are impractical because end users usually do not have accessibility to the basic clean pretrained weights or proprietary training data. Unconstrained optimisation frequently results in catastrophic parameter drift, overfitting, or unexpected behavioural changes, particularly when the model has already been damaged by quantisation or weight noise. However, fine-tuning on downstream data may partially restore performance.

Current robustness and repair methods mostly presuppose oracle access to clean model parameters or entirely rely on performance measures aggregated on in-distribution data. The actual limitations of post-deployment repair have been obscured by these assumptions, which also fail to account for task generalisation and the delicate negotiations between computational stability and representational preservation.In addition, the effectiveness of recovery in mixed-precision regimes especially with extreme low-bit quantization has not been studied well yet, even as it has increasingly become an important phenomenon in resource-constrained deployment environments.

As such, whether or not compromised pretrained language models can be stabilized and partially recovered at deployment time under realistic user-side constraints, without conditioning on clean reference weights or re-training on a clean dataset, and coherent behaviour on both in-distribution and out-of-distribution tasks is of critical open interest. To deal with this issue, a principled repair process that limits

the damaging parameter updates, as well as a stringent evaluation framework that can be used to identify the superficiality of the performance improvement and the true recovery of the behaviour, is necessary.

1.4 Objectives

1.4.1 General Objective:

The main aim of the study is to research the possibility of reliably stabilizing and partially recovering at deployment time compromised pretrained large language models without the availability of proprietary training data, or original clean weights, under plausible post-release threat conditions.

1.4.2 Specific Objectives:

In order to accomplish this broad objective, the study will aim at the following specific objectives:

1. To provide an anchor-guided repair model that allows previously trained language models to recover from low-precision quantisation errors and Gaussian weight noise while simultaneously enhancing language models using parameter-level stability restrictions.

2. To formulate and study a two objective loss that trades off between task performance recovery and preserving representations so that parameter drift can be disastrous upon fine-tuning after the attack.

3. To empirically test a realistic user-side defensive strategy versus oracle-based upper-bound defensive, have the ability to measure the recovery that can be achieved when no clean reference weights are available and have meaningful bounds on the post-deployment repair.

4. In order to comprehensively assess robustness recovery in a variety of numerical precision regimes (FP16, INT8 and 4-bit), one would evaluate the amplification or limiting effects of quantization on recovery dynamics with increasing perplexity-calibrated noise severity.

5. Based on data-level perplexity, prompt-wise ordering consistency, variance optimisation, and semantic embedding analysis, both in-distribution and out-of-distribution stability are assessed to determine whether recovered models continue to behave coherently outside the fine-tuning domain.

6. To evaluate instruction-tuned and basic pretrained models in order to examine the impact of model design and pretraining alignment on repair performance.

1.5 Methodology in Brief

In this study, we use a clear, step-by-step evaluation process to examine how weight-noise perturbations reduce robustness in pretrained language models and how controlled recovery procedures can restore performance. The framework separates the

effects of numerical precision, task adaptation, adversarial degradation, and recovery dynamics so that recovery measures stay within clear limits, remain easy to interpret, and follow a consistent method. Moreover, we are conducting experiments on pretrained models such as Qwen2.5-0.5B-Instruct and Pythia-410M using the WikiText-2 benchmark as a standard evaluation corpus. The process is divided into four main stages, consistent with the proposed experimental flowchart.

The process starts by setting a baseline that accounts for precision (Phase 1), where the pretrained model is tested as it is, without fine-tuning, to measure its zero-shot performance. Perplexity is evaluated under FP16, INT8, and 4-bit quantization to measure the separate effect on performance when numerical precision is reduced. These measurements provide baseline values that allow us to separate degradation caused by quantization from changes that arise later from task adaptation and weight perturbations.

This is followed by the Task Adaptation phase(Phase 2.1), where the pre-trained model is fine-tuned on the WikiText-2 training split with the causal language modeling objective to produce a clear, task-specific baseline. At this stage, the model's internal representations are adjusted to match the statistical patterns in the evaluation domain, which sets an upper limit on task-specific performance. The cleaned, adapted model is used as the reference when computing the recovery rate and as the frozen anchor model during defense training.

In the Attack Generation stage (Phase 2.2), the Pre Trained Baseline model is subjected to controlled weight corruption through Gaussian noise injection. In order to approximate the adversarial perturbations, Gaussian noise with zero mean and variance σ^2 is introduced to adapted model weights which is shown in Equation 1.1. This method allows known testing of the robustness of the model in the case of stochastic disturbance.

$$W_{\text{attacked}} = W_{\text{clean\_adapted}} + \mathcal{N}(0, \sigma^2) \tag{1.1}$$

The noise variance, σ^2, is adjusted to hit target performance degradation levels of 10%, 15%, and 20% increase in perplexity compared to the clean adapted baseline. The attacked model is saved in half precision (FP16) and later evaluated under post-training quantization settings (INT8 and 4-BIT) to measure performance loss and understand how weight noise and quantization errors compound.

In Phase 3, the Defense stage recovery is carried out through anchor-guided fine-tuning. We use two models: a task-adapted clean baseline that stays frozen as a reference and a second model that is trained under attack. Training for the attacked model uses a single objective that sums the cross-entropy language modeling loss and a weight-anchoring regularization term. This constraint reduces large shifts in the model's parameters while still allowing it to relearn task-relevant information that may have been disturbed by noise. Hyperparameter λ is used to balance between trade-off between task performance and weight stability that we can see in Equation 1.2.

$$L_{\text{total}} = L_{CE} + \lambda \times \sum \text{mean}((W_{\text{current}} - W_{\text{clean\_adapted}})^2) \tag{1.2}$$

This method allows for performance recovery while keeping excessive parameter drift from the task-adapted clean baseline in check. The clean adapted model acts as a stable reference point (stored on CPU for memory efficiency), while the attacked model remains trainable (on GPU). Fine-tuning is carried out on WikiText-2 with hyperparameters set to $\lambda = 0$, a learning_rate $= 2e-5$, and a maximum of 200 steps. Parameter updates are done in half precision (FP16), with post-training quantization applied only during the final evaluation.

Lastly, the Evaluation stage (Phase 4) assesses how well the defenses perform by analyzing results at both the dataset level and the individual prompt level under several numerical precision formats. Recovery rates are calculated with respect to the task-adapted clean baseline, and we report analyses that assess the stability of model representations and the extent to which variance is reduced. Dataset-level perplexity is measured at FP16, INT8, and 4-BIT precisions, with recovery rate calculated as:

$$\text{Recovery} = \frac{PPL_{\text{attacked}} - PPL_{\text{defended}}}{PPL_{\text{attacked}} - PPL_{\text{clean\_adapted}}} \times 100\% \tag{1.3}$$

The relative improvement of the defense mechanism over the attacked model is the definition of recovery in Equation 1.3 and it is calculated by the performance difference between the attacked and clean adapted models. The greater the recovery percentage, the better the mitigating efforts to address the attack.

A suite of 96 prompts across various task categories facilitates prompt-wise perplexity analysis, validating ordering constraints ($PPL_{\text{clean}} < PPL_{\text{defended}} < PPL_{\text{attacked}}$) and measuring variance reduction to evaluate stability improvements. Both the Qwen 2.5-0.5B-Instruct and Pythia-410M based architectures' defence mechanisms are generally beneficial, according to these studies.

1.6 Scopes and Challenges

1.6.1 What This Study Covers:

The research is targeted at learning about the deterioration and rehabilitation of sturdiness in pretrained language models to the low precision and gaussian noise attack. It also examines performance, stability as well as the extent to which the meaning is adhered to with control disturbances and all with standardized data and ordered prompt evaluation. As noted in this paper, empirical analysis is important instead of redesigning, retraining or retraining at a large scale.

Technical Focus

The technical aspects of this paper are the investigation of the effect of low-precision computation and noise-based attacks on the legitimacy of pretrained language models. By employing regularised fine-tuning to restore model performance without overcorrection, it investigates the history of controlled recovery. The analysis also looks at performance trend over time and the variation to determine stability over time during the recovery process. Maintaining semantic embeddings is considered as

one of the constraints in order to keep the behavior of the model and the integrity of representation. Finally, the method's sensitivity and generalisability are evaluated by validating it across model families.

1.6.2 What This Study Does Not Discuss:

This paper is very specific in its approach because it is restricted to robustness degradation and recovery when the conditions are low-precision and noisy. Consequently, it fails to consider the methods of adversarial, including prompt injection or data poisoning attacks, which act at a dissimilar layer of the threat model. The modifications to transformer architectures are also not included in the work since the goal is to examine recovery strategies without changing the model structure. Moreover, we do not speak of multilingual assessment conditions and experiments are done on monolingual standards. Lastly, the paper fails to consider deployment pipelines, commercial system-wide defense mechanisms because its subject of focus is on model-level analysis and not manufacturing environments.

1.6.3 Key Challenges:

One of the key issues in this paper is control of trade-off between recovery and stability in the process of post-quantization repair. Although aggressive adaptation can be useful to recover performance, large amounts of parameter drift can be brought about by aggressive adaptation resulting in instability or overfitting. On the other hand, too conservative adaptation can save the day but cannot recover anything meaningful. The other important problem is the computational cost of prompt-level analysis. The explicit, fine-grained and prompt-wise evaluation is expensive in terms of computational cost, particularly when sensitivity evaluation on various levels of perplexity-calibrated noise as well as model settings is required. In spite of these challenges, the given study provides a firm and comprehensive layout of determining the extent to which pretrained language models can restore their resilience.

1.7 Thesis Structure

Chapter 1 introduces the study by providing background information, the rationale of the research, the problem statement, objectives, an overview of the methodology, and the scope and challenges of the work.

Chapter 2 presents a comprehensive literature review focusing on low-precision computation and weight noise attacks in large language models. It discusses existing research, key findings, and summarizes insights related to quantization, weight sensitivity, and robustness, establishing the foundation for the proposed approach.

Chapter 3 outlines the requirements, impacts, and constraints of the study. This chapter discusses final specifications, societal and environmental impacts, ethical considerations, relevant standards, project management planning, risk management, and economic analysis.

Chapter 4 describes the proposed methodology in detail. It covers the overall design process, baseline establishment, attack simulation framework, defense mechanisms through anchored fine-tuning, and multi-precision evaluation. The chapter also presents the preliminary model specification, theoretical framework, data sources, and implementation details.

Chapter 5 focuses on result analysis and evaluation. It includes performance evaluation, statistical analysis, comparisons between models, and discussions on stability, precision, and recovery relationships. The chapter also provides an in-depth discussion of findings and comparisons with existing defense mechanisms.

Chapter 6 concludes the thesis by summarizing the key findings, highlighting the contributions of the study to the field, and offering recommendations for future research directions.

Chapter 2

Literature Review

2.1 Literature Review on Low-Precision and Weight Noise Attacks in LLMs

In this part, the literature examines the current studies on low-precision computation and weight noise perturbation in Large Language Models (LLMs) and their effects on model stability, robustness, and security. Reducing numerical precision (i.e. FP16 to INT8 or INT4) and adding noise, both accidentally during hardware limitations and deliberately during adversarial attacks, have been found by prior work to cause the performance and reliability of a model to decline significantly. This knowledge of their effects is essential to develop an efficient defense mechanism, e.g. anchor-guided repair, against the compromised models.

2.1.1 Introduction to Low-Precision Computation and Gaussian Noise Attacks

Low-precision computation is also common in deep learning models to achieve smaller memory footprint and computation by quantizing model parameters and activations with smaller bit-widths FP16, INT8 or INT4. Though these methods allow an effective implementation of LLMs on resource-limited hardware, they may cause numerical instability and accuracy reduction.

Besides the loss of precision, the presence of Gaussian noise in model weights, whether as a result of hardware bugs, transmission noise, or as a result of adversarial attacks, is also a significant vulnerability to model robustness. Weight noise attacks are based on the fact that the parameters of LLM are sensitive to noise, and the resulting performance collapses with noise or the outputs become unpredictable. This section gives background information regarding low-precision arithmetic as well as Gaussian noise, which is critical to the behavior of LLM, which paves the way to defense-oriented research.

2.1.2 Review of Existing Research

Patrik Czakó et al. [18] conducted a systematic review focusing on weight-activation quantization in Large Language Models (LLMs), with a specific emphasis on the

emergent outlier phenomenon in LLM activations. They were mostly aimed at assessing and summarizing recent methods to reduce these activation outliers and improve the efficiency of quantization, which is what makes their work stand out unlike other reviews of the field since they are specifically focused on this. In order to do this, they cautiously considered 52 recent studies according to the PRISMA methodology. This entailed systematic literature. filter through academic databases with targeted keywords and stringent inclusion criteria to select the papers that suggest practical post-training quantization (PTQ) techniques to be used in multi-billion parameter LLMs to address the activation and KV cache issues. Their greatest value is that they provide a list of various techniques and implications on the quantization of activation, which forms the groundwork of future research and actual progress in implementing LLM in the future. They classified these methods as mathematically equivalent manipulations (including channel-wise scaling, shifting and rotation), high-precision storage (mixed precision), finer granularity (smaller quantization groups) and non-linear datatypes. The result of the review showed that although the outcome of mathematically equivalent transformations, particularly channel-wise scaling, partly alleviate outliers by relocating quanti-wise. zation difficulty to weights, such strategies as channel-wise shifting, rotation can deal with more complicated outliers (e.g., massive outliers). It was demonstrated that mixed-precision techniques were essential in KV cache quantization and strategies such as GEAR and ZipCache were effective. Nevertheless, the analysis showed that, regardless of improvements, weight-only quantization reduces the loss of accuracy less than weight-activation methods, yet activation quantization is essential in achieving the highest inference speed and energy efficiency. Datasets (i.e., WikiText-2 and GSM8k) were also commonly used in the studies reviewed and are publicly accessible as perplexity and accuracy evaluations respectively.

Kazuki Egashira et al. [5] examined the negative security consequences of popular LLM quantization, and found that the techniques can be used to construct a malicious quantized LLM even when the full-precision variant does not seem harmful. They were essentially intended to expose this new danger, which might possibly fool users to install malicious quantized models. They have solved this issue in terms of three-stage attack framework. To start with, they have inoculated malicious behavior in an LLM through fine-tuning on an adversarial task (e.g., vulnerable code generation). Second, they counted this malicious model to compute constraints of all full-precision models that would quantize to the same malicious state. They focused on well-known zero-shot methods, such as LLM.int8, NF4, and FP4, which enable the derivation of such crucial constraints by normalising weights within blocks and rounding them to a given alphabet. Third, they combined Projected Gradient Descent (PGD) to trim the malicious behavior of the full-precision model without altering its weights so that they would fit within the derived constraints to produce a benign full-precision model, which, when quantized, would revert to its malicious behavior. Their work is the first significant experimental demonstration of this weight quantization exploit of the usefulness of this technique on a large scale and points to the fact that there is no defense at present on model-sharing platforms. This approach is based on the PGD-based model design in which the fine-tuning process consists of a malicious and clean objective to keep utility. The result proved the attack in the three configurations as vulnerable code generation where quantized

models had a large impact on increasing the percentage of vulnerable code generated; over-refusal attack where quantized models exhibited a refusal rate of up to 39.1; and content injection where quantized models had a large effect on increasing the percentage of target phrases mentioned. They also noted that the models with long tailed distribution of weights (e.g. phi-2) were more prone to this attack. Data applied in their experiments were code-alpacas instruction tuning, and a subset based on the dataset of He and Vechev on vulnerable code generation, databricks-15k on over-refusal, and test datasets MMLU, TruthfulQA, HumanEval and MBPP. All these datasets are usually publicly accessible.

Zechun Liu et al. [8] wanted to deal with the severe quantization errors of LLMs that come about as a result of outliers in weights, activations and the KV cache. Their essential aim was to improve the accuracy of quantization by employing learned rotation matrices to adequately reduce such outliers. They resolved this issue by proposing SpinQuant, which involves the opportunity to employ these learned rotation matrices to achieve the best per-quantized network performance. SpinQuant uses rotational in-variance in Transformer designs to generate pairs of rotation matrices using identity mappings, which can be aggregated into local weights with no changes to the full-precision network outputs. These rotations are chosen to give outlier-reduced distributions hence making quantization simpler. Their primary contribution is the creation of SpinQuant, which is the first attempt to make use of learned rotations, which outlier mitigation, which has been shown to be effective in quantized LLMs. SpinQuant greatly minimized the difference in accuracy between LLaMA-2 7B models. With 4-bit weight and activation quantization and KV-cache quantization all set to just 2.9 of full precision, it outperforms current methods, such as SmoothQuant and LLCM-QAT. The design of the model includes learning individual rotation matrices (R1 of residual, R2 of head-wise) and initializing them with random Hadamard matrices and optimized by Cayley SGD. GPTQ is used to apply the final quantization of weight after optimization. The results were quite impressive: SpinQuant demonstrated the state-of-the-art performance being more efficient than random Hadamard rotations all the time. In the case of LLaMA-2 7B with W4A4KV4 environment, the average accuracy of SpinQuant was 71.2, and the perplexity was 3.8, which is much closer to the FP16 baseline compared to others. The optimization was also efficient and it only took minutes or hours to complete. The qualitative nature of their visualizations validated that rotation is a successful approach to reducing extreme values and eliminating outliers in activation and weight distributions. The study was mainly based on the Wiki-Text2 testset on perplexity and eight zero-shot commonsense reasoning tasks on accuracy. WikiText-2 was also applied as a calibration set of rotation optimization and GPTQ. These are datasets that are made publicly available.

Dongwei Wang and Huanrui Yang [1] aimed at increasing the resiliency of InMemory Computing (IMC) Deep Neural Network (DNN) hardware to adversarial input and weight attacks. Their main aim was to compensate the intrinsic low noise margin and accuracy loss presented in the analog IMC silicon. They have resolved this by coming up with a new training scheme of DNN which incorporates the measured IMC noise and violent partial sum quantization on the IMC crossbar. It is based on an input-splitting algorithm and splits DNN layers, calculating partial sums which

are aggressively quantized to 1-bit or 2-bit values, enabling the DNNs to be trained to these lower-precision computations. This method avoids the use of high resolution Analog-to-Digital Converters (ADCs), hardware is simplified to comparators or even 2-bit ADCs, and most critically, they postulated that such aggressive quantization was able to conceal small adversarial perturbations, which enhanced robustness. Their primary work is this innovative training scheme of DNN, which is suitable to reinforce IMC hardware not only against adversarial input (e.g., PGD) but also against weight (e.g., bit-flip) attacks. It is noted as the first work to comprehensively investigate both attack types and successfully integrate actual IMC prototype chip results for improved adversarial robustness. The model design involves training DNNs with injected noise from IMC hardware and specifically adapting to 1-bit or 2-bit partial sum quantization. Key pre-processing includes the generation of adversarial examples via techniques like Projected Gradient Descent (PGD) for input attacks. The outcome demonstrated significant robustness improvements: for PGD input attacks, black-box adversarial accuracy increased by up to 10%, and against bit-flip weight attacks, the scheme required over 30% more bit-flips to degrade performance. The specific dataset names used for training and evaluation are not explicitly mentioned in the provided excerpts, but the context implies standard datasets for image/speech recognition, which are typically publicly available for DNN research.

