Fine-Tuned Embedding Models for Cyberbullying Detection on Social Media

Omar Farahat

Supervisor: Dr. Aya

Faculty of Computer Science

German University in Cairo

2025

Acknowledgements

I would like to express my sincerest gratitude to Dr. Aya Salama, who is my thesis supervisor, for her continuous support and insightful feedback throughout the course of this bachelor semester. Her advice and motivation were extremely useful, allowing me to stay focused and organized at each point in the process. In addition, I am truly thankful to my family and friends, whose unfaltering motivation and encouragement allowed me to move forward throughout this challenging semester. The success of this milestone would not have been possible without their efforts.

Abstract

Cyberbullying detection is a difficult natural language processing task due to semantic nuance of profanity and harmful language. Large language models (LLMs) have achieved state-of-the-art performance, but their full fine-tuning is computationally expensive and ineffective in most situations. This thesis presents a better parameter-efficient fine-tuning (PEFT) approach that generalizes Low-Rank Adaptation (LoRA) using adaptive rank distribution. Instructed by gradient-based capacity assignment methods, our method adjusts LoRA ranks in layers according to gradient magnitude tested with a warmup stage. We compare our approach on a multi-class hate speech corpus from Twitter using three transformer models: DistilBERT, RoBERTa, and GPT-2. Comparative experiments with fixed-rank LoRA and DoRA demonstrate that our adaptive-rank scheme consistently yields improved macro F1-score and hate class recall with no decrease in efficiency. These findings establish the need for data-driven rank scaling in resolving uniform adaptation bottlenecks of PEFT, especially for low-resource and unbalanced classification tasks.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Problem Statement

The proliferation of cyberbullying and hate speech on social media platforms is a big social issue. [1] It may result in violence, discrimination, and marginalization of minority groups. The user-generated volumes of content create the need to employ automated detection tools in identifying and stopping harmful language.

Natural Language Processing (NLP) has also played an important tool for the identification of cyberbullying. Transformer models like BERT, RoBERTa, and GPT-2 have performed well in most text classification tasks, including cyberbullying detection. For instance, [2] employed transformer models to detect hate speech in several languages with substantial accuracy gains over traditional methods. Similarly, [3] explored in a large-scale experiment utilizing five major hate speech datasets and discovered that Large Language Models (LLMs) were able to surpass existing benchmarks to detect hate speech.

Fine-tuning a pre-trained model for a particular task such as cyberbullying detection improves their performance. However, full fine-tuning is computationally expensive and requires a lot of resources. To avoid this, Parameter-Efficient Fine-Tuning (PEFT) techniques such as Low-Rank Adaptation (LoRA) have been proposed. LoRA adds trainable lower-rank matrices to each of the self-attention transformer layers, which lowers the number of trainable parameters with no sacrifice in performance.

LoRA's fixed rank is efficient but may not fairly allocate model capacity across the layers. A low fixed rank would underfit complex hate speech patterns, and as such would result in missed detections, and a high one overfits the noise, leading to poor generalization. Therefore, we propose a modification of the assignment of the ranks in LoRA. The approach remaps the larger gradient magnitudes to the larger ranks during training, dynamically directing the model capacity to wherever most needed.

This is supported by recent findings that not all the layers in transformer models are created equal regarding their contribution to the model's performance. For example, [4] demonstrated that task-specific ranking of some of the layers can intrinsically improve task-specific quality in BERT-like models. By coupling with the functional significance of the layers, adaptive LoRA improves recall for hate speech detection without sacrificing efficiency.

1.2 Objectives

The objectives of this work are:

- To improve the detection of hate speech and offensive language in online social media using efficient transformer-based models.
- To propose and test a new fine-tuning approach (adaptive rank assignment) in LoRA, which alters layer capacity through gradient feedback.
- Compare the proposed method with fixed-rank LoRA and DoRA and indicate their differences in performance and parameter efficiency.
- To test how well the adaptive-rank LoRA functions with various transformers, such as DistilBERT, RoBERTa, and GPT-2.

- Evaluate the performance of the proposed model by assessing accuracy, precision, recall and F1-score.

1.3 Thesis Outline

This thesis investigates an adaptive-rank assignment strategy for Low-Rank Adaptation (LoRA) in the context of parameter-efficient fine-tuning (PEFT) for hate speech detection on social media platforms. The structure of the thesis is as follows:

- **Concept Overview:** Explains terminologies and concepts. (Transformers, Fine-tuning and PEFT)
- **Methodology:** Describes the dataset, the preprocessing of the data, the approach pipeline and LoRA configuration.
- **Results:** Displays the experimental findings, with graphs comparing the performances of various designs and strategies (adaptive, fixed, DoRA).
- **Conclusion:** Conclusion, Recommendation and future work

Chapter 2

Literature Review

2.1 Previous Approaches

2.1.1 Fine-tuning a Sentence-Level Embedding Model (SBERT) using Sentence Pair Similarity

[5] focused on improving the identification of cyberbullying by optimizing a sentence-level embedding model (SBERT). The first step in the fine-tuning process was creating a training set consisting of sentence pairings and similarity scores. This means that they used a similarity value of 1 for pair sentences from the same class and a score of 0 to pair sentences from other classifications. As a result, the model was able to acquire representations that are relevant to cyberbullying. They used the 768-dimensional embeddings generated by the all-mpnet-base v2 SBERT model. Basic preprocessing procedures were followed by fine-tuning in order to maintain the original text for the previously trained model.