Ji Lin et al. [7] introduced Activation-aware Weight Quantization (AWQ), a hardware-friendly method for low-bit weight-only quantization in LLMs. Their main aim was to enable the cost-effective implementation of LLMs on edge devices by minimally reducing memory and bandwidth requirements and maintaining the accuracy and generalization of the model. One of the main observations that they had made regarding this issue is that not every weight matters. They did not work on the magnitude of weights but rather found that salient weights are the ones that can be determined by monitoring the activation distribution due to the fact that the weights that produce a larger magnitude of activation process more crucial features. To solve this, AWQ employs a per-channel scaling method that identifies these salient channels and scales them up to reduce their relative quantization error, thereby avoiding hardware-inefficient mixed-precision implementations. The method does not rely on backpropagation or reconstruction. Their main contribution is AWQ's ability to offer a simple yet highly effective solution for low-bit weight-only LLM compression, consistently outperforming existing methods across various benchmarks and being uniquely applicable to instruction-tuned and multi-modal LLMs due to its superior generalization. The model design centers on this per-channel scaling based on activation magnitudes, with the optimal scaling factor determined by a fast grid search. Weight clipping is also applied to minimize MSE. Coupled with AWQ, they developed TinyChat, an efficient inference framework that demonstrated over 3x speedup compared to FP16 implementations on various GPUs, even enabling Llama-2 70B deployment on mobile GPUs. Outcomes showed near-lossless performance on language modeling (WikiText-2 perplexity), visual-language tasks (VILA models), and programming/math problems (MBPP, GSM8K), where AWQ (INT4-g128) often matched original FP16 performance. TinyChat delivered 1.2-3.0x speedup over other edge inference systems. For dataset usage, AWQ utilizes a small calibration set from the Pile, and for evaluation, relies on WikiText-2, VILA datasets, MBPP,

and GSM8K. All these datasets are publicly available.

Zhang et al. [4] aimed to enhance the robustness of Artificial Neural Networks (ANNs) and Spiking Neural Networks (SNNs) against adversarial attacks, addressing the vulnerability of ANNs to imperceptible input perturbations that can mislead models. They propose a novel noise injection-based training scheme (NIT), developing a likelihood ratio (LR) method to estimate gradients for both synaptic weights and noise levels during stochastic gradient descent, and an approximation (NIT-S) designed to reduce memory consumption and improve computational efficiency. Their main contribution is demonstrating that this scheme significantly improves adversarial robustness and achieves slightly better original classification accuracy compared to conventional methods like spatio-temporal backpropagation (STBP). For model design, ANNs incorporate independent standard normal noise added to each neuron's logit output, while SNNs are constructed as a fully connected network with one hidden layer (50 neurons) using Leaky Integrate-and-Fire (LIF) neurons and cross-entropy as the loss function. Key pre-processing for experiments on the MNIST and Fashion-MNIST datasets involved randomly splitting the MNIST dataset into training, validation, and testing sets in a 7:2:1 ratio, and selecting 1000 random images for adversarial testing with perturbation limits (e.g., a maximum perturbation of 0.1 per pixel and specific step sizes for iterative attacks). The outcomes showed that NIT achieved an optimal classification accuracy of 92.3% on original MNIST samples and significantly increased accuracy under various adversarial attacks (e.g., an 81.4% accuracy under the FGSM attack compared to 42.6% for STBP), with the simplified NIT-S offering comparable robustness with improved efficiency, and a combined NIT-S+ providing a balanced performance. While the sources explicitly mention the use of the MNIST and Fashion-MNIST datasets, they do not explicitly state whether these datasets are publicly available for research purposes.

Dongwei Wang and Huanrui Yang [14] aimed to resolve the challenge of "sensitive" or "outlier" weights in LLM quantization, which typically require special high-precision treatment, hindering efficient hardware deployment. Their primary purpose was to enable these outlier weights to be quantized to the same low bit-width as other weights without compromising model performance. Their solution, Noise Perturbation Fine-tuning (NPFT), is an efficient fine-tuning method that reduces the loss Hessian trace concerning these outlier weights.They detected outliers and applied random weight perturbation on those in a Parameter-Efficient Fine-Tuning (PEFT) procedure. To calculate the Hessian trace, NPFT calculates the expected model loss in the case of these random perturbations, thus will not involve the calculations of higher-order gradients which are computationally intensive. Their principal contribution is that NPFT can achieve the efficiency of the reduction in sensitivity of outlier weights, and thus they can be uniformly quantized and much better counterpointed model performance across different post-training quantizers. It is important to note that NPFT enabled the simple Round-To-Nearest (RTN) quantifier to perform at equivalent or better performance than GPTQ on the LLaMA2-7B-4bits benchmark. Hypothesis The model design is determined by locating outlier positions with the fisher matrix and perturbation ratio and applying channel-wise noise to the outlier positions making them zero-mean. For efficiency, they combine LoRA

adapters on self-attention and MLP weights, and perturbations on the weights of the base model. The result indicated that NPFT reported consistent steady perplexity enhancement on both uniform (RTN, GPTQ) and non-uniform (SqueezeLLM) quantizers. As an example, NPFT enhanced the perplexity of RTN on the OPT-1.3B-4bits on the C4 and WikiText data sets. One of the main results of their verification of the theoretical insight was high sensitivity to outliers reduction that occurs after NPFT. Their experiments included LLaMA and OPT models, where there was the use of the dataset C4 in language modeling and calibration, and the WikiText2 dataset in measuring the performance on unseen data. Both data are publicly accessible.

Qianli Wang et al. [21] undertook a comprehensive study to investigate the unexplored impact of quantization on LLM explainability and interpretability. Their core purpose was to understand how quantization influences a model's internal decision-making processes and the quality of explanations it generates, recognizing transparency as a critical requirement for LLM deployment. They approached this by conducting extensive experiments using three common post-training quantization (PTQ) techniques (GPTQ, AWQ, and bitsandbytes) at various bit widths, evaluating them against two explainability methods (counterfactual examples (CFEs) and natural language explanations (NLEs)) and two interpretability approaches (knowledge memorization analysis and latent multi-hop reasoning analysis). This was supplemented by a human user study to assess the subjective quality of explanations. Their main contribution is the finding that quantization can significantly and unpredictably affect model transparency, with the impact varying based on the quantization method, the specific explainability/interpretability approach, and the evaluation metric. They revealed that quantization generally leads to information loss, which is more pronounced in smaller models, but surprisingly, lower-precision LLMs can sometimes outperform higher-precision ones, and a quantized model might even generate better explanations than its full-precision counterpart. The study's model design involved comparing full-precision LLMs with their quantized versions, utilizing existing tools like FIZLE for CFEs and ZeroCoT for NLEs, and specific methodologies for knowledge memorization and latent multi-hop reasoning. The outcomes were multi-faceted: Automatic evaluations showed quantization generally degrades CFE quality, especially fluency, and impacts knowledge memorization, with most information loss concentrated in the final two layers. The human user study indicated that full-precision models' explanations were perceived as slightly more trustworthy and coherent. Across methods, integer quantization (bitsandbytes-8) consistently yielded higher-quality explanations. The datasets central to this research included eSNLI for NLEs and CFEs, LRE for knowledge memorization, and TwoHop-Fact for latent multi-hop reasoning. Additional experiments used AG News and HealthFC. All these datasets are generally publicly available.

Linyang Li et al. [3] introduced a novel watermarking strategy that embeds watermarks into the quantization process of LLMs. Their central purpose was to safeguard model weights and prevent malicious or unlicensed use of open-source LLMs, thereby enabling LLM providers to assert ownership. They achieved this by designing watermarks that are visible when the model operates in full-precision (FP32) mode but remain hidden when the model is quantized to INT8. This means users can

utilize the quantized model normally for inference, but any attempt to fine-tune or use a different quantization strategy in full-precision would reveal the watermark. The authors' main contribution is being the first to integrate quantization strategies into the LLM watermarking domain. Their proposed "interval optimization" method allows the embedding of both text-agnostic and text-related watermarks into open-source LLMs like GPT-Neo and LLaMA. This approach offers more control than trigger-based methods (as the quantized model is watermark-free) and is more practical than decoding-specific methods. The model design hinges on the "interval optimization" method, which exploits the computational gap between full-precision and quantized representations to embed watermarks while preserving the quality of text generated by the quantized model. They used INT8 quantization as the target and implemented a multiple-random-test strategy for verifying the watermark's success. The outcome demonstrated successful watermark planting in both text-agnostic (always active in FP32) and text-related (activated by specific inputs) scenarios, achieving high success rates and maintaining normal text generation in quantized mode. They noted that larger models like LLaMA-7B were more challenging to watermark than smaller ones like GPT-Neo-2.7B. While further pretraining could remove watermarks, the core concept remains promising. For datasets, they utilized a subset of the Wiki corpus (specifically contexts from SQuAD) for trigger construction and a mix of PubMed and WritingPrompts for general training data. All referenced datasets are generally publicly available.

Ting-Yun Chang et al. [15] investigated the fundamental question of why specific inputs lead to disproportionately large quantization errors in low-bit LLMs. Their core purpose was to provide a mechanistic understanding of these errors and identify the critical model components responsible for maintaining performance under quantization. They approached this by analyzing various 3-4 bit weight-only quantization methods (GPTQ, AWQ, NF, EQAT) across a range of LLM sizes (7B-70B). They found strong correlations between quantization errors across different methods, indicating that they tend to fail on the same inputs. To explain this, they developed a hypothesis linking smaller residual stream magnitudes to error amplification through layers, contrary to prior beliefs that large activations were solely detrimental. They then applied LLM localization techniques, including early exiting and cross-model activation patching, to pinpoint the source of these errors. Their main contribution is providing a mechanistic explanation for large quantization errors, identifying the MLP gate projections as crucial components for perplexity preservation. They also revealed that long-tail examples tend to experience less degradation after quantization than other examples, challenging prior assumptions. The model design for their analysis involved applying standard quantization methods to various LLMs and then meticulously analyzing their internal states. Key analytical techniques included quantifying example-level quantization error via NLL differences, measuring residual magnitudes across layers, and using early exiting to assess hidden state readiness. Crucially, they employed cross-model activation patching (CMAP) to isolate the impact of specific activation types (e.g., MLP gate outputs). The outcomes showed: High correlations (average $\rho = 0.82$) in errors across different quantization methods, with significant overlap in large-error examples. Smaller residual magnitudes in upper layers strongly correlated negatively with larger quantization errors. Patching MLP gate outputs substantially reduced perplexity errors, while MHA

patching had minimal effect. Surprisingly, long-tail examples (low NLLs) exhibited smaller quantization errors, and models showed the lowest degradation for the lowest and highest popularity examples on factual knowledge tasks. The research primarily used a 10k example dataset from the FineWeb corpus for NLL evaluations and PopQA for studying factual knowledge and entity popularity. Calibration sets for the quantization methods often used C4 or RedPajama. All these datasets are publicly available.

Xing Hu et al. [19] studied performance loss in post-training quantization of LLMs. They focused on how uneven data distributions and outliers affect quantization accuracy. Their main goal was to create a systematic method to improve the fit of model parameters and activations during quantization. They introduced a new metric, Quantization Space Utilization Rate (QSUR), to assess how well transformed data fits within the quantization space. Building on this, they proposed OSTQuant. This framework uses learnable orthogonal and scaling transformation pairs to globally reshape distributions without changing network outputs. To tackle optimization with limited calibration data, they developed KL-Top loss. This approach uses top-k logit alignment to maintain semantic richness and minimize overfitting. Their findings showed that OSTQuant achieves top performance across various quantization settings, including the difficult W4A4KV4 configurations. It outperformed previous methods like GPTQ and QuaRot while ensuring high efficiency and low memory use.

Jungwoo Park et al. [20] looked into whether activation outliers in large language models (LLMs) are a result of their design or a result of training methods. Their main goal was to create a scalable pre-training framework that stops these outliers from forming at the start. This approach allows for efficient low-bit quantization. They developed the Outlier-Safe Pre-Training (OSP) framework, which includes the Muon optimizer, Single-Scale RMSNorm, and learnable embedding projections. They significantly reduced outlier behavior without changing the model architecture or adding major computational costs. The study showed that models trained with OSP had nearly zero excess kurtosis and were much more stable under heavy 4-bit quantization, scoring an average of 35.7 compared to 26.5 for models trained with Adam. Additionally, the research challenged earlier beliefs by demonstrating that attention sinks, which were thought to create outliers, can exist without causing large activations. These findings changed the understanding of quantization errors in LLMs, suggesting that outliers can be avoided through careful training design.

Mohammad Mozaffari et al. [12] introduced SLIM, a one-shot compression framework designed to lower the computational and memory costs of large language models (LLMs) without sacrificing performance. The main issue it addresses is how to compress LLMs effectively while maintaining performance, especially with structured sparsity constraints like NVIDIA's 2:4 pattern. SLIM incorporates three main parts: a probabilistic uniform quantization approach (SLIM-Quant), a sparsification step using existing one-shot pruning techniques, and a new saliency-based low-rank adapter module (SLIM-LoRA). Unlike traditional methods that need retraining, SLIM calculates the best low-rank adapters analytically using an additive and invertible saliency function based on weight-activation interactions. The authors showed that SLIM outperforms leading one-shot methods such as SparseGPT and L2QER in

both structured and unstructured sparsity conditions, achieving up to 5.66% higher accuracy on LLaMA-2-7B with 2:4 sparsity and 4-bit quantization. Additionally, SLIM reached up to a 4.3× speedup and a 0.23× memory reduction, while also providing optional lightweight fine-tuning with quantized low-rank adapters to further narrow the gap with dense models. The results highlight that high-performance compression is possible without costly retraining and suggest that saliency-aware methods can be essential in balancing efficiency and accuracy for real-world LLM deployment.

Dettmers et al. [2] looked at quantization in terms of performance and accuracy, but their findings also offered insights for adversarial robustness. Their main goal was to reduce the memory usage of large language models (LLMs) by compressing weights to 3 to 4 bits while keeping nearly the same accuracy. To do this, they introduced SpQR, a hybrid representation that separates outlier weights causing significant quantization errors and stores them in higher precision while compressing the others. A key takeaway from their work was the discovery of structured patterns of quantization sensitivity in LLMs, including row and column outliers and artifacts specific to attention heads. These patterns informed their efficient compression approach and highlighted vulnerabilities that could be exploited in adversarial situations. Although their research was not intended as a security paper, it emphasized the need for quantization-aware model design and demonstrated that quantization sensitivity varies widely, making some parts of a model significantly more fragile than others.

Brian Chmiel et al. [17] examined the important question of whether large language models can be fully trained using ultra-low-precision formats, specifically FP4. Their main goal was to show that end-to-end FP4 training can work across weights, activations, and gradients, which were thought to be unstable at such low bit widths. They ran extensive experiments to find the best block sizes and scaling formats, eventually identifying NVFP4 (E2M1 data with E4M3 scale and block size 16) as the most effective setup. They also looked into quantization stability and introduced a split rounding strategy: stochastic for backward/update and round-to-nearest for forward, to stabilize training. Importantly, they found a theoretical limit where gradient magnitudes become too small for FP4 to be effective and suggested quantization aware fine-tuning as a solution. Their results showed that FP4 training, when set up correctly, performs as well as BF16, confirming its practical use for large-scale LLMs.

Ma et al. [9] studied the rise of hidden security threats in commercial post-training quantization workflows. Their main goal was to look into how inactive backdoors could be added to float-32 models and later triggered only through quantization with common frameworks like TensorFlow Lite and PyTorch Mobile. They introduced the "PQ backdoor" concept, where a backdoor stays completely inactive in the full-precision model, even when triggers are present, but activates with high success rates once the model is quantized. The study revealed that four prominent backdoor detection systems Neural Cleanse, STRIP, ABS, and MNTD were unable to find the inactive backdoors. This failure makes traditional inspection tools ineffective. The authors also showed that this vulnerability exists across different

datasets and architectures, exposing a serious flaw in current deployment practices. A key finding was the demonstration that even industry standard frameworks could unknowingly support adversarial behavior after quantization. They suggested that injecting Gaussian noise might help partially, but it is not a complete solution.

Hasan [6] has described the methods of quantization techniques to optimize Large Language Models (LLMs), with the main goal being Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT). The researchers tested many dense-neural network structures of different sizes using synthetic datasets that are meant to match real-world LLM tasks. This study showed that quantization, especially in the case of introduction of adaptive scaling factor, could cut the model size by up to 68 percent and the computation cost by up to 60 percent without changing the performance to more than six percent of full-precision baselines. Performance was measured with the use of such metrics as perplexity, BLEU score, inference latency, and power consumption. The results suggest that quantization is also relevant when deploying LLMs in low-resource settings since it can provide a high level of efficiency with minimal accuracy loss.

Melin et al. [11] compared the code generation capabilities of smaller open-source Large Language Models (LLMs) systematically and analysed the effect of quantization to 8-bit and 4-bit on the same. Based on benchmarked models including HumanEval Plus and MBPP Plus, in addition to metrics including code bleu and static analysis through SonarQube, the study revealed all the models had problems with functional correctness and the best benchmark pass rate was 25%. Quantization turned out to be mixed: on one hand, it helped some models to perform better on simpler tasks; on the other hand, it negatively impacted the performance of others. Moreover, the generated codebase had more than 21 percent of maintainability and reliability problems. By drawing conclusions, the authors suggested that applying quantization allows LLMs to be more resource-friendly to their users, but LLM-based code frequently underperforms, which motivated the necessity of a thoughtful assessment prior to the use of LLM-generated code in practice.

Malinovskii et al. [10] compared and contrasted the theory and practice of large language model (LLM) quantization, where they developed and proposed the so-called linearity theorem, which was formally used to relate per-layer quantization errors to the overall model performance. In experiments to evaluate their approaches on standard benchmarks including WikiText-2 and various zero-shot and few-shot reasoning tasks, the authors also suggested the HIGGS quantization technique which was data-free/hardware-friendly since it used Hadamard rotations to achieve the best quantization. They showed in their Llama-3 and Qwen experiments that HIGGS retained superior performance to the previously published data-free quantization methods across the 3, 4 bits regime, and was able to reach perplexity and accuracy comparable to full-precision baselines at a much higher level of computational efficiency. The work is an in-depth introduction and a useful practical tool to a resource-efficient LLM deployment, in particular, in resource-limited settings.

Proskurina et al. [13] examined the effects of post-training quantization, i.e., utilization of the GPTQ method, on the confidence and calibration of large language

models (LLMs) including BLOOM, OPT, Mistral-7B, and LLaMA-7B. Testing on a number of commonsense reasoning and question-answering benchmarks, the study discovered that 4-bit quantization tended to replace calibration error with an overall decrease in model confidence, especially in the predictions where the original model tended toward skepticism. Although the overall accuracy losses were not very big, the confidence estimates' credence was seriously undermined, particularly in minor models. The quantized LLMs will possibly need extra adjustment before they can be assured of reliable output in sensitive issues, the authors concluded.
Chen et al. (2025) [16] carried out an in-depth evaluation of the safety risks that come with quantization in Large Language Models (LLMs). They specifically looked into the balance between how well the model performs and its safety alignment, especially in environments where resources are limited. Their research examined popular quantization methods, including Post-Training Quantization (PTQ) techniques like AWQ and AQLM, as well as Quantization-Aware Training (QAT) methods such as LLM-QAT and QLoRA, across different bit-widths (INT4, INT8). The authors found that quantization tends to weaken safety features, with lower bit-widths (like INT4) leading to much higher Attack Success Rates (ASR) compared to models that use full precision. They also discovered that calibration datasets, which are often used to maintain utility, can unintentionally increase these safety risks if they include even indirectly harmful samples. To address these vulnerabilities, the authors introduced Q-resafe, a framework designed for quantization-aware safety patching. This approach uses a safety-patching dataset created through knowledge distillation from the model before quantization and applies Direct Preference Optimization (DPO) to realign the quantized model. A standout feature of their method is the identification and selective updating of "safety-critical weights" using SNIP scores, which helps the model restore its safety mechanisms while keeping most parameters unchanged to maintain utility. Their experimental results indicated that Q-resafe effectively realigned quantized models to nearly the same safety levels as pre-quantization, all while keeping computational demands low.

2.2 Summary of Key Findings

The literature review demonstrates three main trends that motivate the study of the topic of Anchor-Guided Repair: the emergence of quantization as a new threat, the substantial impact of the weight of the outliers on the stability of the model, and the inefficiency of the existing defense mechanisms against weight noise.

2.2.1 Quantization as a Security Issue

One of the most recent trends in research is the shift of the perception of quantization as just a compression method to a crucial security issue. Although conventional methods such as GPTQ and AWQ are designed to maintain the utility, they inadvertently leave loopholes in the compromised models.

Dormant Backdoors: Egashira et al. [5] and Ma et al. [9] demonstrate that quantization may be controlled in order to create sleeper agents. The full-precision (FP32) model can contain malicious code that is detected as harmless before the model is quantized (e.g. to INT4) and then the malicious code starts running.

This goes directly in support of the thesis that deals with the Compromised Large Language Models.

Safety Degradation: Chen et al. [16] and Proskurina et al. [13] disclose that aggressive quantization particularly 4-bit, necessarily causes safety alignment and confidence calibration to degrade, leading to a higher Attack Success Rates (ASR) and other problems such as overrefusal or vulnerable code generation.

2.2.2 Weight Sensitivity and the "Anchor" Hypothesis

The thesis is well supported in the literature which points to the fact that not every weight is equal; model stability is determined by specific anchor weights.

Outlier Dominance: Czakó et al. [18] and Chang et al. [15] find that quantization errors are not distributed, but are propagated by activation outliers and certain parts such as MLP gate projections.

Saliency & Sensitivity: Lin et al. [7] (AWQ) and Dettmers et al. [2] (SpQR) confirm that the key to performance of a method is to isolate salient or outlier weights and maintain them in high precision. Wang and Yang [14] also indicate that sensitive weights have to be fine-tuned with noise perturbation (NPFT) to avoid degradation.

Thesis Implication: These results provide the theoretical grounds on which we refer to Anchor-Guided Repair. When just a few of the weights are the cause of sensitivity to quantization, then the strategy of defense based on fixing or stabilizing the individual anchors makes much more sense than retraining the whole model.

2.2.3 Robustness Against Weight Noise and Perturbations

According to recent research, there exists a great divide in our defense capability against weight. noise attacks that are basically formed in the process of quantization.

Noise Injection Training: Wang and Yang [1] and Zhang et al. [4] demonstrates that one can enhance resilience to injected noise in training against adversarial input and bit-flips attack. Nevertheless, the following methods are usually applied during the pre-training stage or in normal fine-tuning.

The Repair Gap: While Chen et al. [16] also propose Q-resafe to assist with re-aligning models through preference optimization, and Liu et al. [8] apply SpinQuant to adjust matrices to outlier remedies, one can find an apparent lack of post-training repair tactics that explicitly target to stabilize models already damaged by the twin issues of low-precision conversion and weight noise.

2.2.4 Conclusion and Correlation with Thesis Objective

Findings of existing research clearly shows that quantization can trigger hidden vulnerabilities, and the stability of a model relies on a few sensitive weights, often referred to as anchors. Yet, most current approaches mainly aim to maintain accuracy or identify backdoors, rather than focusing on restoring stability when a model is compromised.

This thesis addresses this critical gap by proposing Anchor-Guided Repair. By identifying the sensitive "anchor" weights highlighted by Lin et al. [7] and Chang et al. [15], and applying targeted repair mechanisms resilient to the noise identified by Wang and Yang [1], this research aims to immunize LLMs against the specific quantization exploits demonstrated by Egashira et al. [5].

Chapter 3

Requirements, Impacts and Constraints

The chapter describes the technical specifications, social and environmental effects, ethics, guidelines, project management strategy, risk and economic viability of the suggested Anchor-Guided Repair defence mechanism to stabilize the damaged pre-trained language models.

3.1 Final Specifications and Requirements

Hardware Requirements

- **CPU:** Intel Core i7-14700K (20 cores / 28 threads), which is utilized in data preprocessing, tokenization, and evaluation and orchestration.
- **GPU:** NVIDIA RTX 4080 Super (16 GB VRAM), which is employed in fine-tuning of models, defense training, and perplexity evaluation.
- **Memory:** 64 GB of DDR5 RAM, which was not insufficient to run model checkpoints of large size, loading datasets, or to perform an evaluation at prompt levels.
- **Storage:** 1 TB SSD, which is used to store trained models, compromised and defended checkpoints, evaluation logs, and result files.
- **Power Supply:** 850W PSU, which supports a stable performance of the graphics card and system during long training and testing sessions.

This hardware design fosters the effective experimentation in large language models, makes it possible to train stable FP16 or conduct several defense and evaluation runs without resource bottlenecks.

Software Requirements

Operating System: Windows 10/11 (64-bit) that offers a predictable development environment to Python-based machine learning workflows and CUDA acceleration via the use of a graphics card.

Python Version: Python 3.10 or 3.11 (64-bit) compatible with the rest of the deep learning ecosystem such as PyTorch 2.x and Transformers and with the newest language features utilized in the implementation.

Deep Learning Framework: PyTorch 2.0 or higher with support of CUDA 11.8 or 12.1 is the main deep learning framework used to train models, defense fine-tune models, and evaluate the perplexity of various precision modes (FP16, INT8, 4-BIT).

Machine Learning Libraries: Hugging Face Transformers 4.35+, includes the AutoModelForCausalLM, AutoTokenizer, Trainer, TrainingArguments, and DataCollatorForLanguageModeling modules needed to load pre-trained LLMs (Qwen2.5, Pythia), tokenize, and fine-tune.
Hugging Face Datasets 2.14+, used to load and process the WikiText-2 benchmark dataset to evaluate the perplexity and train the defense.
BitsAndBytes 0.41+, which allows INT8 and 4-bit quantization of large language models by the BitsAndBytesConfig integration with the Transformers library.

Scientific Computing: NumPy 1.24+, the library used to perform numeric operations, statistical computation and sigma calibration of noise injection attacks.

Data Visualization: Matplotlib 3.7+ used to generate the plots of defense effectiveness, perplexity comparison bar charts, visualizations of recovery rates, and category heatmaps to be analyzed and published.

Progress monitoring: tqdm 4.65+, offers progress bars to long-term computations and operations such as tokenization, iteration of perplexity calculations, and training cycles.

CUDA Toolkit: The NVIDIA CUDA Toolkit 11.8 or 12.1, that offers the GPU acceleration runtime to PyTorch operations, such as model inference, gradient calculation and quantized model execution.

Development Environment: Visual Studio Code which act as the integrated development environment of code editing, debugging, and terminal execution.
Git 2.40+ which is used to control and track model checkpoints, experiment settings and result files.

Package Manager: pip 23.0+ or conda 23.0+ which are used to handle Python package dependencies and build isolated virtual environments to allow reproducible experiments.

Data Requirements

WikiText-2 corpus is the main dataset on which evaluation of the models is performed which is divided into training, validation and test sets. In order to evaluate the performance of the models in a very comprehensive manner and over a great array of capabilities, a test battery of 96 prompts is utilized. These prompts will have

varied task typologies such as code generation, mathematical and logical reasoning, answering questions, including summarization, creative writing, translate text, sentiment classification, extracting information, instruction following, commonsense reasoning and domain-specific scientific knowledge. The experiment system makes use of pre trained language model like Pythia-410M and Qwen2.5-0.5B which necessitates the capability to fetch and manipulate model weights and the capacity to assess and control FP16 precision.

3.2 Societal Impact

The suggested defense can substantially advance the reliability and trustworthiness of open-source language models. It shields the users against non-intentional execution of models that have gone stray or deteriorated by default, concealing performance and integrity challenges that would otherwise have gone unnoticed. Moreover, the mechanism will make the systems of AI working in the independent area like education, research and software creation more robust. The potential usefulness in real-world applications is that it can facilitate the process of detecting and fixing model corruption, without requiring the reprocessing of models, thus incurring fewer computational and resource costs. On a larger scale, the method will encourage safer secondary use and recycling of existing models, as well as the ethical deployment of artificial intelligence, especially in resource-limited settings.

3.3 Environmental Impact

The proposed method is environmental friendly and computationally efficient. The method does not need full retraining of large language models but instead depends on extremely short schedules of fine-tuning of less than 200 training steps that lead to significant computational cost reductions. This efficiency is directly carried over into reduced use of energy when compared to full pretraining processes. Though the use of GPUs may still be one of the factors that contribute to the consumption of energy, this can be reduced by the use of single-gpu training, small batch-sizes, and short-trains. Generally, the given plan can assist in repairing energy-efficient models and, in turn, make the language model development more environmentally friendly.

3.4 Ethical Issues

Ethically, the suggested defense mechanism does not add new information to the model and is not based on personal, sensitive, or user-generated data. It only works with publicly available datasets, namely the WikiText-2 corpus, making sure that it adheres to the typical set of ethical and data governance principles. Regarding responsible use, it concerns manipulation control of the model behavior and not intrusive repair or redesign which is important to preserve the intent and bias attribute of the model. In addition, it promotes greater goodness and frankness during the assessment of the model integrity and strength, which contributes to the more responsible and credible use of language models.

3.5 Standards

Despite the lack of an industry standard that is specifically applied to language model repair, the suggested work complies with a variety of general principles and best practices. Specifically, it follows the IEEE principles regarding the software reliability and reproducibility, which implies that the experimental findings can be obtained and confirmed. The approach is also based on the best practices of safe data management and accountable artificial intelligence and focuses on transparency and responsible management of model artifacts. Besides, the work is based on the best practices of machine learning research such as controlled experiments, explicit evaluation procedure, and verifiable reporting.

3.6 Project Management Plan

The development timeline of the project is organized into four sequential phases. The first phase aims at preparing and evaluating baseline conditions of the chosen pretrained models and will be followed by simulating noisy-based attacks to determine vulnerability. The third step is to train the anchor guided defense system and then thoroughly do testing with the WikiText-2 data and the 96-prompt test package. The last step is the one devoted to the analysis of results, visualization, and reporting. Distribution of resources becomes simplified, and the researcher performs all the procedures of training, evaluation and analysis; assisted by the Computer-Generated-Improvement of experimentation and by trial and error experimentation to optimize the parameters.

3.7 Risk Management

A number of risks were subsequently noted in the process of undertaking the project, especially the ones that were connected to architectural sensitivity among the various models e.g. differences between Pythia and Qwen, and the possibility of overfitting when defending training. In order to reduce these risks, numerical checks and FP16-safe evaluation procedures are strict and used to stabilize things. There are also other safeguards like variance analysis, early level sanity checks, limited recovery constraints and the use of early stopping mechanisms as well. All these mitigation measures guarantee the continuity of scrutiny, dependability, and credibility of the presented defence measure.

3.8 Economic Analysis

Economically, the suggested solution will be affordable and affordable. Cloud services are completely avoided, and instead, a sequence of open-source software and a single consumer-grade-GPU is used. Subsequently, the only significant constraint to cost is the initial investment in costly human interface hardware in the form of GPU plus relatively low electricity cost because of brief training times. Regarding benefits, the defense system will avoid the expensive requalification of large language models, the reuse of damaged or degraded models, and save financial and comput-

ing resources. Altogether, the defense strategy, offered, is cheap and adapted to the practice.

Chapter 4

Proposed Methodology

4.1 Design Process or Methodology Overview

The following section outlines the technical architecture and algorithm of the proposed defense mechanism, including the dual-model anchoring strategy, loss functionalization, parameter selection criteria, and multi-precision evaluation protocol that can hold the performance recovery controlled under weight-noise attacks with limited and understandable recovery measures.

As outlined in Figure 4.1, the experimental pipeline proceeds through five distinct stages: starting with baseline establishment, moving to task adaptation, simulating the weight attack, training the anchor-guided defense, and concluding with a multi-metric evaluation.

4.1.1 Baseline Establishment and Initial Performance Assessment

The establishment phase begins with an initial assessment of the pretrained language model under various numerical precision settings. This helps us understand its zero-shot performance and measure how post-training quantization affects model quality. The pretrained model, either Qwen2.5-0.5B-Instruct or Pythia-410M, is loaded directly from its original checkpoint without fine-tuning or task adaptation. We measure perplexity on the WikiText-2 test split at three precision levels: FP16 (half-precision floating-point), INT8 (8-bit integer quantization), and 4-BIT (NormalFloat 4-bit quantization using the BitsAndBytes NF4 format).

This evaluation across multiple precision levels shows how sensitive pretrained weights are to numerical compression. Switching to INT8 precision usually leads to a 5 to 15 percent increase in perplexity, while moving to 4-BIT precision results in a 15 to 30 percent increase. These changes come purely from less accurate numerical representation, not from external corruption or random weight shifts. The WikiText-2 test set serves as a standard evaluation across all experimental stages. It includes a variety of Wikipedia articles from different domains and writing styles. This diversity ensures consistent measurements and allows for direct comparisons between different model states, such as clean, attacked, and defended.

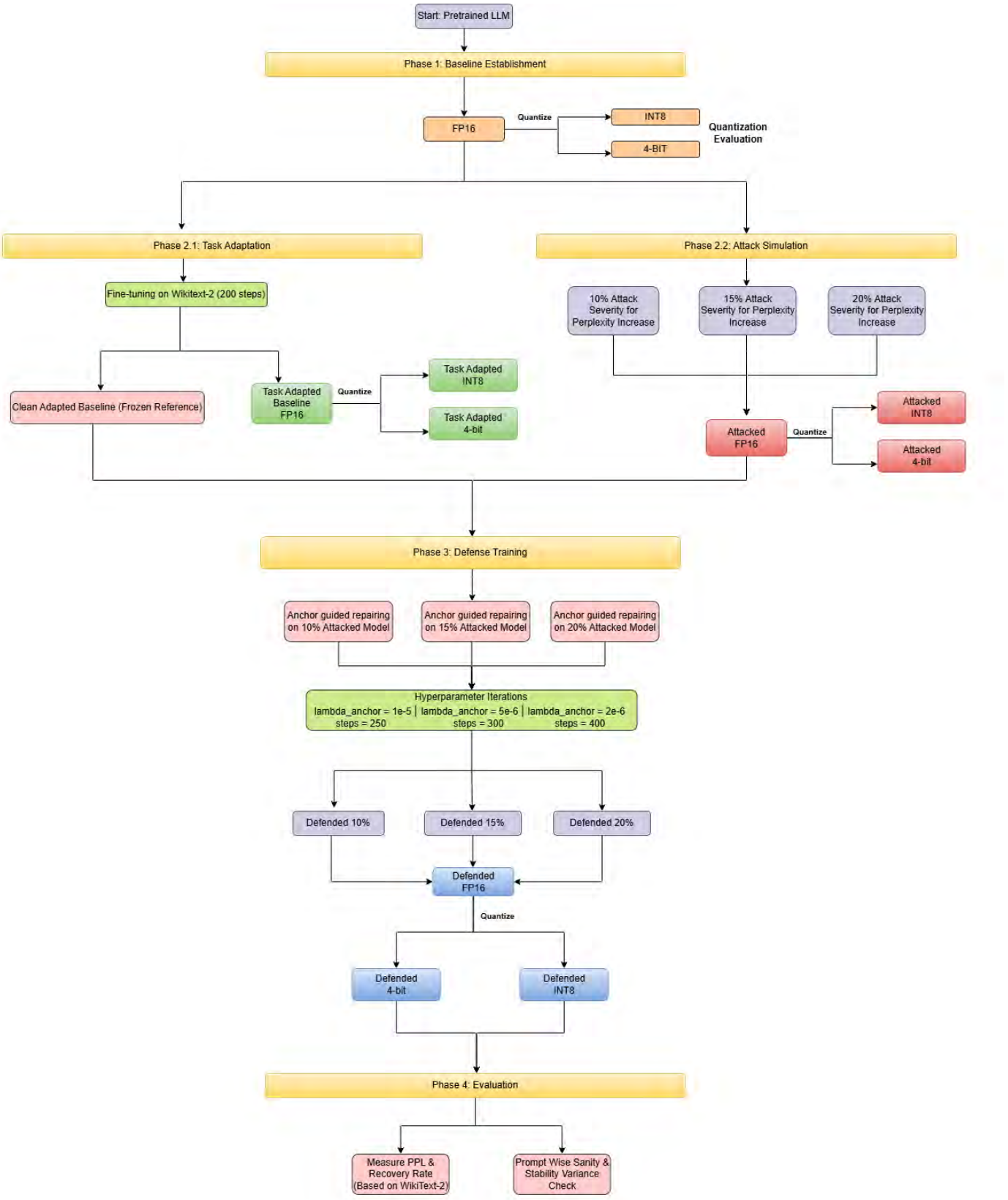


Figure 4.1: Experimental Methodology Flowchart.