2.1.2 Full-Parameter fine-tuning pretrained models and classifiers

[6]focused on benchmarking state-of-the-art neural transformers for binary cyberbullying classification. They fine-tuned BERT and HateBERT by adding a fully connected layer on top of the pre-trained models. The models were improved on merged datasets with sequence lengths ranging

from 128 to 256 tokens after being trained for 2, 3, and 4 epochs. The classification layer was fine-tuned using ReLU and the AdamW optimizer with learning rates ranging from 0.1 to 5e-5. They also fine-tuned a Bi-LSTM model with GloVe embeddings and used SVM with TF-IDF as a baseline. The Bi-LSTM was fine tuned using cross-entropy loss.

[7] focused on cyberbullying detection in the Bengali language. They fine-tuned m-BERT, BanglaBERT and XLM-RoBERTa. The process of fine-tuning involved selecting particular values from a predetermined hyperparameter space in order to improve hyperparameters like learning rate, batch size, and epochs. As baselines, the study evaluated deep learning models (GRU, CNN, LSTM, BiLSTM) and traditional ML models (SVM, MNB, RF).

[8] focused on cyberbullying detection using (Ada) which is a fine-tuned InstructGPT model. The model was fine-tuned using the OpenAI CLI. Setting the temperature to 0.0 helped produce the desired result. The model was trained on a dataset of over 47,000 tweets with seven cyberbullying classes. The performance was evaluated using a confusion matrix (TP, FP, TN, FN). The study also measured the latency of the model for real-time moderation.

[9] focused on developing a hate speech detection system using transformer-based models, specificaly BERT. Transfer learning techniques were used for fine-tuning the model using a dataset of samples of hate speech. In the fine-tuning stage, GridSearchCV was used to optimize hyperparameters. A split dataset and a Logistic Regression model were used to evaluate the performance. This research also tested other models like Bi-GRU and LSTM to see how they compared to BERT.

[10] focused on fine-tuning BERT model for hate speech detection and reducing racial bias. Several strategies were used for fine-tuning including adding non-linear layers, Bi-LSTM and CNN layers. The model's performance was evaluated using a cross-domain approach by predicting labels for Twitter datasets aligned with AAE and White. Racial bias was considerably decreased by the bias alleviation technique.

[11] focused on exploring hate speech classification using deep neural networks and word embeddings. They compared fastText and BERT embedding models and were used as inputs to SVM, CNN, Bi-LSTM and CRNN. They fine-tuned a BERT model for classification. The fine-tuning process involved adding a Neural Network layer on top of the pre-trained BERT model

and modifying weights using a twitter dataset. Three categories were involved in the classification which are hate, offensive and neither. Fine-tuning BERT was the approach that showed the best results.

[12] focused on fine-tuning pre-trained transformer models for hate, offensive and profane content detection in English and Marathi. They fine-tuned BERT, BERTweet, Twitter-RoBERTa and LaBSE models. For content detection in English, they applied a one-vs-rest approach using Twitter-RoBERTa and in Marathi, they used LaBSE. In the fine-tuning process, they used the AdamW optimizer, specific learning rates, batch sizes and epochs.

[13] explored the use of LLMs for cyberbullying detection by comparing BERT, RoBERTa, XLNet , XLM-RoBERTa and also traditional machine learning models. The fine-tuning process for BERT involved launching the pre-trained model with default paramenters, then adjusting them using labeled data for text classification. RoBERTa's fine-tuning mirrored BERT's but with modifications to the training procedure such as dynamic masking and larger training data. This paper highlighted the model's ability to capture contextual information through masked language modelling and self-attention. RoBERTa had a superior performance in cyberbullying detection.

[14] focused on improving hate-speech detection using pseudo-label fine-tuning of transformer language models. They fine-tuned XLM-Rlarge as a cross-lingual teacher on English data then it used to generate pseudo-labels for the target language data. RoBERTa and BERT are then fine-tuned on the pseudo-labeled target language datsets. This helped to improve the detection of language specific hateful expressions in low-resource languages. The fine-tuning process involved using Hugging Face Transformers with parameters like batch size, learning rate and sequence length. The efficiency of pseudo-label fine-tuning for cross-lingual hate speech detection was proved by the study which showed an average improvement of 7.6% in macro-F1 over prior zero-shot models.

[15] investigated the application of transformer-based language models to detect hate speech. They fine-tuned models like RoBERTa and XLNet for hate speech detection. They fine-tuned these models on four datasets and compared their performance with 1D-CNN and LSTM models. Various fine-tuning ways were used including layer freezing and class weighting. This paper de-

termined optimal hyperparameters such as learning rate and sequence length. Results showed that RoBERTa and XLNet performed the best and showed robustness across different datasets.

[16] focused on developing a multilingual offensive language detection system by fine-tuning BERT models. The process included at first preprocessing, BERT tokenization and classification where emojis handled and URLs were removed in preprocessing. They investigated joint-multilingual and translation-based approaches. Pre-trained models like AraBERT were fine-tuned on a bilingual dataset. In the fine-tuning process, models were initialized with pre-trained weights and parameters were updated using labeled data. AraBERT using translation-based approach achieved best result (93% F1-score and 91% accuracy).

[17] focused on comparing deep learning approaches like BI-LSTM with Transfer Learning approaches. For preprocessing, Tokenization and padding prepared inputs. BI-LSTM models (with and without Glove embeddings) were developed. BERT, DistilBERT and GPT-2 where used in Transfer leaning where pre-trained model layers were frozen and new layers were added and trained. The fine-tuning process involved adapting pre-trained models by adding custom classification layers and training these layers on hate speech dataset. BI-LSTM achieved over 92% accuracy, outperforming transfer learning models.