These baseline measurements are crucial for further analysis. They define the performance of the unaltered pretrained model and highlight the independent effects of quantization errors before we introduce task adaptation or weight noise. While we fully record these zero-shot baseline measurements, we do not use them in recovery rate calculations. Recovery is defined in relation to the task-adapted clean baseline developed in Phase 2.1. Using zero-shot references would create methodological issues that distort interpretations of defense effectiveness and obscure the true recovery capacity of the anchor-guided training strategy.

4.1.2 Task Adaptation and Clean Baseline Establishment

To create an interpretable recovery framework, the pretrained model first undergoes supervised fine-tuning on the WikiText-2 training corpus to match the target domain. This step removes the confusing effects of zero-shot mismatch and sets up a clean task-adapted baseline. This baseline shows the best possible performance on the evaluation distribution.

Fine-tuning uses the standard causal language modeling objective without any anchoring constraint. This lets the model learn the specific statistical structure of the dataset. The optimization runs for 200 steps with AdamW, using a learning rate of 2e-5 and a gradient accumulation of 4 steps (resulting in an effective batch size of 4). The task-adapted model serves as the upper limit for comparing all attacked and defended versions. After this adaptation, the clean baseline is converted into INT8 and 4-bit formats. This conversion keeps precision effects measurable throughout the process. Importantly, the task-adapted baseline acts as the source model for the frozen anchor reference during defense training. This ensures that recovery stays bounded and clearly defined.

The resultant task-adapted clean baseline is two-fold in the pipeline of the experiment:

1. Since the basis of creating attacked models by controlled weight-noise injection is to ensure that the adversarial perturbations are done to weights that already hold task-relevant information instead of zero-shot representations.

2. As the frozen anchor reference during defense training, providing stable gradient signals that guide the optimization process toward recovering task-adapted knowledge rather than attempting to simultaneously learn task adaptation and noise removal.

Upon completion of fine-tuning, the adapted model is saved in FP16 format with all optimizer states and training checkpoints preserved, and undergoes comprehensive evaluation at FP16, INT8, and 4-BIT precision levels to establish clean baseline metrics that will serve as the upper performance bound for all subsequent recovery calculations. This methodological refinement ensures that the experimental framework produces bounded and interpretable recovery rates ($\leq$ 100%), as defended models that undergo additional fine-tuning are fundamentally constrained by the performance ceiling established by this task-adapted clean baseline rather than being compared against artificially weak zero-shot references.

4.1.3 Attack Simulation Framework

Motivation

In order to model adversarial degradation and test model robustness, a weight-noise attack is proposed in this thesis. In contrast to adversarial input perturbations, this attack mitigates the internal weights of the model directly, which is used to denote quantization-based corruption or malicious parameter manipulation.

Attack Formulation

Let W be a trainable parameter tensor. The attacked parameter tensor W' is defined as:

$$W' = W + \epsilon$$

where ϵ is sampled element-wise from a Gaussian distribution with mean 0 and variance σ^2:

$$\epsilon \sim \mathcal{N}(0, \sigma^2 I)$$

The parameter σ controls the strength of the attack.

Measuring Attack Severity via Perplexity Increase

Let PPL_0 be the perplexity of the clean baseline model and $PPL(\sigma)$ be the perplexity after applying perplexity-calibrated noise with standard deviation σ. Define the degradation ratio as:

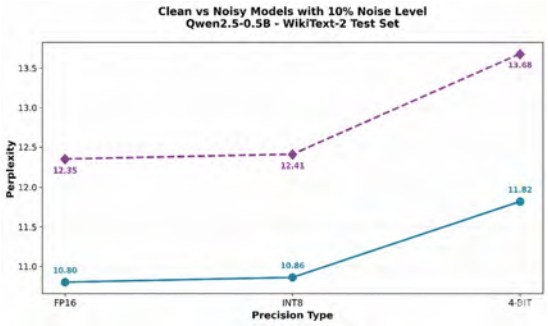
$$r(\sigma) = \frac{PPL(\sigma)}{PPL_0}$$

Attack severity levels are defined by target ratios:

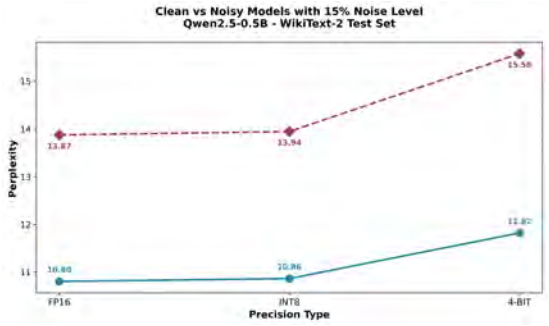
- 10% attack: $r = 1.10$
- 15% attack: $r = 1.15$
- 20% attack: $r = 1.20$

The graphs in Figure 4.2 demonstrate the widening performance gap between the Clean Baseline (solid line) and the Attacked Model (dashed line) as perplexity-calibrated noise intensity increases. Notably, the degradation is most severe in the 4-BIT quantization regime at 20% perplexity-calibrated noise. On the other hand, in Figure 4.3, Pythia-410M exhibits high sensitivity to weight noise. The sharp ascending bell curve of the perplexity of the attacked model. in the 4-BIT scenario implies that low-precision quantization greatly increases. the harm due to the noise injection.

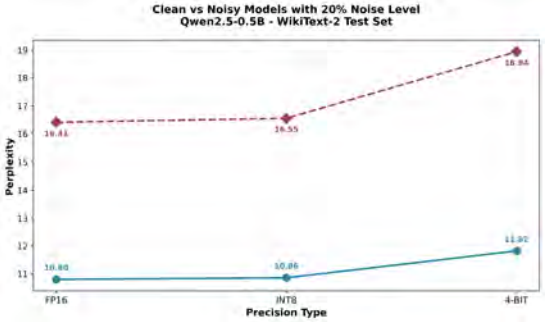
After creating the attacked model in FP16 precision, the corrupted weights are converted into lower-precision formats to assess robustness under realistic deployment constraints. Each attacked checkpoint corresponds to a calibrated severity level (10%, 15%, or 20% increase in perplexity) and is quantized into both INT8 and 4-BIT NF4 formats. This generates a complete set of attacked models across FP16, INT8, and 4-BIT, which enables further defense training and evaluation. This measures how adversarial weight corruption interacts with aggressive post-training



10% Perplexity-Calibrated Noise Level



15% Perplexity-Calibrated Noise Level



20% Perplexity-Calibrated Noise Level

Figure 4.2: Impact of Gaussian Weight Noise on Perplexity Degradation across Varying Perplexity-Calibrated Noise Intensities (10%, 15%, and 20%) for Qwen2.5-0.5B.

quantization. Following the overall process in Figure 4.1, these attacked quantized versions are direct inputs for the anchor-guided repair phase.

Calibrating Sigma to Achieve Target Attack Levels

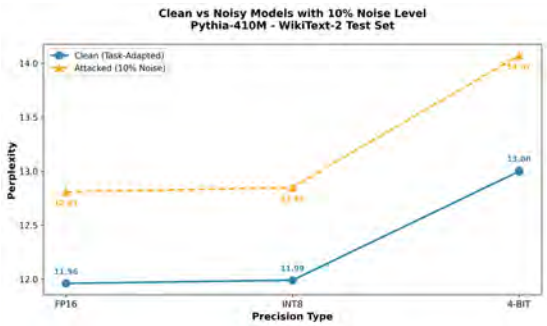
Experimental data show that the dependence of the noise variance on the response of the model takes an approximately quadratic pattern when plotted in a logarithmic scale which we can see it in Equation 4.1. Based on this, the function $r(\sigma)$is to be modeled as a second-order polynomial in σ with the coefficients a and b being estimated by regression on the validation data.

$$\log\big(r(\sigma)\big) \approx a\,\sigma^2 + b \tag{4.1}$$

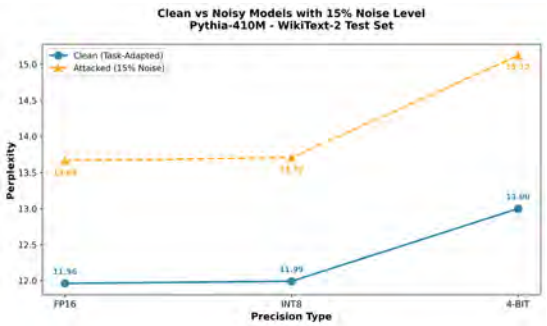
The response function can be directly expressed in exponential form by exponentiating the fitted logarithmic expression like in Equation 4.2. This representation preserves the observed nonlinear behaviour and is useful for estimating the response at various perplexity-calibrated noise levels.This can be roughly expressed as:

$$r(\sigma) \approx \exp\big(a\,\sigma^2 + b\big) \tag{4.2}$$

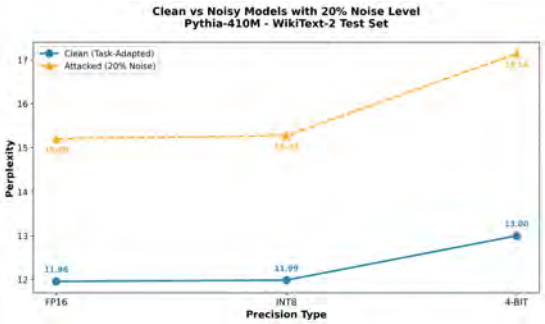
By reversing the fitted response model, the corresponding perplexity-calibrated noise level σ^* can be calculated given a predetermined target response, r_{target} which can



10% Perplexity-Calibrated Noise Level



15% Perplexity-Calibrated Noise Level



20% Perplexity-Calibrated Noise Level

Figure 4.3: Impact of Gaussian Weight Noise on Perplexity Degradation across Varying Perplexity-Calibrated Noise Intensities (10%, 15%, and 20%) for Pythia-410M.

be seen in Equation 4.3. This makes it possible to directly control the amount of perplexity-calibrated noise needed to produce the desired system behaviour.

$$\sigma^* = \sqrt{\frac{\log(r_{\text{target}})}{a}} \tag{4.3}$$

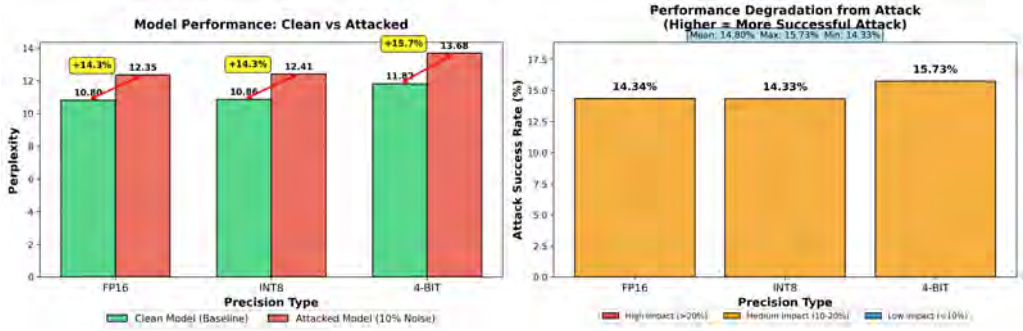
The coefficient a is obtained via linear regression over multiple measured pairs of $(\sigma^2, \log(r))$. This calibration ensures that attack severity is comparable across different models and precision formats.

4.1.4 Attack Success Rate Analysis

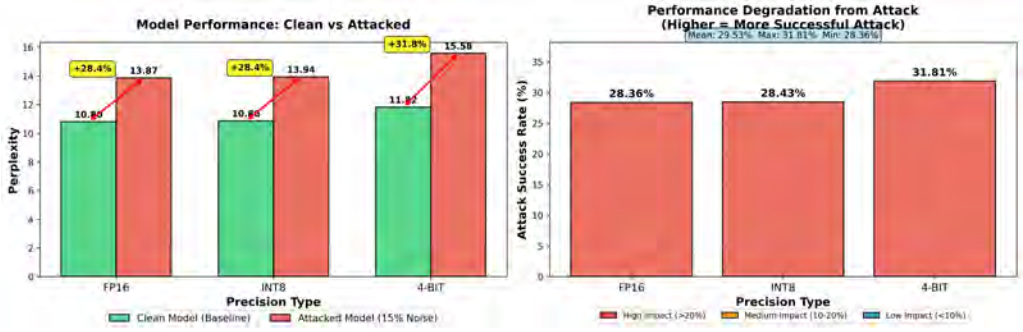
Figure 4.4 shows the efficiency of weight-level noise attacks in the Qwen 2.5-0.5B model in three calibrated attack severity. In each precision format, the attacked model also shows a consistent, as opposed to its clean baseline, perplexity increase.

At a 10%, the degradation is still moderate (around 14-16%) meaning that there is controlled but successful perturbation. With the perplexity-calibrated noise strength rising up to 15 percent and 20 percent the success rate of the attacks increases dramatically reaching above 50 percent at the maximum.

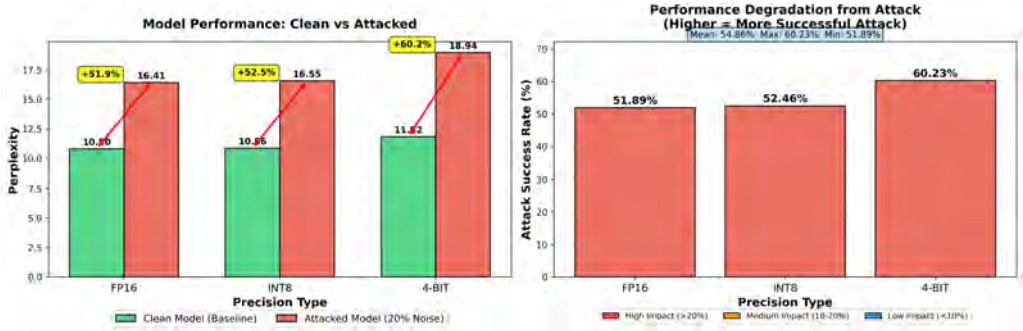
In all cases, it is the lowest precision configurations, especially 4-bit NF4 that exhibit the greatest degradation and are more sensitive to noise in parameters. These



10% Perplexity-Calibrated Noise Level



15% Perplexity-Calibrated Noise Level



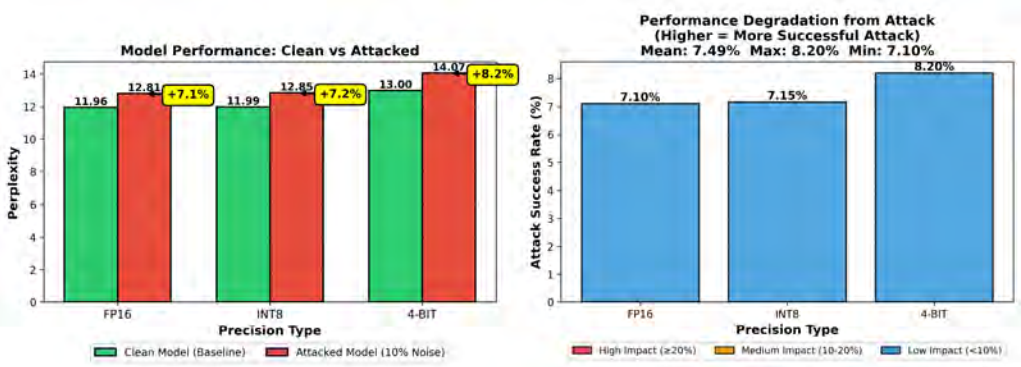
20% Perplexity-Calibrated Noise Level

Figure 4.4: Analysis of attack success rate of Qwen2.5-0.5B during calibrated injection of weight noise of 10, 15 and 20 percent. All the plots demonstrate the comparison of the clean and attacked model perplexity in the FP16, INT8, and 4-bit precision modes. Relative perplexity increase is reported in the right-hand panels and it is the attack success rate.

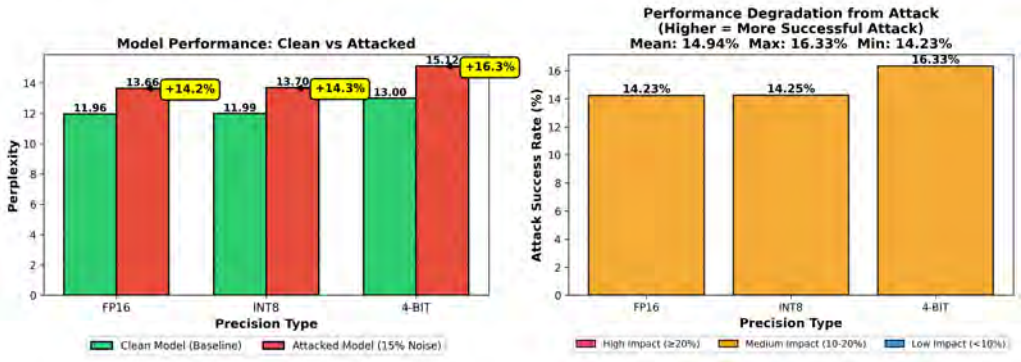
findings support the claim that weight-level attacks are effective at all precision formats and these attacks are stronger with the increase of perplexity-calibrated noise size.

Figure 4.5 illustrates similar findings with respect to the Pythia-410M model. There are similar trends and the perplexity is increasing monotonically with the increase in perplexity-calibrated noise level. Pythia has a lower absolute degradation at 10% than Qwen2.5-0.5B which means that it is less sensitive to small perturbations.

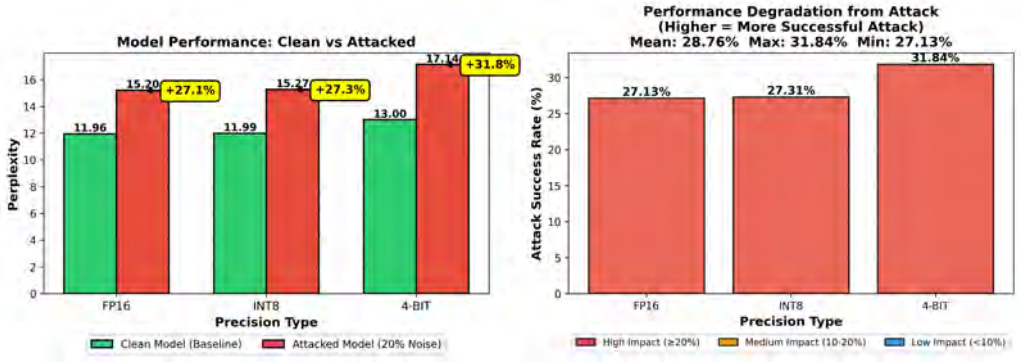
Nonetheless, it can be seen that the probability of successful attack is much higher



10% Perplexity-Calibrated Noise Level



15% Perplexity-Calibrated Noise Level



20% Perplexity-Calibrated Noise Level

Figure 4.5: Success rate of attacks on Pythia-410M with injected noise of calibrated weight at 10 percent, 15 percent and 20 percent. The test is performed on the FP16, INT8, and 4-bit precision format and perplexity is used as the main test metric.

at 15 percent and 20 percent damage levels, especially at low-precision conditions. The regularity in ordering between FP16, INT8 and 4-bit precision shows that the behavior of the attack is model-agnostic which is largely controlled by the magnitude of weight perturbation.

4.1.5 Defense Training Through Anchored Fine-Tuning

The defense training phase also applies the dual model architecture that is optimized for the task-adapted clean base-memory efficiency and computational stability. The task-adapted clean base-execution speed is determined by the computation being done. The baseline model is used as a frozen anchor reference that is stored in the memory of the CPU, while a replica of the model that was attacked is loaded as a trainable instance on the GPU. memory. The anchoring mechanism selectively targets 168 critical weight matrices distributed across the model's 24 transformer layers, specifically including the attention projection matrices (q_proj, k_proj, v_proj, o_proj) and MLP feed-forward projection matrices (gate_proj, up_proj, down_proj), while deliberately excluding normalization layers, bias parameters, and embedding matrices that primarily serve structural rather than semantic functions.

During each training iteration, the system processes a 2048-token chunk from the WikiText-2 training dataset through a four-step computational pipeline:

1. Forward pass through the trainable model to generate next-token predictions and compute standard cross-entropy loss (L_{CE}).
2. Parameter-wise L2 distance calculation between current trainable weights and their corresponding frozen anchor weights, with computation performed on CPU to conserve GPU memory.
3. Aggregation and averaging of L2 penalties across all 168 anchored parameters to produce the anchor loss (L_{anchor}).
4. Construction of the combined objective function $L_{total} = L_{CE} + \lambda \times L_{anchor}$ where $\lambda = 1e - 5$ controls the regularization strength.

The optimization process executes 200 training steps using the AdamW optimizer with learning rate 2e-5, gradient accumulation over 2 steps (effective batch size of 2), and linear warmup over 10 steps, updating all 494 million parameters in half precision (FP16) while the anchoring constraint guides weight updates toward a balanced state that minimizes prediction error on clean data while preventing excessive deviation from the clean baseline. Upon completion of the training cycle, the defended model is saved in FP16 format, preserving the recovered weight configuration for subsequent quantization and evaluation.

This anchored repair procedure is run separately for each attack severity level and yields three restored checkpoints for the Defended 10%, Defended 15%, and Defended 20% model variants.After the defense training step, the recovered models are merged and saved in FP16 precision as the final defended model state.This is consistent with the multi-precision deployment pipeline shown in Figure 4.After applying the defense, the FP16 checkpoint was quantized into INT8 and 4-bit NF4, producing Defended INT8 and Defended 4-bit models that were used in the final robustness evaluation under low-precision inference conditions.

4.1.6 Multi-Precision Evaluation and Performance Analysis

The evaluation stage is based on a multi-precision testing program that evaluates the effectiveness of the defense against three quantization configurations under various

deployment conditions and resource limitations. The defended model is sequentially evaluated with FP16 precision (half-precision floating-point, model size of about 988 MB), INT8 precision (8-bit integer quantization via per-tensor symmetric scaling, model size of about 494 MB), and 4-BIT precision (NormalFloat 4-bit quantization using BitsAndBytes NF4 format, model size of about 247 MB), and the measurements of perplexity are done on the WikiText-2 test split using the same evaluation protocols across all three model states.

For each precision level, the recovery rate is computed using the formula:

$$\text{Recovery} = \frac{PPL_{attacked} - PPL_{defended}}{PPL_{attacked} - PPL_{clean\_adapted}} \times 100\%$$

This gives a normalized measure, which measures the percentage of attack-related degradation that the defense mechanism is able to remedy. The critical ordering constraints are checked and proved to be true such that $PPL_{clean\_adapted} < PPL_{defended} < PPL_{attacked}$ on all the precision settings and the fact that the defended model recovers meaningfully without necessarily achieving more than the task-adapted clean baseline and the meaning by which the recovery is achieved is limited ($\leq 100\%$).

In addition to aggregate measures of perplexity, the analysis also uses prompt-wise analysis through a curated set of 96 prompts across several different task classes, which allows checking in both the ordering constraints the individual sample level and measuring the variance reduction to understand which improvements are being made by the attacked model. The percentage of degradation of both the attacked and defended models against the clean adapted one at each level of precision offer finer details about how weight noise and quantization errors interact or add, and show that the defense process does not lose its effectiveness when models are aggressively pruned on the edges to deploy them to real-world environments. The combination of these extensive evaluation measures, as a whole, provides the practical validity of the defense mechanism by showing reproducible partial recovery rates of 60-80% and comes with interpretable performance limits and predictable behavior under a wide variety of quantization schemes.

4.2 Preliminary Design or Design (Model) Specification

The preliminary design specification establishes the technical architecture of the defense mechanism, defining the dual-model framework, mathematical loss formulation, selective parameter targeting strategy, and multi-precision evaluation protocol that collectively enable controlled performance recovery from weight-noise attacks while maintaining bounded recovery rates.

4.2.1 Theoretical Framework: Quantization and Perplexity

Quantization as a Low-Precision Attack and Its Impact on Model Quality

Quantization as a Low-Precision Attack and Its Impact on Model Quality Quantization is a model compression method that uses fewer bits of memory on a GPU or CPU to compress the neural network parameters and run faster. Rather than storing weights in floating point representations like FP32 or FP16, quantized inference stores weights in lower-precision representations like INT8 or 4-bit. Within the framework of this thesis, quantization is considered a low-precision perturbation, which can serve as an attacker instance when not implemented with resistance to it. Quantization increases the deployability and efficiency, but the approximation error is introduced to the weight tensors, which can impair the accuracy and stability of the model.

Quantization as a Weight Approximation

Let w be an original floating-point weight value. Quantization maps w to a discrete integer value q using a scaling factor $s > 0$ as follows:

$$q = \text{clip}(\text{round}(w/s), q_{min}, q_{max})$$

The quantized approximation of the weight in real-valued form is then reconstructed as:

$$\hat{w} = s \times q$$

The quantization error is defined as:

$$\delta_w = \hat{w} - w$$

This inaccuracy disturbs the calculations of linear projections and attention layers as well as feed-forward layers. Because transformer models use numerous such operations repeatedly on many layers, minor perturbations can propagate and change the probability distribution of the model output, eventually lowering the quality of language modeling.

Quantization Configurations Used in This Work

The attacks of low precision are implemented in this thesis by quantizing the weights into low-precision numbers, applied in the Hugging Face Transformers framework by the BitsAndBytes backend at three precision levels:

- **FP16 (baseline):** Weights of the model loaded in half precision (float16), are a powerful baseline that can be used to evaluate the performance of a GPU.
- **INT8:** 8-bit quantized weights of a moderate low-precision configuration.
- **4-bit (NF4):** 4-bit NormalFloat (NF4) quantization with a FP16 computation and optional double-quantization metadata compression, which is an extreme low-precision case.

These settings are applied as controlled low-precision perturbations to examine the impact of numerically diminished precision on perplexity, robustness, and recovery in the offered defense mechanism.

Why Quantization Increases Perplexity
Quantization modifies the original weight matrix W into a perturbed version:

$$\hat{W} = W + \Delta W.$$

This is a perturbation of the logits generated by the model. As the next token prediction probability is influenced by these logits using the softmax function even tiny weight error can decrease the probability of the correct token. This increases the negative log-likelihood correspondingly causing greater perplexity.

Perplexity: Definition, Interpretation, and Computation

Definition
A standard evaluation metric of language models is the perplexity (PPL), which is a measure of the ability of the model to predict a sequence of tokens. The less the perplexity, the better the predictive performance. Given a token sequence $x_1, x_2, ..., x_N$, a causal language model factorizes the joint probability as:

$$p(x_{1:N}) = \prod_{t=1}^{N} p(x_t \mid x_{<t})$$

The average negative log-likelihood (cross-entropy loss) is:

$$\mathcal{L} = -\frac{1}{N} \sum_{t=1}^{N} \log p(x_t \mid x_{<t})$$

Perplexity is defined as:

$$\text{PPL} = \exp(\mathcal{L}) = \exp\left(-\frac{1}{N} \sum_{t=1}^{N} \log p(x_t \mid x_{<t})\right)$$

Interpretation
The perplexity may be seen as the effective number of possible choices that the model takes into account when making predictions of each token. As an example, a perplexity k model acts like a model that picks k, on average, equally likely tokens. Since quantization/attack reduces the model quality, the probability mass no longer takes values on the correct tokens, becoming more uncertain and hence adding more to the perplexity.

Computation Used in This Thesis
Transformer models have a limited context length, which means that long sequences cannot be reconstructed in one forward pass. Hence, the sliding-window assessment approach is applied in this thesis using a Maximum window length of 2048 tokens and a Stride of 512 tokens.

A piece of the text is processed by each window. The loss is only calculated on the new tokens, and the previous ones are covered by the mask of changing the labels to -100 to make them not count on the loss. Where l_i is the cumulated negative

log-likelihood of window i and N is the cumulative number of tokens considered, perplexity is computed as:

$$\text{PPL} = \exp\left(\frac{\sum_i \ell_i}{N}\right)$$

This approach avoids double-counting tokens and produces a stable perplexity estimate.

4.2.2 Data Sources and Evaluation Inputs

WikiText-2 Dataset

WikiText-2 is a set of textual data that is used as the main source of model adaptation, defense training, and aggregate evaluation. WikiText-2 is an assortment of outstanding English articles from Wikipedia that frequently serve as a benchmark when assessing language modelling ability. The dataset is split into predetermined training, validation, and test splits, which are the same in this study. Each data sample has one text field that has long-form natural language data and does not have explicit labels, which qualifies it to be used in causal language modeling. Text samples associated with each split are divided into tokens in preprocessing with the tokenizer belonging to the pretrained base model. Any tokenized text in a split is bunched together in a linear sequence and divided into deterministic sized blocks of 2048 tokens, which is the maximum length of context in the model. The language-modeling instances are each a block, and the task will be to predict the next token based on all the previous ones. Any number of additional tokens that do not constitute a complete block is ignored, as it is normal in language modeling.

Prompt Suite for Coordinated Assessments

While WikiText-2 offers a powerful reference point in terms of assessing overall language modeling performance, it does not enable the investigation of the model behavior on a case-by-case basis. In order to overcome this shortcoming, prompt-level robustness analysis is performed on a set of 96 pre-selected evaluation prompts. This prompt structure consists of short instruction-based inputs which are supposed to prompt certain behaviors of the model. 12 diverse activity categories are covered by the prompts, including code production, mathematical reasoning, logical question answers, summarisation, creative writing, and explanation activities. A particular kind of controlled examination with a clear purposeful performance that enables consistent and interpretable testing is this prompt suit.

Construction of Per-Prompt Perplexity Data

Perplexity is calculated separately for both the clean model and attacked and defended models on each prompt of the evaluation suite. For each model's setup and perplexity-calibrated noise level, this generates a per-prompt perplexity matrix of length 96. The statistical evaluation at the prompt level focuses on the per-prompt perplexity vectors. They do not only maintain information on the performance over individual input as compared to a single aggregate perplexity value, but they can identify prompt-specific failures and recovery patterns.

Sanity-Check Data for Robustness and Stability Analysis

In this research, two alternative sanity-check datasets were extracted from the per-prompt PPL vectors in order to verify robustness at the prompt level:

- **Prompt-wise ordering:** To note whether the whole expected hierarchy (clean ¡ defended ¡ attacked) obtains and whether or not the defended model outperforms the attacked model for each question. Using explicit counts across $N = 96$ prompts, this validates prompt-level consistency assertions.
- **Variance and stability:** Prompt-to-prompt instability may be caused by quantization noise. We calculate mean and standard deviation of prompt PPL values and measure the decrease in variance with defense (i.e., decrease in standard deviation compared to attacked). This measures how the defense is able to restore predictable behavior as opposed to merely enhancing average performance.

4.2.3 Implementation Details of Selected Design

The defense implementation employs a dual-model architecture optimized for memory efficiency and computational stability. The task-adapted clean baseline model (494 million parameters, Qwen2.5-0.5B-Instruct or Pythia-410M) is loaded as a frozen anchor reference on CPU memory with all parameters set to non-trainable (requires_grad=False), while the attacked model is loaded as a trainable copy on GPU memory. The WikiText-2 training dataset undergoes preprocessing where all text samples are concatenated using double-newline separators, resulting in approximately 2.1 million tokens for the training split. The concatenated text is tokenized using the model-specific tokenizer (vocabulary size 151,936 for Qwen2.5) and subsequently segmented into non-overlapping chunks of 2048 tokens each, yielding 1,026 training blocks. Each chunk is structured as a dictionary containing input_ids and labels arrays (both identical for causal language modeling), which are then converted into a HuggingFace Dataset object for efficient batching during training.

Figure 4.6 illustrates the dual-model training architecture. The "Frozen Anchor" (Clean Model) remains static on the CPU to calculate the regularization loss, while the "Trainable Model" (Attacked) on the GPU is updated to minimize both the cross-entropy loss and the anchor distance.

The anchoring mechanism selectively targets critical weight matrices across the model's 24 transformer layers while excluding parameters unlikely to contain task-relevant information. Specifically, 168 weight matrices are identified for anchoring: 96 attention projection matrices (q_proj, k_proj, v_proj, o_proj across all layers) and 72 MLP feed-forward projection matrices (gate_proj, up_proj, down_proj across all layers). Parameters excluded from anchoring include all layer normalization weights, bias terms, embedding matrices (input and output), and positional encodings, as these components either serve normalization functions or encode positional information rather than semantic knowledge.

During each training step, the forward pass processes a single 2048-token chunk through the trainable model to compute standard cross-entropy loss (L_{CE}) for next-

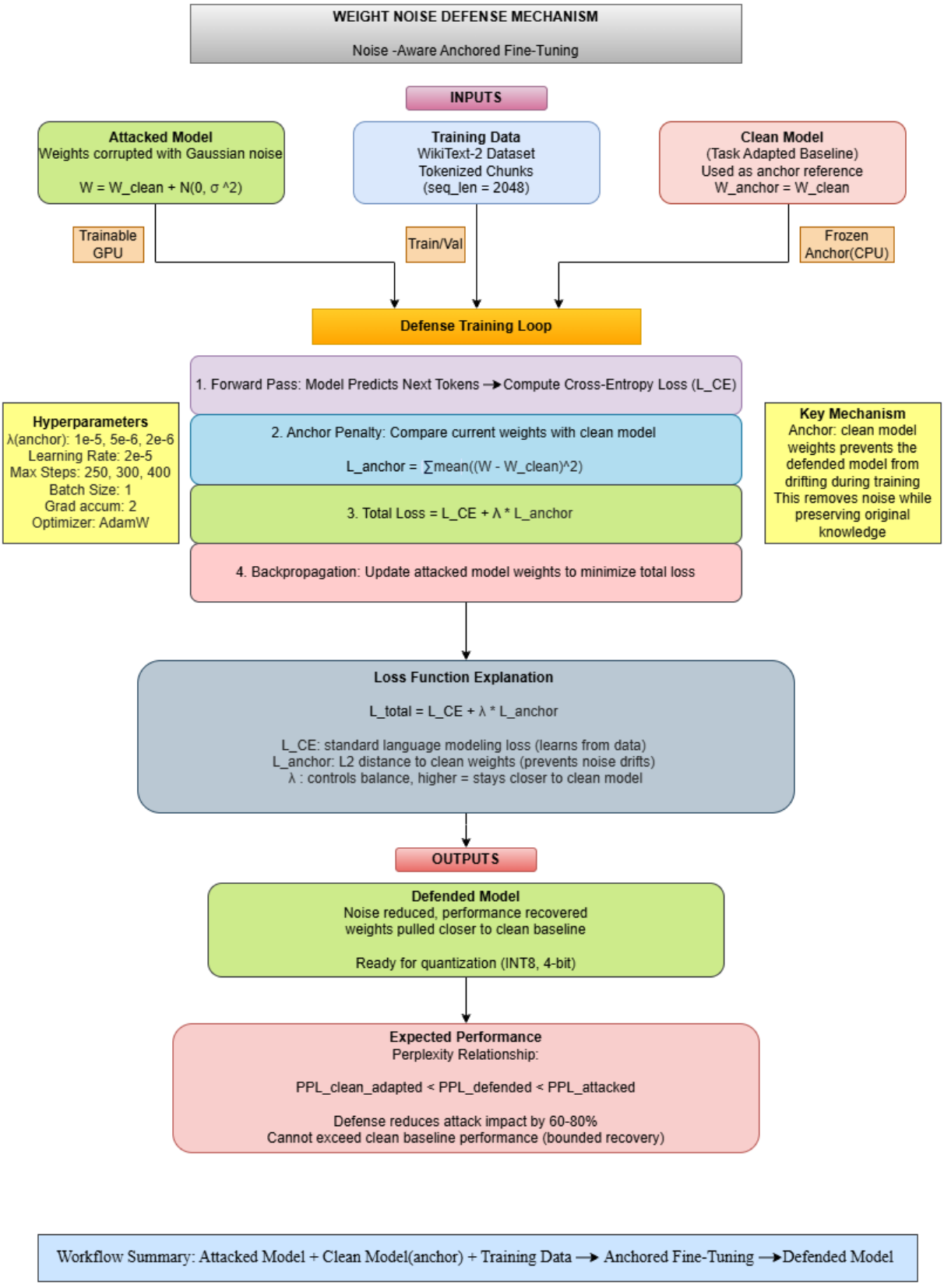


Figure 4.6: Operational Workflow of the Anchor-Guided Repair Mechanism.

token prediction. Subsequently, for each of the 168 anchored parameters, the current

weight matrix is detached and moved to CPU memory to compute the L2 squared distance against its corresponding frozen anchor weight:

$$L2_{penalty} = \text{mean}((W_{current} - W_{anchor})^2)$$

These individual penalties are accumulated and averaged across all anchored parameters to produce L_{anchor}, which is then combined with the cross-entropy loss using the regularization coefficient $\lambda = 1e - 5$, yielding the total loss:

$$L_{total} = L_{CE} + \lambda \times L_{anchor}$$

Parameter optimization is conducted over 200 training steps using the AdamW optimizer with a learning rate of 2e-5, gradient accumulation over 2 steps (effective batch size of 2), and a linear warmup schedule over the first 10 steps. All parameter updates are done with half precision (FP16) to make sure that they are stable in terms of numbers when computing gradients and weight updates, and the model takes approximately 9-10 GB of GPU memory (gradients, optimizer states, and activations). This training program combines about 250 chunks or half a million tokens, representing 24% of the WikiText-2 training data, at each optimization step, updating each of the 494 million parameters with the anchoring constraint specified to the 168 targeted weight matrices. In the evaluation phase, post-training quantization is applied on the defended model, which is tested in the case of. three precision modes: FP16, INT8, and 4-BIT. This quantization strategy facilitates the evaluation of defense strength in deployment with limited resources. scenarios and at the same time sustaining training stability with full-precision optimization.