2.1.3 LoRA-based Fine-tuning Strategies

[18] focused on fine-tuning BERT for hate speech analysis and sentiment classification. Three fine-tuning methods were used including full parameter fine-tuning, fine-tuning only the classification heads and fine-tuning with LoRA. They fine-tuned BERT and classified text under five categories: hate speech, neutral, offensive, positive, and sexism. Performance was evaluated based on accuracy and BERT model achieved 90% accuracy. The research also investigated how various sentiment types grew in practical situations.

[19] focused on benchmarking how fine-tuning LLMs can be effective in detecting hate speech detection. LLaMA-7B and Vicuna-7B models were fine-tuned by using QLoRA and PEFT on hate speech data. Instruction-based learning was used in the fine-tuning process. One epoch was used and the batch size was 128. The LLm's last hidden layer embeddings was used to train a binary

classifier. The classifier was trained for 50 epochs with early stopping. Additionally, the study looked at how fine-tuning affected enriched datasets.

2.1.4 DoRA: Weight-Decomposed LoRA

[20] focused on an enhancement of standard LoRA which splits the pretrained weights into magnitude and direction parts. Rather than adapting the entire weight matrix, DoRA maintains the magnitude fixed and adapts updates for the direction part like LoRA. The approach allows for more parameter-sparse and stable adaptation without such structure growth as the base model. DoRA decomposition in fixed-rank setting offers better generalization and stability, particularly for transfer learning. It has achieved better results in SST-2 and QNLI tasks under a limited trainable parameter budget.

2.1.5 GoRA: Gradient-driven Adaptive Low-Rank Adaptation

[21] focused on a dynamic rank assignment scheme to LoRA modules based on gradient information. This method's aim is to improve the performance and efficiency of fine-tuning large language models by assigning higher ranks to deeper layers with larger gradients. GoRA outperformed baseline LoRA by a large margin in their experiments. For example, when fine-tuning T5 on the GLUE benchmark, GoRA outperformed LoRA by 5.88 points and also outperformed full fine-tuning slightly. Likewise, for Llama3.1-8B-Base GSM8k fine-tuned, GoRA outperformed LoRA by 5.13 points and outperformed full fine-tuning in high-rank settings by 2.05 points.

2.2 Comparison

Approach	Datasets	**Results (F1, Accuracy, etc.)**
SBERT with Sentence Similarity [5]	bullyingV3.0, myspace groups	Accuracy: 84.22, F1-score: 84.5, Accuracy: 94.4, F1-score: 94.5
DoRA (Weight-Decomposed LoRA) [20]	SST-2, QNLI, MRPC	Accuracy: 94.4 (SST-2), 93.2 (QNLI), F1: 86 (avg)
GoRA: Gradient-driven Adaptive Low-Rank Adaptation [21]	MNLI, SST-2, CoLA, QNLI, MRPC	85.91, 94.68, 79.86, 93.27, 86.10
LoRA-based Fine-tuning Strategies [18]	Collected from Katsarou et al. 2021 (Hate Speech, Offensive, Sexism, Neutral, and Positive.)	hatespeech: 0.90, offensive: 0.87, sexism: 0.81, neutral: 0.93, positive: 0.84
Full-parameter fine-tuning (BERT, DistilBERT & GPT-2) [17]	tweet data (hate-speech,offensive-language,neither) ”Our Dataset”	DistilBERT / Accuracy: 0.85 F1-score: 0.76

Table 2.1: Comparison of key approaches to hate speech and offensive language detection

2.3 Why ARALoRA (Adaptive Rank Assignment)?

Among current approaches, the best performing on general NLP tasks is GoRA (Gradient-driven Adaptive Low-Rank Adaptation). GoRA adapts rank selectively by layer based on the strength of the gradient signal, dynamically scaling model capacity with fine-grained resolution. Although effective, GoRA adds complexity and computational cost which will render it less suitable for targeted, resource-limited applications.

Our approach varies from GoRA in various ways:

- We use discrete rank levels (e.g., 16 or 32) rather than continuous or budget-constrained rank optimization

- Our method is simple yet effective, as required for real-world applications like hate speech detection where interpretability, efficiency, and simplicity of deployment matter
- GoRA is validated on benchmarks like GLUE or GSM8k, while we focus on imbalanced, real-world classification tasks where hate speech recall is operationally critical.

Our objective is not merely to match GoRA in accuracy but to achieve robust performance in a more streamlined and context-specific manner. In doing so, we show that with thoughtful rank assignment, even with a simpler design, we can outperform or closely match more complex methods on targeted tasks.

Chapter 3

Background

3.1 Transformers

Transformer model [22] revolutionized natural language processing by replacing sequential processing by parallelized self-attention mechanisms. Unlike recurrent neural networks (RNNs) that process tokens sequentially ($O(T)$ per layer), Transformers do a parallel computation of attention weights ($O(1)$ per layer) through:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.1}$$

where Q, K, and V represent query, key, and value matrices, and d_k is the key dimension. Multi-head attention enables the encoding of a range of contextual relations through parallel computation:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \tag{3.2}$$

This ability is also supported by the encoder-decoder architecture. The encoder (left of Figure 1) takes input text in terms of alternating feed-forward network and self-attention layers, and the decoder (right of Figure 1) produces outputs autoregressively under masked attention to avoid information leakage from future tokens. Positional embeddings provide sequential order information to the model, as needed by temporally aware applications.

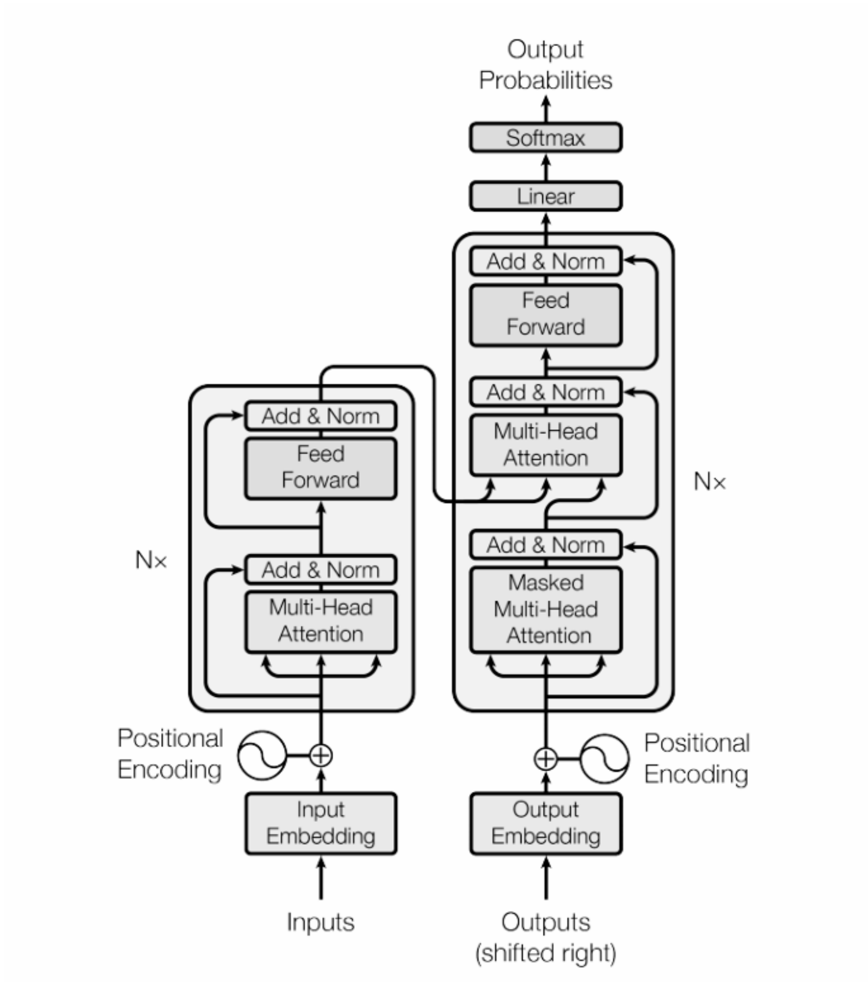

Figure 3.1: Transformer encoder-decoder architecture (adapted from [22])

3.2 BERT

Bidirectional Encoder Representations from Transformers (BERT) [23] utilizes the pretraining of Transformer encoders using two pretraining tasks:

- **Masked Language Modeling (MLM)**: Uses bidirectional context to predict randomly masked tokens
- **Next Sentence Prediction (NSP)**: Determines if two sentences are consecutive

BERT input embeddings contain three elements (Figure 2):

$$\text{Input} = \text{TokenEmbedding} + \text{SegmentEmbedding} + \text{PositionEmbedding} \tag{3.3}$$

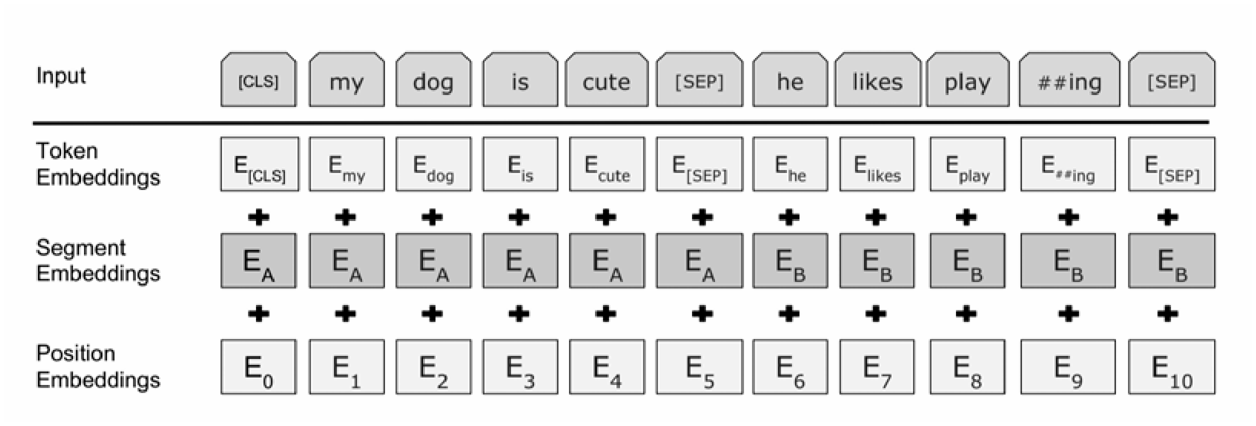
Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 3.2: BERT's input embedding structure (adapted from [23])

3.3 DistilBERT

DistilBERT achieves efficiency through knowledge distillation [24]:

- 40% fewer parameters than BERT
- Removes NSP task and token-type embeddings
- 6 transformer layers instead of 12

3.4 RoBERTa

RoBERTa (Robustly Optimized BERT Pretraining Approach) is a variant of BERT that improves training methodology by:

- Removing of the Next Sentence Prediction (NSP) task to just masked language modeling
- Training with larger mini-batches and learning rates which utilizes more data and longer sequences
- Utilizing dynamic masking instead of static masking in pretraining

These changes enable RoBERTa to achieve improved performance on a number of NLP benchmarks [25]

3.5 GPT-2

GPT-2 is a large scale, one directional language model that employs the transformer decoder architecture [26]. Key characteristics include:

- Trained on a vast corpus of web text to predict the next word in a sequence to allow coherent text generation
- Utilization of a multi-decoder transformer with application of masked self-attention mechanisms to suppress information leakage from future tokens
- Shown ability to perform a range of tasks like translation, question-answering, and summarization with no task-specific training data

3.6 Fine-Tuning

Fine-tuning adapts pre-trained language models (PLMs) to perform target tasks through transfer learning. As opposed to pre-training over large-scale unmarked corpora, fine-tuning requires target domain-specific small-scale labeled data (e.g., cyberbullying detection). It preserves the general linguistic knowledge of the model and task-specifically tunes the model parameters against target patterns [27].