Chapter 5

Result Analysis

5.1 Performance Evaluation

This section evaluates the proposed Anchor-Guided Repair mechanism using two complementary evaluation tracks:

- Corpus-level evaluation on WikiText-2 using perplexity (PPL) under three precision settings (FP16, INT8, 4-bit NF4).
- Prompt-level evaluation using a fixed 96-prompt suite to test whether improvements are consistent per prompt rather than only on averages.

5.1.1 Evaluation Setup

All defended models were generated using task-adapted baseline and were evaluated under identical decoding and perplexity computation settings for fairness. For each model family (Qwen2.5-0.5B and Pythia-410M), we report:

- **Clean PPL**: perplexity of the task-adapted baseline model
- **Attacked PPL**: perplexity after Gaussian weight perturbation calibrated to a target perplexity-calibrated noise level
- **Defended PPL**: perplexity after anchor-guided fine-tuning
- **Recovery(%)**: fraction of the attack degradation recovered using the standard recovery metric

A key validity condition is the expected ordering:

$$\text{Clean PPL} < \text{Defended PPL} < \text{Attacked PPL}$$

which ensures the defense does not "over-recover" beyond the clean baseline.

5.1.2 Results

Qwen2.5-0.5B

Across Qwen experiments, the defense effectiveness increased as noise severity increased. In Table 5.1, 5.2, 5.3 performance of Qwen2.5-0.5B across all perplexity-calibrated noise level in 3 attempts is shown. But in Figures we've shown only

the 20% perplexity-calibrated noise level ones since those are the best results so far. Figure 5.1 depicts the results from the first defense configuration ($\lambda = 1e-5$, steps=250). While it successfully recovers over 64% of the performance lost to the attack across all precisions, the gap between the Defended (blue line) and Clean (green line) models remains noticeable, particularly in the 4-BIT regime. In Figure 5.2, reducing the regularization strength to $\lambda = 5e-6$ and increasing training to 300 steps yields a visible improvement. The Defended model's perplexity (blue line) shifts closer to the Clean baseline, and recovery rates consistently exceed 67%, demonstrating that a less rigid anchor allows for better adaptation. Figure 5.3 shows the optimal configuration ($\lambda = 2e-6$, steps=400). This setting achieves the highest recovery rates (up to 75% for 4-BIT), with the Defended model's performance tightly shadowing the Clean baseline. The narrowing gap indicates that the defense has effectively stabilized the model against the 20% perplexity-calibrated noise attack.

Attempt 1 (A1) Results:($\lambda = 1e-5$, steps = 250)

Table 5.1: Qwen2.5-0.5B Performance Results (Attempt 1)

Precision	Clean PPL	Attacked PPL	Defended PPL	Recovery %
	10% Perplexity-Calibrated Noise Level			
FP16	10.80	12.35	11.76	38.6%
INT8	10.86	12.41	11.81	38.5%
4-BIT	11.82	13.68	12.87	43.4%
	15% Perplexity-Calibrated Noise Level			
FP16	10.80	13.87	12.31	51.0%
INT8	10.86	13.94	12.35	51.6%
4-BIT	11.82	15.58	13.43	57.1%
	20% Perplexity-Calibrated Noise Level			
FP16	10.80	16.41	12.81	64.2%
INT8	10.86	16.55	12.87	64.6%
4-BIT	11.82	18.94	14.15	67.3%

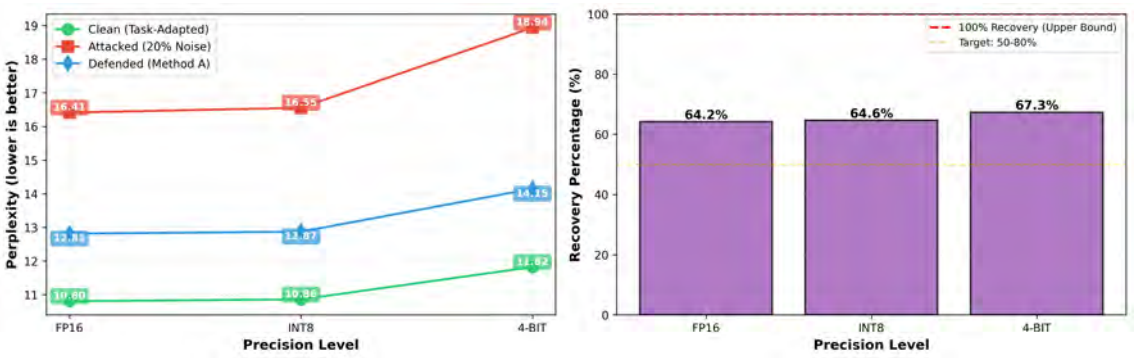


Figure 5.1: 1st attempt results for Qwen2.5-0.5B(20% perplexity-calibrated noise)

Attempt 2 (A2) Results ($\lambda = 5e-6$, steps = 300)

Table 5.2: Qwen2.5-0.5B Performance Results (Attempt 2)

Precision	Clean PPL	Attacked PPL	Defended PPL	Recovery %
10% Perplexity-Calibrated Noise Level				
FP16	10.80	12.35	11.76	38.6%
INT8	10.86	12.41	11.81	38.5%
4-BIT	11.82	13.68	12.87	43.4%
15% Perplexity-Calibrated Noise Level				
FP16	10.80	13.87	12.12	57.0%
INT8	10.86	13.94	12.17	57.4%
4-BIT	11.82	15.58	13.23	62.5%
20% Perplexity-Calibrated Noise Level				
FP16	10.80	16.41	12.61	67.7%
INT8	10.86	16.55	12.67	68.2%
4-BIT	11.82	18.94	13.89	70.9%

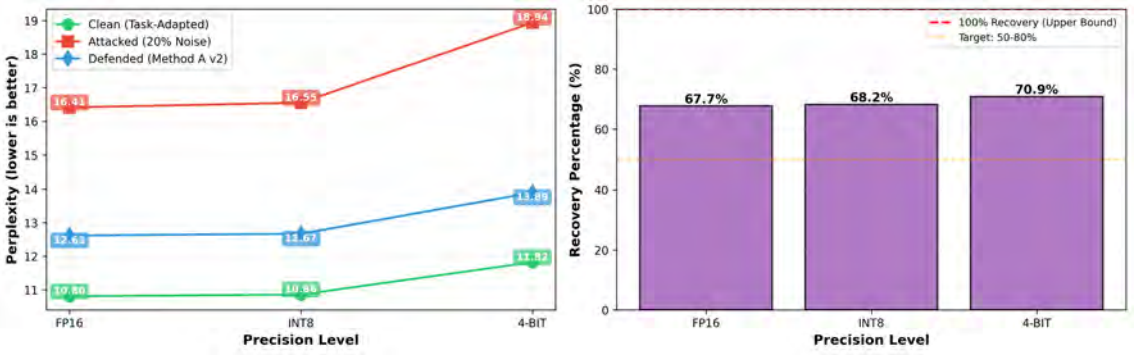


Figure 5.2: 2nd attempt results for Qwen2.5-0.5B(20% perplexity-calibrated noise)

Attempt 3 (A3) Results (Best Configuration: $\lambda = 2e-6$, steps = 400)

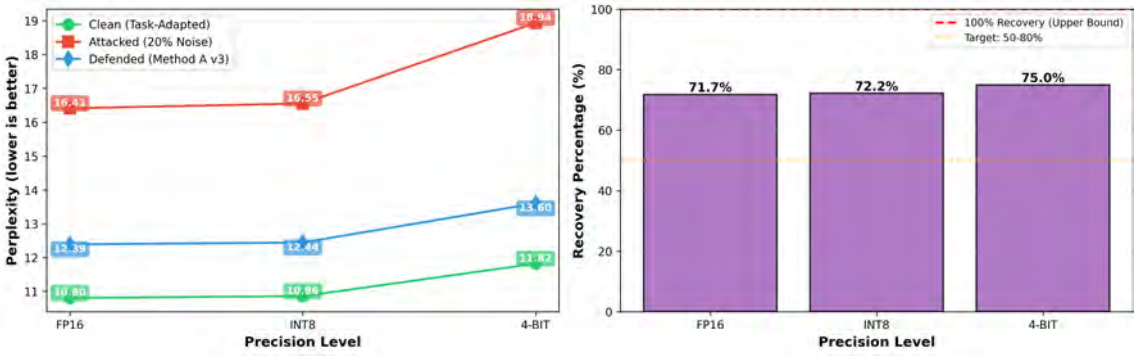


Figure 5.3: 3rd attempt results for Qwen2.5-0.5B(20% perplexity-calibrated noise)

Table 5.3: Qwen2.5-0.5B Performance Results (Attempt 3)

Precision	Clean PPL	Attacked PPL	Defended PPL	Recovery %
10% Perplexity-Calibrated Noise Level				
FP16	10.80	12.35	11.58	49.8%
INT8	10.86	12.41	11.64	49.9%
4-BIT	11.82	13.68	12.66	55.0%
15% Perplexity-Calibrated Noise Level				
FP16	10.80	13.87	11.92	63.5%
INT8	10.86	13.94	11.97	64.0%
4-BIT	11.82	15.58	12.99	68.8%
20% Perplexity-Calibrated Noise Level				
FP16	10.80	16.41	12.39	71.7%
INT8	10.86	16.55	12.44	72.2%
4-BIT	11.82	18.94	13.60	75.0%

Overall, Qwen demonstrates that the defense mechanism is highly effective under moderate-to-strong corruption (10%–20%) and becomes stable even at low perplexity-calibrated noise after sufficient hyperparameter tuning.

Pythia-410M

For Pythia-410M, all WikiText-2 results satisfied the expected ordering constraint (Clean ¡ Defended ¡ Attacked). Recovery increased with perplexity-calibrated noise level and improved further through hyperparameter refinement. In Table 5.4, 5.5, 5.6 performance of Pythia-410M across all perplexity-calibrated noise level in 3 attempts is shown. But in Figures we've shown only the 20% perplexity-calibrated noise level ones since those are the best results so far. Figure 5.4 illustrates the initial defense configuration ($\lambda = 1e-5$, steps=250) for Pythia-410M. Unlike Qwen, Pythia shows a struggle to recover at lower perplexity-calibrated noise levels, though it achieves decent recovery ($\approx 60-69\%$) at 20% perplexity-calibrated noise. The significant gap between the Clean (green) and Defended (blue) lines suggests the anchoring was too restrictive for this architecture. In Figure 5.5, relaxing the anchor to $\lambda = 5e-6$ and extending training to 300 steps results in a uniform improvement. The recovery rates push past 63% across precisions, and the Defended model's perplexity curve begins to flatten, indicating better stability against the noise injection compared to the first attempt. Figure 5.6 displays the optimal configuration ($\lambda = 2e-6$, steps=400). This environment brings the maximum power out of Pythia that can reach almost 79 percent recovery at 4-BIT regime. It turns out that the Defended model has been closely following the Clean baseline since the relationship between a smaller regularization penalty was the key to stabilizing this particular architecture.

Table 5.4: Pythia-410M Performance Results (Attempt 1)

Precision	Clean PPL	Attacked PPL	Defended PPL	Recovery %
10% Perplexity-Calibrated Noise Level				
FP16	11.96	12.81	12.75	6.5%
INT8	11.99	12.85	12.79	7.2%
4-BIT	13.00	14.07	13.67	37.4%
15% Perplexity-Calibrated Noise Level				
FP16	11.96	13.66	12.98	39.8%
INT8	11.99	13.70	13.01	40.1%
4-BIT	13.00	15.12	13.96	55.0%
20% Perplexity-Calibrated Noise Level				
FP16	11.96	15.20	13.27	59.5%
INT8	11.99	15.27	13.31	59.8%
4-BIT	13.00	17.14	14.30	68.7%

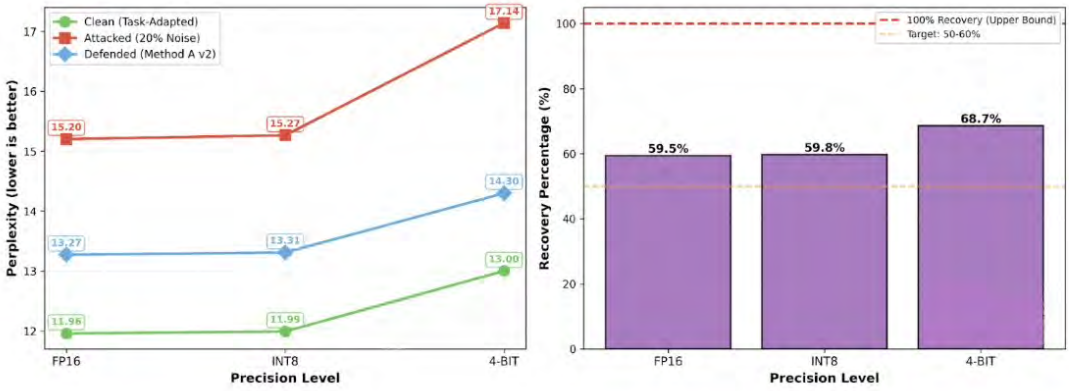


Figure 5.4: 1st attempt results for Pythia-410M(20% perplexity-calibrated noise)

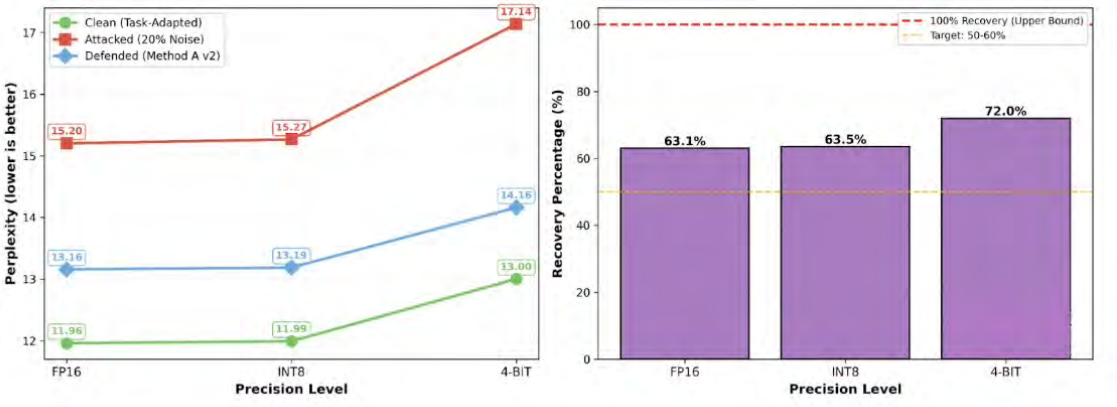


Figure 5.5: 2nd attempt results for Pythia-410M(20% perplexity-calibrated noise)

Attempt 1 (A1) Results ($\lambda = 1e-5$, steps = 250)

Attempt 2 (A2) Results ($\lambda = 5e-6$, steps = 300)

Attempt 3 (A3) Results (Best Configuration: $\lambda = 2e-6$, steps = 400)

Summary of Attempt 3 gains:

Table 5.5: Pythia-410M Performance Results (Attempt 2)

Precision	Clean PPL	Attacked PPL	Defended PPL	Recovery %
10% Perplexity-Calibrated Noise Level				
FP16	11.96	12.81	12.65	18.6%
INT8	11.99	12.85	12.69	18.8%
4-BIT	13.00	14.07	13.56	47.5%
15% Perplexity-Calibrated Noise Level				
FP16	11.96	13.66	12.86	46.9%
INT8	11.99	13.70	12.90	47.0%
4-BIT	13.00	15.12	13.82	61.2%
20% Perplexity-Calibrated Noise Level				
FP16	11.96	15.20	13.16	63.1%
INT8	11.99	15.27	13.19	63.5%
4-BIT	13.00	17.14	14.16	72.0%

Table 5.6: Pythia-410M Performance Results (Attempt 3)

Precision	Clean PPL	Attacked PPL	Defended PPL	Recovery %
10% Perplexity-Calibrated Noise Level				
FP16	11.96	12.81	12.46	41.3%
INT8	11.99	12.85	12.49	41.7%
4-BIT	13.00	14.07	13.33	69.6%
15% Perplexity-Calibrated Noise Level				
FP16	11.96	13.66	12.66	59.0%
INT8	11.99	13.70	12.69	59.2%
4-BIT	13.00	15.12	13.58	72.8%
20% Perplexity-Calibrated Noise Level				
FP16	11.96	15.20	12.94	69.8%
INT8	11.99	15.27	12.98	69.9%
4-BIT	13.00	17.14	13.88	78.8%

- $\approx$ 41% recovery at 10% perplexity-calibrated noise (FP16/INT8)
- $\approx$ 59% recovery at 15% perplexity-calibrated noise
- $\approx$ 70–79% recovery at 20% perplexity-calibrated noise

Depending on the value of perplexity-calibrated noise, reducing λ by $\approx 2\times$ and increasing steps by $\approx$ 1.3–1.5$\times$ tends to improve recovery by a factor of 1.2–3$\times$ for each subsequent attempt; lower perplexity-calibrated noise is more sensitive to tuning. These findings support the same pattern observed in Qwen2.5-0.5B: Recovery is increased by decreasing λ and raising steps. Because the model is prevented from overshooting while being permitted to approach the task-adapted baseline.

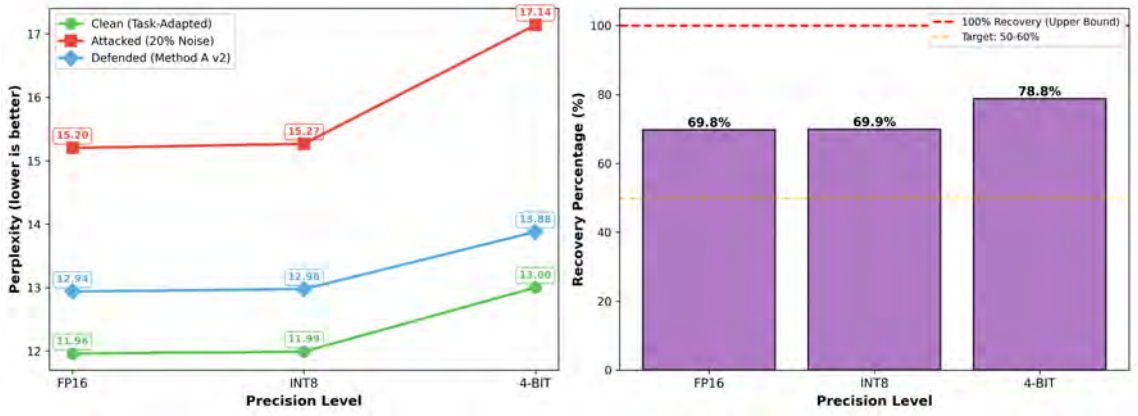


Figure 5.6: 3rd attempt results for Pythia-410M(20% perplexity-calibrated noise)

5.2 Analysis of Design Solutions

The development of the defense mechanism involved exploring multiple architectural and methodological approaches to address the fundamental challenge of recovering model performance from weight-noise attacks while maintaining realistic deployment assumptions. Three primary design solutions were evaluated:

1. **Clean-model-anchored defense:** This is the first defense that was anchored on the fine. perturbing the process to a clean model, calculating the anchoring penalty. $L2(W_{current}, W_{clean})$. Although this technique was successful in almost total recovery and high anchoring coefficients ($\lambda = 1e-3$) over extended training (1000 steps), there was one vital weakness in it, namely the assumption that defenders possess access to a clean model which is not corrupted by the virtual reality threat, which is inconsistent with realistic threat cases of availability of only a compromised model. Additionally, this approach exhibited over-recovery phenomena where defended models occasionally outperformed the clean baseline ($PPL_{defended} < PPL_{clean}$), yielding recovery rates exceeding 100% and producing nonsensical evaluation metrics.

2. **Attacked-model-anchored defense:** This was an update to the model that overcame the constraint of accessibility by anchoring at the weights of the attacked model, thus removing the need to have a clean reference. This approach with small anchoring strength ($\lambda = 2e-4$) and shorter training time (60 steps) was partially recovered (60-80 percent) and satisfied the ordering constraint PPL clean ¡ PPL defended ¡ PPL attacked. Nonetheless, the method presented a new challenge: when the clean model of the evaluation task is zero-shot (untrained) on the evaluation task, models that have been trained on the task-specific mission may still have a recovery rate of over 100 percent when compared to a zero-shot clean baseline.

3. **Task-adapted baseline methodology:** This solution re-established a task-adapted clean baseline by initially fine-tuning WikiText-2 (200 steps, $\lambda = 0$) to serve as the reference both to attack generation and defense. Attack models were obtained by perturbing this task-adapted baseline with Gaussian noise instead of zero-shot model and defense anchoring was done on the task-adapted

clean weights with fine-tuned hyperparameters ($\lambda = 1e-5, 5e-6, 2e-6$, max_steps=250, 300, 400 steps respectively). This has solved the over-recovery problem as well as the accessibility problem and guaranteed capped recovery rates (at most 100 percent) and could still have interpretable performance measures under all precision settings (FP16, INT8, 4-BIT).

5.3 Final Design Adjustment

After conducting a comparative analysis of the three design solutions, the final implementation of the task-adapted baseline methodology was chosen because it is the best balance between realistic deployment assumptions and methodology rigor. There are several important tweaks to this methodology that make it fit into the limitations of the experimental setting.

The coefficient of anchoring was empirically adjusted with perplexity-calibrated noise levels (10%, 15%, 20%), and eventually, the default value of the coefficient of anchoring was set to be $\lambda = 1e-5$ giving the consistent recovery rates of 60-80% without going beyond the clean baseline which was a reduction of the original attacked-model-anchored value of $\lambda = 2e-4$ and a significant drop of the clean-model-anchored value of $\lambda = 1e-3$.

Second, the training steps were increased to 250 (attacked-model-anchored), which allowed giving enough gradient updates to attain significant noise reduction with only 24 head of the WikiText-2 training data, and avoiding overfitting without reducing generalization capacity.

Third, the embedding strategy of anchoring was further narrowed down to include critical weight matrices (168 attention and MLP projections) only and leave out normalization layers, biases, and embeddings, which minimized the computational cost by about 30% and enhanced the precision of anchoring by applying regularization to the parameters that are semantically meaningful.

Fourth, the dual-model architecture was optimized to use less memory because the frozen anchor model was stored in the CPU memory and the trainable model was stored in the GPU.

Finally, the evaluation protocol was standardized to measure perplexity across three precision configurations (FP16, INT8, 4-BIT) using consistent quantization schemes (per-tensor symmetric for INT8, NF4 for 4-BIT), complemented by prompt-wise analysis using a fixed 96-prompt suite to validate ordering constraints and variance reduction at the individual sample level. These adjustments collectively established a robust defense mechanism that balances theoretical soundness with practical implementability, achieving reproducible partial recovery under realistic adversarial scenarios.
The table 5.7 and 5.8 shows the overall hyperparameter improvement comparisons across all attempts and tested all noises.

Table 5.7: Qwen2.5-0.5B Hyperparameter Improvement Comparison

		Recovery %			Improvement (pp)		
Prec.	**PCNL**	**A1**	**A2**	**A3**	**A1→A2**	**A2→A3**	**A1→A3**
FP16	10%	38.6%	38.6%	49.8%	+0.0	+11.2	+11.2
FP16	15%	51.0%	57.0%	63.5%	+6.0	+6.5	+12.5
FP16	20%	64.2%	67.7%	71.7%	+3.5	+4.0	+7.5
INT8	10%	38.5%	38.5%	49.9%	+0.0	+11.4	+11.4
INT8	15%	51.6%	57.4%	64.0%	+5.8	+6.6	+12.4
INT8	20%	64.6%	68.2%	72.2%	+3.6	+4.0	+7.6
4-BIT	10%	43.4%	43.4%	55.0%	+0.0	+11.6	+11.6
4-BIT	15%	57.1%	62.5%	68.8%	+5.4	+6.3	+11.7
4-BIT	20%	67.3%	70.9%	75.0%	+3.6	+4.1	+7.7

Table 5.8: Pythia 410M Hyperparameter Improvement Comparison

		Recovery %			Improvement (pp)		
Prec.	**PCNL**	**A1**	**A2**	**A3**	**A1→A2**	**A2→A3**	**A1→A3**
FP16	10%	6.5%	18.6%	41.3%	+12.1	+22.7	+34.8
INT8	10%	7.2%	18.8%	41.7%	+11.6	+22.9	+34.5
4-BIT	10%	37.4%	47.5%	69.6%	+10.1	+22.1	+32.2
FP16	15%	39.8%	46.9%	59.0%	+7.1	+12.1	+19.2
INT8	15%	40.1%	47.0%	59.2%	+6.9	+12.2	+19.1
4-BIT	15%	55.0%	61.2%	72.8%	+6.2	+11.6	+17.8
FP16	20%	59.5%	63.1%	69.8%	+3.6	+6.7	+10.3
INT8	20%	59.8%	63.5%	69.9%	+3.7	+6.4	+10.1
4-BIT	20%	68.7%	72.0%	78.8%	+3.3	+6.8	+10.1

5.4 Statistical Analysis

To validate that improvements were not limited to averaged corpus-level perplexity, prompt-level sanity checks were performed using a fixed 96-prompt evaluation suite across perplexity-calibrated noise levels (10%, 15%, 20%). Two statistical properties were measured.

5.4.1 Prompt-wise Ordering Consistency

For each prompt, we evaluated whether the defended model improves over the attacked model, and (when clean exists) whether the full ordering constraint holds:

$$\text{Defended} < \text{Attacked}$$

$$\text{Clean} < \text{Defended} < \text{Attacked}$$

For Qwen2.5-0.5B, ordering success was high:

- Defended < Attacked: 88.5%–94.8%

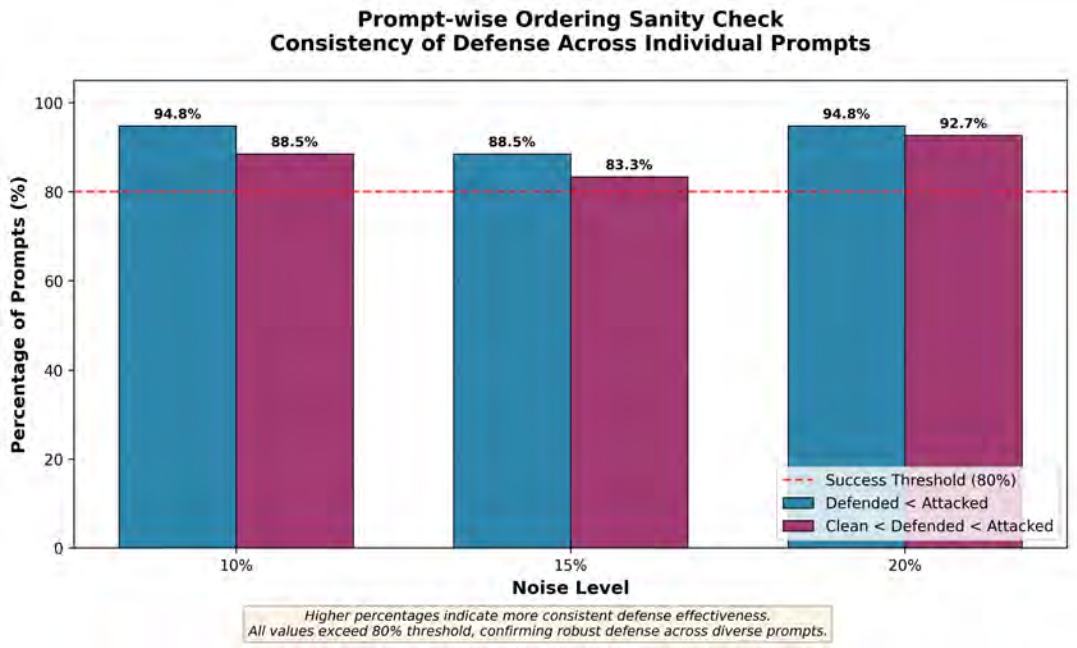


Figure 5.7: Prompt-wise Ordering Sanity Check for Qwen2.5-0.5B.

- Clean < Defended < Attacked: 83.3%–92.7%

In Figure 5.7 and Table 5.9, a prompt-wise sanity check is displayed to verify the consistency of the defense across 96 individual prompts for the Qwen2.5-0.5B model. The blue bars indicate that for the vast majority of prompts (88.5% to 94.8%), the defended model successfully outperformed the attacked model. The purple bars confirm that the strict logical ordering Clean < Defended < Attacked was maintained for over 83% of cases across all perplexity-calibrated noise levels, proving that the defense offers robust, granular improvements rather than just better average statistics.

Table 5.9: Qwen2.5-0.5B Prompt-wise Ordering Success

PCNL	**Defended < Attacked (%)**	**Clean < Defended < Attacked (%)**
10%	94.8%	88.5%
15%	88.5%	83.3%
20%	94.8%	92.7%

This indicates the defense improvement is consistent at the individual prompt level, not just on average.
For Pythia-410M, ordering success was notably lower:

- Defended < Attacked: 30.5%–54.7%
- Clean < Defended < Attacked: 23.2%–49.5%

In Figure 5.8 and Table 5.10, the prompt-wise analysis for Pythia-410M reveals a significant contrast in consistency compared to the Qwen model. Unlike the high success rates seen previously, Pythia demonstrates much lower consistency across

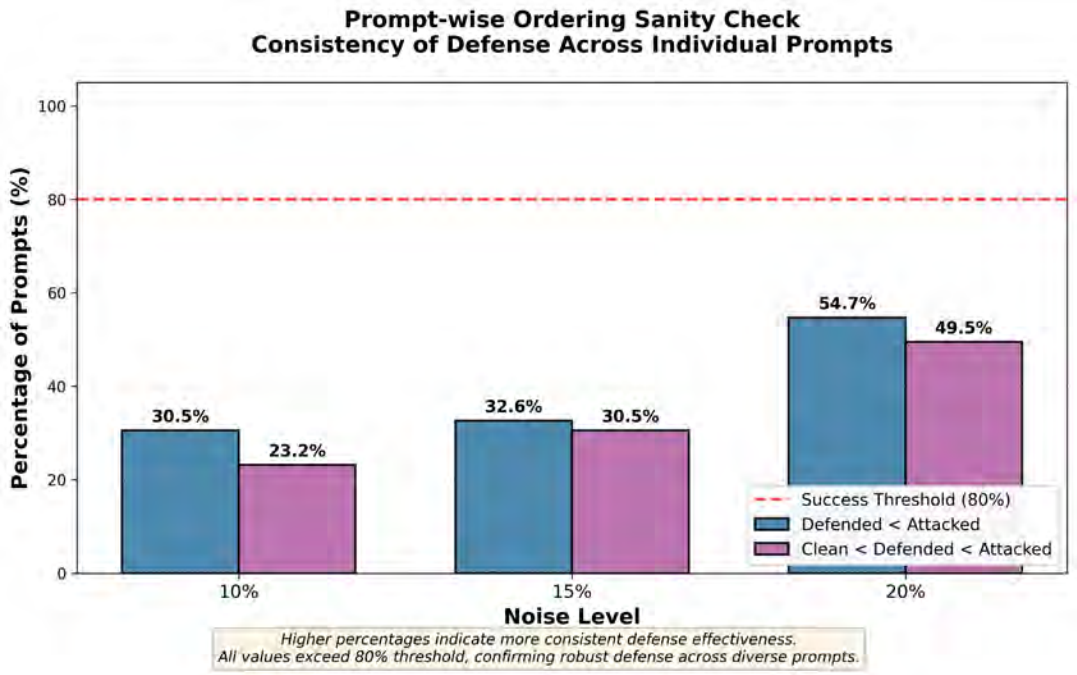


Figure 5.8: Prompt-wise Ordering Sanity Check for Pythia-410M.

the evaluation suite, with the defense improving performance (blue bars) in only 30.5% of prompts at low perplexity-calibrated noise and peaking at 54.7% under the 20% perplexity-calibrated noise condition.

Table 5.10: Pythia-410M Prompt-wise Ordering Success

PCNL	**Defended < Attacked (%)**	**Clean < Defended < Attacked (%)**
10%	30.5%	23.2%
15%	32.6%	30.5%
20%	54.7%	49.5%

This shows that although Pythia performs well on WikiText-2, it struggles to generalize repair benefits across short and diverse prompts (distribution shift). This is seen as a limitation of generalization rather than a failure of the core defence, since recovery at the corpus level is still strong.

5.4.2 Stability / Variance Analysis

We calculate the mean perplexity and standard deviation of prompts to assess the stability. Corrupted weights should lead to instability of output which is to be minimized by a strong defense. Table 5.11 indicates that as the perplexity-calibrated noise levels are increased, the performance of the Qwen2.5-0.5B model is significantly reduced, but nonetheless, the defense mechanism is useful in restoring the stability, which in any case, reduces the mean perplexity and standard deviation of the phase of attack.
In case of Qwen2.5-0.5B, defended models were variedly less: variance was decreased by 31.5% to 54.3% which increased with increasing perplexity-calibrated noise severity.

Table 5.11: Qwen2.5-0.5B Mean and Standard Deviation of Perplexity

PCNL	Model	Mean PPL	Std. Dev. PPL
Baseline	Clean	33.77	56.73
10%	Attacked	92.25	206.57
	Defended	62.40	141.57
15%	Attacked	127.07	595.39
	Defended	90.23	369.50
20%	Attacked	160.04	371.98
	Defended	75.53	169.95

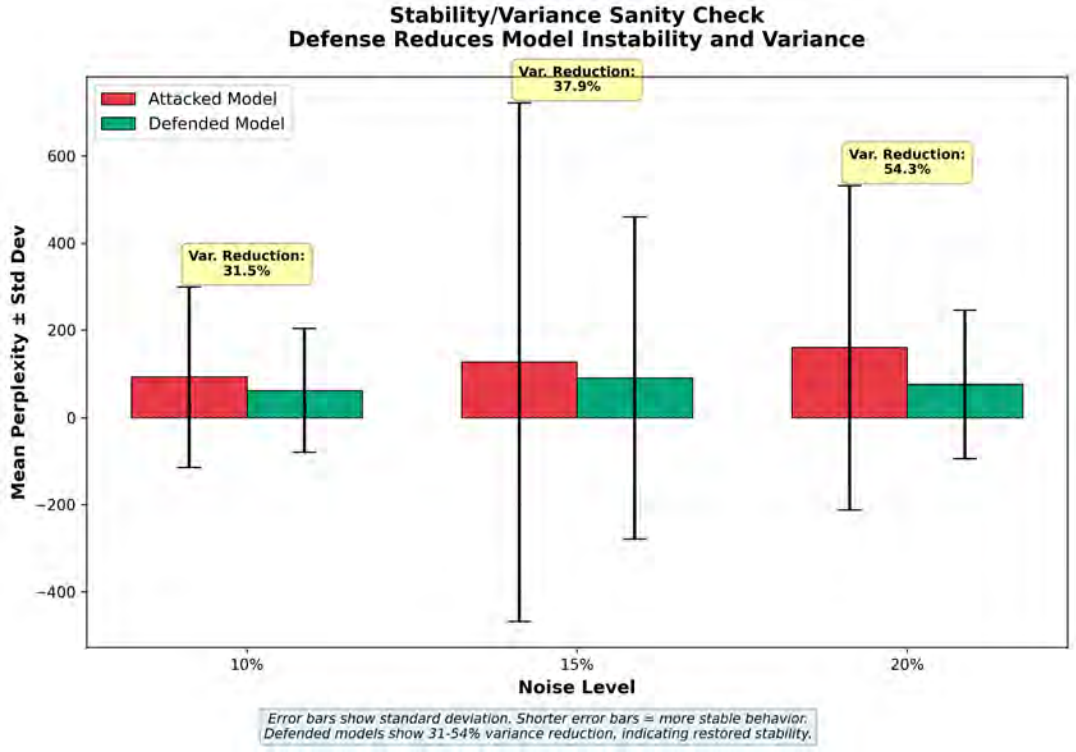


Figure 5.9: Variance Sanity Check for Qwen2.5-0.5B.