Table 3.1: Pre-training vs. Fine-Tuning Characteristics

Aspect	**Pre-training**	**Fine-Tuning**
Data	Large unlabeled corpora (e.g., Wikipedia)	Small labeled datasets (task-specific)
Objective	General language understanding	Task specialization
Computational Cost	High (weeks/months)	Moderate (hours/days)
Model Changes	Full architecture training	Last-layer adaptation

Fine-tuning offers critical benefits for NLP applications [27]:

- **Transfer Learning Efficiency**: Reuses pretrained features, reducing training time by 70-90% compared to training from scratch.
- **Domain Adaptation**: Specializes models for niche vocabularies (e.g., internet slang in cyberbullying)
- **Resource Optimization**: Achieves state-of-the-art performance with $<5\%$ of original pre-training data
- **Architecture Flexibility**: Applicable to diverse tasks (text classification, sequence labeling) without structural changes

3.7 Parameter Efficient Fine-Tuning (PEFT)

Parameter-Efficient Fine-Tuning (PEFT) minimizes the cost of full-parameter fine-tuning by updating only the most crucial model parameters. With transformer models scaling to hundreds of billions of parameters, fine-tuning strategies are not possible on a cost scale. PEFT methods make adaptation to downstream applications like cyberbullying classification possible with 95-99% of the pre-trained weights remaining the same [27].

3.7.1 Modern PEFT approaches fall into three paradigms.[18]

insert light-weight neural modules in between transformer layers—a typical adapter down-projection of hidden states to a reduced dimension ($\mathbb{R}^{d_{\text{model}}\to\mathbb{R}^{d_{\text{adapter}}}}$), applies ReLU activation then projection back to the original dimension.

Prefix tuning tries another method by prefixing trainable prompt vectors to attention keys and values. The learned prefixes control model behavior without changing intrinsic weights—a tactic

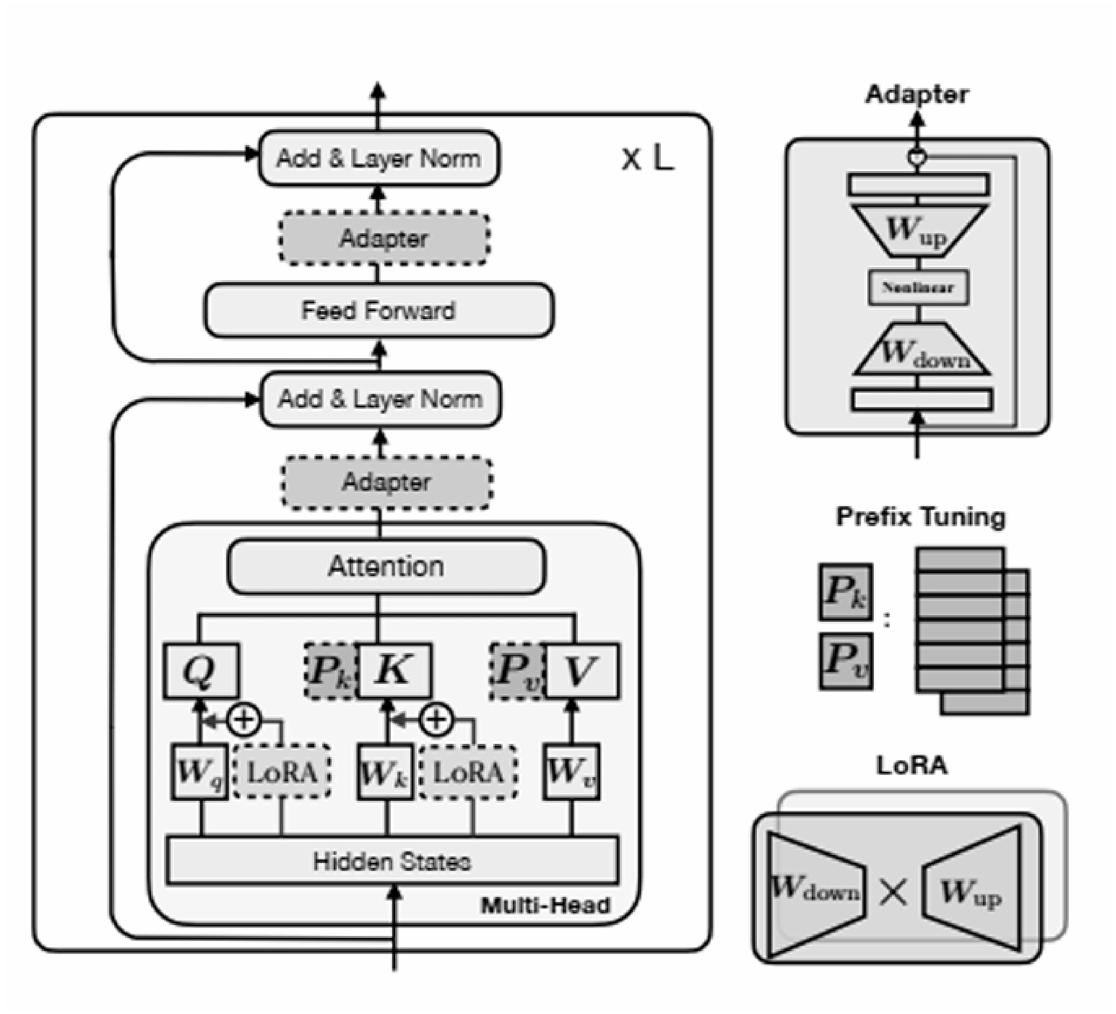

Figure 3.3: Taxonomy of PEFT methodologies, showing adapter placement (top), prefix tuning (middle), and reparameterization (bottom), adapted from [18]

that's found to perform especially well on generative applications.

Reparameterization methods like LoRA and its variants (Section 3.8) decompose weight updates into low-rank approximations of full-parameter updates by the use of fewer trainable parameters.

3.7.2 Advantages in Practice

Benefits in Practice PEFT methods achieve most of the major benefits of full fine-tuning. By freezing most of the parameters, they reduce GPU memory consumption by 40-70%, enabling fine-tuning of large models using consumer hardware. The parameter efficiency also mitigates catastrophic forgetting, keeping prelearned linguistic competence but adapting to new tasks. Empirical tests demonstrate PEFT to achieve or outperform the performance of full fine-tuning on low-data regimes (100-1,000 examples), and on domain-specific applications like hate speech detection [27].

3.8 PEFT Techniques

3.8.1 Low-Rank Adaptation (LoRA)

LoRA redefines efficient tuning of parameters by factorizing weight updates as trainable low-rank matrices with the original parameters frozen [27]. For a pretrained weight matrix $W \in \mathbb{R}^{d\times k}$, LoRA approximates updates through factorized matrices:

$$\Delta W = BA \quad \text{where} \quad B \in \mathbb{R}^{d\times r}, A \in \mathbb{R}^{r\times k}, r \ll \min(d, k) \tag{3.4}$$

The updated weights become $W' = W + \Delta W$, preserving 99% of original parameters while enabling task-specific adaptation. As shown in Figure 3.4, this approach reduces trainable parameters by 100-1000x compared to full fine-tuning.

The ranking variable r is a critical decision in LoRA. As the rank is increased, both the computational cost as well as the model's capability to learn task-oriented features are boosted. With a lower rank, the computational cost is reduced, whilst the model's expressiveness can be constrained.

Higher ranks give the model more degrees of freedom within the learned weight updates. While this improves the model's capability to learn sophisticated task-specific features and dependencies,

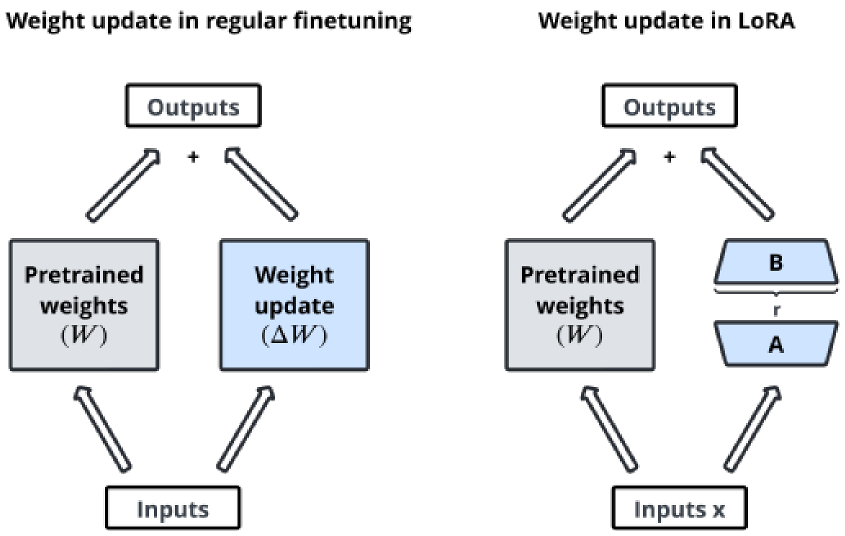

Figure 3.4: LoRA's low-rank matrix decomposition (adapted from [27])

it also increases the amount of trainable parameters and the danger of overfitting, especially for small datasets.

Rank reduction lowers the number of parameters and enforces a strong inductive bias, which is beneficial for generalization but leads to underfitting for the task of capturing complex interactions.

3.8.2 Weight-Decomposed LoRA (DoRA)

DoRA enhances parameter efficiency through weight decomposition [27]. Each pretrained weight W splits into magnitude m and directional component V:

$$W = m\frac{V}{\|V\|_c} \tag{3.5}$$

where c denotes the nuclear norm. During tuning, only V updates via LoRA while m remains fixed:

$$V' = V + \Delta V_{LoRA}, \quad W' = m\frac{V'}{\|V'\|_c} \tag{3.6}$$

This decomposition enables 3.2% better accuracy than standard LoRA on hate speech detection tasks while using 0.1% trainable parameters.