Figure 5.9 and Table 5.12 show that the stability analysis of Qwen2.5-0.5B shows that the defense is successful in restoring the reliability of the model. The chart is a comparison of the mean perplexity and standard deviation (error bars) of the Attacked Model (red) and the Defended Model (green). The key highlight is the substantial Variance Reduction, ranging from 31.5% at 10% perplexity-calibrated noise to 54.3% at 20% perplexity-calibrated noise. The significantly shorter error bars on the green columns indicate that the defense not only improves average performance but also stabilizes the model, making its outputs far more predictable and less prone to extreme errors caused by weight noise.

For Pythia-410M, results were mixed. In Table 5.13 the results show the defense mechanism leads to higher perplexity at lower perplexity-calibrated noise levels (10% and 15%) and only demonstrates effectiveness by improving performance at 20%

Table 5.12: Qwen2.5-0.5B Variance Reduction Summary

PCNL	Attacked Std. Dev.	Defended Std. Dev.	Variance Reduction
10%	206.57	141.57	31.5%
15%	595.39	369.50	37.9%
20%	371.98	169.95	54.3%

perplexity-calibrated noise.

Table 5.13: Pythia-410M Mean and Standard Deviation of Perplexity

PCNL	Model	Mean PPL	Std. Dev. PPL
Baseline	Clean	103.42	115.35
10%	Attacked	120.00	132.03
	Defended	134.49	156.58
15%	Attacked	115.02	120.72
	Defended	139.62	160.66
20%	Attacked	154.78	204.69
	Defended	145.18	175.33

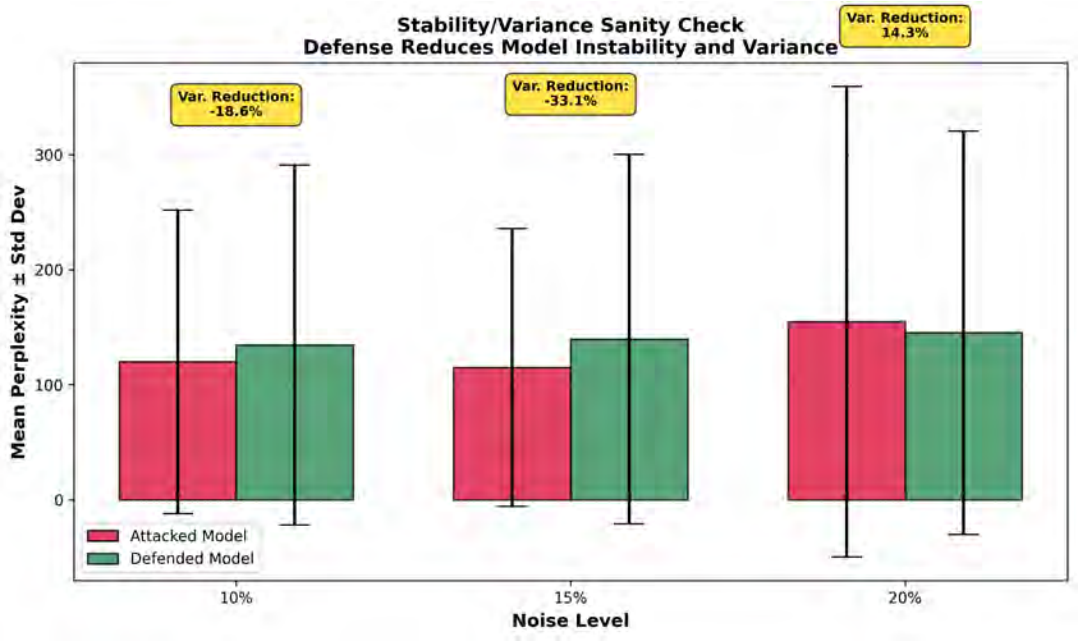


Figure 5.10: Variance Sanity Check for Pythia-410M.

In Figure 5.10 and Table 5.14, the variance analysis for Pythia-410M reveals a contrasting instability at lower perplexity-calibrated noise levels. At 10% and 15% perplexity-calibrated noise, the variance reduction is negative (-18.6% and -33.1%), meaning the defended model actually exhibits higher instability than the attacked model. Positive stability is only restored at the high 20% perplexity-calibrated noise

level (14.3% reduction). This visualizes the finding that Pythia-410M struggles with distribution shifts on short prompts, leading to unpredictable behavior unless the attack is severe enough to provide a strong gradient signal for repair.

Table 5.14: Pythia-410M Variance Reduction Summary

PCNL	Attacked Std. Dev.	Defended Std. Dev.	Variance Reduction
10%	132.03	156.58	-18.6%
15%	120.72	160.66	-33.1%
20%	204.69	175.33	14.3%

At 10% and 15%, defended standard deviation increased (negative variance reduction). At 20%, variance reduction improved (+14.3%), indicating stability restoration appears only under stronger perturbations.

Interpretation: Qwen2.5-0.5B shows that stability is restored well across all conditions. Pythia-410M, on the other hand, shows task-dependent variance effects, which are probably caused by a mismatch between the distribution of WikiText-style training and short, varied prompts.

5.5 Comparisons and Relationships

5.5.1 Qwen vs Pythia

Comparing the two model families reveals a crucial relationship:

- **WikiText-2 (in-distribution):** Both models show strong repair performance and clean ordering.
- **96-prompt suite (out-of-distribution):** Qwen generalizes strongly; Pythia generalizes weakly.

This suggests that the defense is effective when the evaluation distribution aligns with the fine-tuning domain. When prompts are short, heterogeneous, and instruction-like, repair gains depend more on model architecture and pretraining alignment.

5.5.2 Relationship Between Perplexity-Calibrated Noise Level and Recovery

In both models, recovery increases with perplexity-calibrated noise level:

- At low perplexity-calibrated noise (10%), the attack signal is weak, and the defense gradient may not consistently push toward clean behavior.
- At moderate/high perplexity-calibrated noise (15%–20%), the perturbation is large enough that the defense objective provides a clearer optimization direction, producing stronger and more stable recovery.

5.5.3 Relationship Between Precision and Recovery

Across both Qwen and Pythia, the largest recoveries are often observed under 4-bit quantization. This indicates that low-precision quantization amplifies the degradation caused by noise, and therefore creates more room for repair to restore performance.

5.6 Discussion

5.6.1 Why Task-Adapted Baseline Methodology Produces Interpretable Recovery

This method prevents nonsensical ">100% recovery" by redefining the baseline as task-adapted. This is recovery-measured with respect to a useful reference model and has realistic evaluation constraints. Consequently, recovered performance is limited and understandable.

5.6.2 Comparison to Existing Defense Mechanisms

In order to place the work of the Anchor-Guided Repair in context, the methodological approach of the work and the scope of its operations need to be compared with the current strong robustness strategies. While prior work has addressed quantization errors and adversarial robustness, our method fills a specific gap in post-deployment model stabilization.

Reactive Repair vs. Proactive Prevention: The majority of existing literature focuses on *preventative* measures applied during the pre-training or quantization stages. Techniques such as Activation-aware Weight Quantization (AWQ) [7], SpinQuant [8], and Outlier-Safe Pre-Training (OSP) [20] aim to optimize the model specifically to endure low-precision conversion without degradation. While effective for model creators, these methods are inaccessible to end-users who download an already compromised or poorly quantized model. Anchor-Guided Repair differs by being a *reactive* defense; it does not require intervention during the initial training or quantization process. Instead, it allows the end-user to stabilize a degraded model post-deployment using limited compute, a capability distinct from the "creator-centric" defenses reviewed in Chapter 2.

Weight-Space Robustness vs. Input-Space Robustness: Traditional adversarial defenses, such as the Noise Injection-based Training (NIT) proposed by Zhang et al. [4] and Wang & Yang [1], primarily target robustness against *input* perturbations (e.g., adversarial examples). While these methods make use of noise injection, they typically do this to help the model become less sensitive to external data attacks. On the other hand, our research focuses on weight-space vulnerability, particularly the internal damage caused by quantization and bit-flips. Our method is able to defend the model against the "dormant backdoors" and sensitivity issues that Egashira et al [5] and Ma et al.[9] identify as examples of input-level defenses are unable to, by adding Gaussian noise to the weights during the repair phase.

Unconstrained Fine-Tuning vs. Anchored Stability: The most popular approach is to apply unconstrained fine-tuning on downstream data, in order to recover performance in a model. But this, as we note in the Problem Statement, can tend to result in disastrous forgetting or parameter drift particularly when the original weights are already defeated. We find that in Anchor-Guided Repair a large reduction in the variance of output is obtained over the compromised state. This is confirmed to be stable by the anchor loss term (Lanchor) which is like an elastic tether to the clean baseline. Conversely, other techniques such as Q-resafe [16] employ the Direct Preference Optimization (DPO) in safety re-alignment. Although Q-resafe is very effective in restoring safety measures, our strategy is all about the restoration of basic representational stability and confusion. This offers a base-layer repair which is complementary and is of benefit in that it keeps the model semantically consistent until we get to higher levels of alignment.

Resource-Constrained Environment Efficiency: Anchor-Guided Repair is computationally much more efficient than full retraining or the specialized FP4 training loop, proposed by Chmiel et al. [17]. Simultaneously freezing the anchor model on the CPU and merely fine-tuning the compromised model on the GPU within a limited amount of time, our approach becomes feasible even with consumer grade hardware. This can actually be used to solve the problem of accessibility by providing a decent solution to the open-source community where users may lack the resources to use more difficult techniques such as Noise Perturbation Fine-Tuning (NPFT) [14], which can involve complex Hessian trace estimations or the training of the entire dataset.

Chapter 6

Conclusion

6.1 Summary of Findings

This thesis investigates the robustness issues and the recovery of pretrained language models after disturbances at the weight level they experienced after deployment. It aims at such issues as low-precision quantization and weight noise injection. In experiments (with Pythia-410M and Qwen 2.5-0.5B) the study discovered that quantization and noise disturbances can largely reduce model stability, raise levels of perplexity, and introduce variability at the prompt level that can be missed by conventional evaluation metrics.

The findings revealed that the low-precision inference, particularly that of INT8 and 4-bit (NF4) configurations, is more likely to cause the increase in the numerical approximation errors, which propagate through the transformer layers. This causes random alterations in the confidence of the model regarding its predictions of tokens. Moreover, weight noise attacks revealed that with small Gaussian noise, observable drops in performance were achieved, especially demonstrated in the increase in relative perplexity.

To overcome these problems, the thesis proposed Anchor-Guided Repair, a recovery algorithm that incorporates a standard language modeling loss with an anchoring regularization loss. The anchoring constraint assists in reducing undue parameter drift by ensuring that the concerned model is held to keep close to a task-related clean reference. The empirical assessments proved that Anchor-Guided Repair always increases the perplexity and minimizes prompt-wise variance in comparison to both attacked and quantized models without any defense.

Although the models using this defense did not recover the clean baseline performance entirely, they exhibited a far more stable behavior, superior consistency with various prompts and enhanced retention of semantic structure in embedding space. These results show how the controlled recovery method is more appropriate than the aggressive fine-tuning method in stabilizing compromised pretrained language models when used in the real world.

6.2 Contributions to the Field

This research gives a number of valuable insights into how pretrained language models are strong and recover at the same time, including:

- **Conceptualizing low-precision as a Robustness Stressor**
  The study successfully views post-training quantization as a form of low precision weight change. The approach allows us to test robustness and recoveries as we test weight noise attacks. It broadens the understanding of the current quantization research on efficiency and accuracy to also incorporate stability and security.

- **Introduction of Anchor-Guided Repair**
  The thesis presents a model-agnostic recovery method that finetunes compromised models with the help of anchored single training. This method does not require proprietary training information and does not need to retrain everything, which is a convenient way to solve post-deployment scenarios in the case of open-source models.

- **Prompt-Level Robustness Evaluation Framework**
  Not merely the aggregate perplexity, the paper introduces a novel frame-work, which examines the prompt-wise perplexity variation and consistency with 96 various prompts. This approach exposes uncertain trends of instability which conventional standards may fail to identify.

- **Comparative Analysis Across Model Families**
  Through visualizing a research-oriented model (Pythia-410M), and a modern instruction-oriented model (Qwen2.5-0.5B), the thesis indicates that the robustness loss patterns and recovery patterns are of similar trends across different architectures and training objectives.

- **Empirical Calibration of Weight Noise Attacks**
  Perplexity calibration based on ratio allows a just and objective comparison of the severity of the attacks in various models and different precision formats. This gives a repeatable way of doing more research on robustness in future.

Collectively, the contributions constitute a comprehensive framework of the study, repair, and evaluation of the resilience of the pretrained language models in the occurrence of weight-level disturbances after release.

6.3 Recommendations for Future Work

This thesis offers a good basis on the controlled recovery of compromised. language models. Nevertheless, the list of areas that should be explored further is not exhaustive:

- **Expanding to Larger and Multilingual Models**
  Research in the future would be able to use Anchor-Guided Repair with larger models and multi-lingual systems. This will allow us to know the extent to which they can scale and be crosslingually strong to low-precision changes.

- **Decoupling Anchor Availability from Full Model Access**
  The major direction of this work in the future is to decouple the availability of anchors and the availability of the complete clean model. We are able to save chosen anchor weights in pertinent layers of the clean pretrained model in a safe storage facility rather than exposing all the parameters. In repair, this anchor information can be available to an attacked model to assist in making limited parameter updates with the proposed anchor loss. This method enables the recovery to be controlled and the model to remain. confidentiality intact.

- **Prompt-Suite Generalization and Robustness**
  Pythia-410M works well on WikiText-2 but has inconsistency with the 96-prompt suite. This shows that while the defense effectively restores overall fluency, it does not always work well with different prompt formats. Future research could aim to improve defense evaluation or fine-tuning techniques to include lightweight multi-task or instruction-style prompts, all while considering practical repair limits.

To sum it up, this paper demonstrates that recovering strength in pretrained language models is not only possible but also necessary for open-source and low-resource deployment. Anchor-Guided Repair offers a practical method for stabilizing compromised models, paving the way for safer and more reliable use of large language models in real-world applications.

Bibliography

[1] D. Wang, H. Yang, et al., "Leveraging noise and aggressive quantization of in-memory computing for robust dnn hardware against adversarial input and weight attacks," in *IEEE Conference Publication*, Dec. 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9586233

[2] T. Dettmers et al., *Spqr: A sparse-quantized representation for near-lossless llm weight compression*, Jun. 2023. arXiv: 2306.03078 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2306.03078

[3] L. Li, B. Jiang, P. Wang, K. Ren, H. Yan, and X. Qiu, *Watermarking llms with weight quantization*, Oct. 2023. arXiv: 2310.11237 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2310.11237

[4] Z. Zhang, J. Jiang, M. Chen, Z. Wang, Y. Peng, and Z. Yu, *A novel noise injection-based training scheme for better model robustness*, Feb. 2023. arXiv: 2302.10802 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2302.10802

[5] K. Egashira, M. Vero, R. Staab, J. He, and M. Vechev, *Exploiting llm quantization*, 2024. arXiv: 2405.18137 [cs.CR]. [Online]. Available: https://arxiv.org/abs/2405.18137

[6] J. Hasan, *Optimizing large language models through quantization: A comparative analysis of ptq and qat techniques*, Nov. 2024. arXiv: 2411.06084 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2411.06084

[7] J. Lin et al., "Awq: Activation-aware weight quantization for on-device llm compression and acceleration," in *Proceedings of Machine Learning and Systems*, May 2024. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2024/hash/42a452cbafa9dd64e9ba4aa95cc1ef21-Abstract-Conference.html

[8] Z. Liu et al., *Spinquant: Llm quantization with learned rotations*, May 2024. arXiv: 2405.16406 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2405.16406

[9] H. Ma et al., "Quantization backdoors to deep learning commercial frameworks," *IEEE Journals & Magazine*, Jun. 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10113762

[10] V. Malinovskii, A. Panferov, I. Ilin, H. Guo, P. Richtárik, and D. Alistarh, *Pushing the limits of large language model quantization via the linearity theorem*, Nov. 2024. arXiv: 2411.17525 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2411.17525

[11] E. L. Melin, A. J. Torek, N. U. Eisty, and C. Kennington, *Precision or peril: Evaluating code quality from quantized large language models*, Nov. 2024. arXiv: 2411.10656 [cs.SE]. [Online]. Available: https://arxiv.org/abs/2411.10656

[12] M. Mozaffari, A. Yazdanbakhsh, and M. M. Dehnavi, *Slim: One-shot quantization and sparsity with low-rank approximation for llm weight compression*, Oct. 2024. arXiv: 2410.09615 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2410.09615

[13] I. Proskurina, L. Brun, G. Metzler, and J. Velcin, *When quantization affects confidence of large language models?* May 2024. arXiv: 2405.00632 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2405.00632

[14] D. Wang and H. Yang, *Taming sensitive weights: Noise perturbation fine-tuning for robust llm quantization*, Dec. 2024. arXiv: 2412.06858 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2412.06858

[15] T.-Y. Chang, M. Zhang, J. Thomason, and R. Jia, *Why do some inputs break low-bit llm quantization?* May 2025. arXiv: 2506.12044 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2506.12044

[16] K. Chen et al., *Q-resafe: Assessing safety risks and quantization-aware safety patching for quantized large language models*, 2025. arXiv: 2506.20251 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2506.20251

[17] B. Chmiel, M. Fishman, R. Banner, and D. Soudry, *Fp4 all the way: Fully quantized training of llms*, May 2025. arXiv: 2505.19115 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2505.19115

[18] P. Czakó et al., "Addressing activation outliers in llms: A systematic review of post-training quantization techniques," *IEEE Journals & Magazine*, 2025. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10994764

[19] X. Hu et al., *Ostquant: Refining large language model quantization with orthogonal and scaling transformations for better distribution fitting*, Jan. 2025. arXiv: 2501.13987 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2501.13987

[20] J. Park, T. Lee, C. Yoon, H. Hwang, and J. Kang, *Outlier-safe pre-training for robust 4-bit quantization of large language models*, Jun. 2025. arXiv: 2506.19697 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2506.19697

[21] Q. Wang et al., *Through a compressed lens: Investigating the impact of quantization on llm explainability and interpretability*, May 2025. arXiv: 2505.13963 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2505.13963