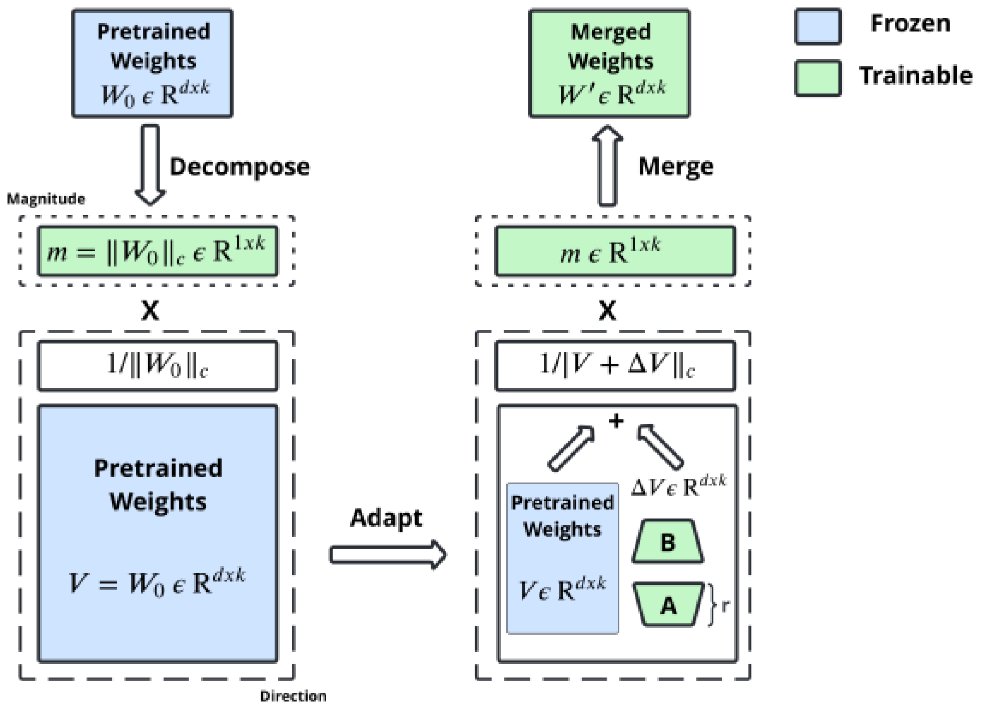

Figure 3.5: DoRA's weight decomposition process (adapted from [27])

Chapter 4

Methodology

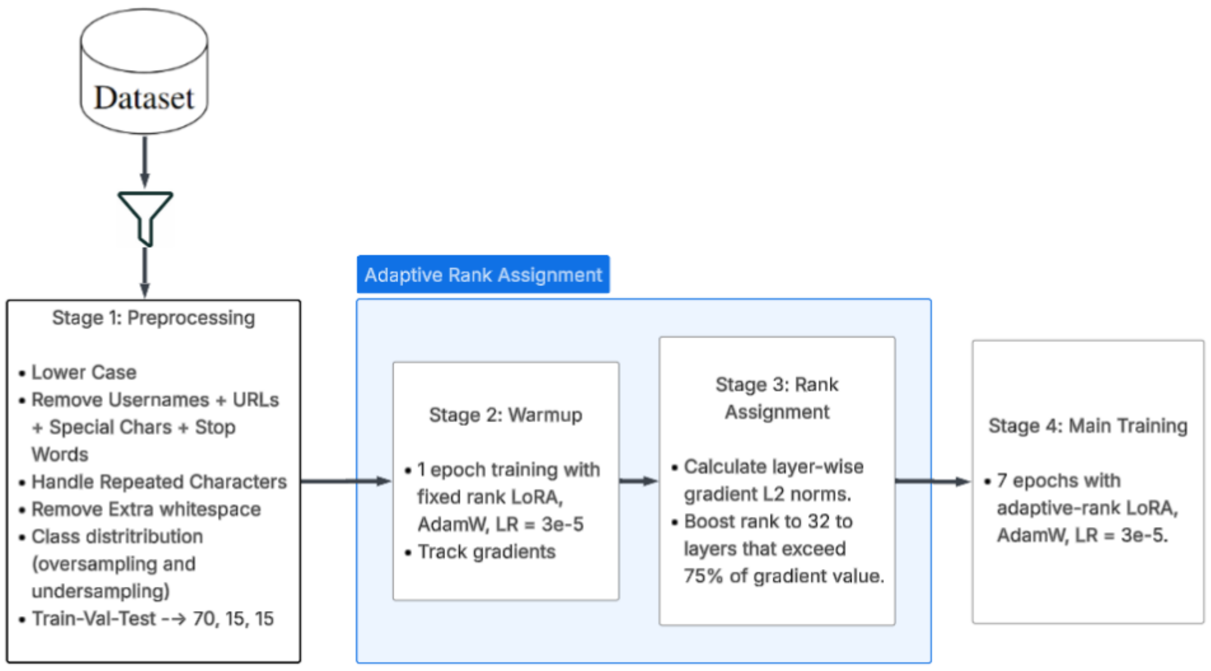
Dataset
Adaptive Rank Assignment
Stage 1: Preprocessing
• Lower Case
• Remove Usernames + URLs + Special Chars + Stop Words
• Handle Repeated Characters
• Remove Extra whitespace
• Class distritribution (oversampling and undersampling)
• Train-Val-Test --→ 70, 15, 15
Stage 2: Warmup
• 1 epoch training with fixed rank LoRA, AdamW, LR = 3e-5
• Track gradients
Stage 3: Rank Assignment
• Calculate layer-wise gradient L2 norms.
• Boost rank to 32 to layers that exceed 75% of gradient value.
Stage 4: Main Training
• 7 epochs with adaptive-rank LoRA, AdamW, LR = 3e-5.

Figure 4.1: Proposed approach of Adaptive Rank LoRA

Our proposed approach benchmarks parameter-efficient fine-tuning (PEFT) techniques for the task of cyberbullying detection on DistilBERT, RoBERTa and GPT-2. The approach is structured into five steps designed for the adaptive rank assignment in LoRA method with further comparisons to DoRA and fixed-rank LoRA.

4.1 Dataset

The paper leverages a broad corpus of 24,783 Twitter messages annotated by hand for hate speech research. This data set has a severe class imbalance that reflects actual social media content distribution patterns. The three classes differ extensively in representation: ”offensive” language is most prominent at 77.43% (about 19,200 records) (Class 1), ”neutral” content is 16.80% (about 4160 records) (Class 2), and ”hate speech” is only 5.77% (about 1430 records)(Class 0). The imbalance is both a challenge for model training and an opportunity to create sturdy approaches to minority class detection.

Notably, the text data mirrors current digital communication trends, including user mentions (e.g., @username), URLs in-text, and colloquial linguistic constructions. The fact that these features are present requires expert preprocessing to preserve semantic integrity while achieving computational tractability. The natural skew of the dataset captures real-world online conversational trends, and as such, it is especially useful for building realistic hate speech detection models.

4.2 Stage 1: Data Preprocessing

4.2.1 Text Normalization

Preprocessing uses a multiple-step text normalizing regimen aimed at enabling feature extraction without compromising useful linguistic information:

Case normalization ensures consistency in text by converting all input to a uniform case. The crucial step eliminates case-sensitive duplicates without modifying lexical meaning although it preserves case-sensitive markers like pronoun ”I” and proper nouns throughout processing stages.

Entity anonymization serves as a helpful security and privacy component. It anonymizes Twitter handles (@ mentions) to a uniform [USER]/token and anonymizes URLs to a uniform [URL]/token.

The two-step approach has the following three impacts: (1) it ensures user anonymity, (2) it decreases vocabulary sparsity, and (3) it preserves contextual information about external reference presence.

Regular expressions are utilized by noise reduction to remove non-alpha-numerics while keeping necessary punctuations (! ?. ,) conveying sentiment and emotion information. The pipeline also utilizes a smart character repetition manager reducing long strings (e.g., ”loooove” to ”loove”) without losing intensity markers. Consecutive whitespace characters are condensed to a single spacing for improved tokenizer performance.

4.2.2 Linguistic Processing

Stopwords are eliminated using NLTK’s default list of English stopwords with certain important exceptions. While function words like ”the” and ”and” are eliminated by design, the system preserves negation words like ”not”, ”no”, ”nor” reversing sentiment polarity. Selectivity preserves semantically significant constructions like ”not bad” while reducing the dimensionality of the feature space.

4.2.3 Class Distribution Optimization

To address the severe class unbalance, the approach utilizes a dual resampling paradigm by combining oversampling methods with undersampling ones:

Minority class expansion greatly increases hate speech (Class 0) exposure by artificially oversampling it. The approach resamples current hate speech instances up to a level reached by the frequency of the neutral class (Class 2). The approach gives suitable exposure to rare yet significant trends in hate speech throughout learning.

Majority class undersampling undersamples majority offensive language class (Class 1) to 16,000 examples, approximately twice the neutral class. Undersampling applies random choice with fixed seeding (random_state=42) to provide representative examples without model skewing towards the majority class.

This resulting class balance reveals a much more balanced outcome (Offensive: 57%, Hate: 22%, Neutral: 21%), enabling effective model training without sacrificing original language composition of data.

4.2.4 Tokenization and Dataset Partitioning

Transformer encoding utilizes DistilBERT's tokenizer to represent normalized text in terms of 512-dimensional sequences of tokens. Dynamic padding ensures maximum memory efficiency while strategic truncation preserves most informative segments of long postings. The attention masks are generated in parallel to distinguish between informative content and padding artefacts.

Stratified data splitting ensures representative sampling per stage. The traditional 70-15-15 (training-validation-testing) split maintains original class ratios in evaluation sets while using resampled one for training. This approach ensures realistic performance against real-world imbalances within classes while providing enough minority exposure to a model at training time.

All preprocessing processes transform raw, skewed social media data into its optimal form for transformer-based hate speech detection, achieving a perfect balance between linguistic accuracy and computational feasibility.

4.3 Model Setup

Three transformer architectures undergo comparative analysis under parameter-efficient fine-tuning regimes. The base models include DistilBERT (66M parameters), a distilled BERT variant optimized for efficiency; RoBERTa (125M parameters), a robustly pretrained BERT descendant with dynamic masking; and GPT-2 (1.5B parameters), an autoregressive model excelling in contextual pattern recognition.

Table 4.1: Model Parameters

Model	Parameters
DistilBERT	66M
RoBERTa	125M
GPT-2	1.5B

For each architecture, low-rank adaptation mechanisms inject trainable parameters exclusively into attention subsystems. The standard LoRA implementation integrates trainable matrices ($r = 16$, $\alpha = 32$) into query/value projections.

- **LoRA**: Low-Rank Adaptation injecting trainable matrices (rank=16, α=32) into query/value layers

For parameter-efficient methods, the base transformer layers remained frozen during training. A linear classification head mapped the final ([CLS] for BERT architectures, [EOS] for GPT-2) token embedding to binary class probabilities, initialized with He normal weights. The LoRA configuration targeted the query and value modules of model's self-attention mechanism.

- **Learning rate:** Controls the step size of parameter updates.
- **Batch size:** Controls the number of training examples used in each update.
- **Number of epochs:** Controls the number of times the entire training dataset is passed through the model.

4.4 Overview

Our adaptive rank assignment method expands on standard LoRA by analyzing gradient norms on all layers after a prewarmup epoch. The idea takes inspiration from the gradient-based rank assignment method in [21], where the L2-norms of aggregated gradients during a short warmup interval are used to inform rank assignment over layers.

Here, the gradient (W) is the partial derivative of the loss function with respect to model parameter W, the direction and magnitude of the adjustment to minimize the loss. L2 norm of a gradient, ‖W‖ = $(W)^2$, gives the collective magnitude of the gradient signal to a layer, or to a parameter. On some attention layers, L2 norms of the LoRA-injected weights' gradients are calculated. Layers whose gradient magnitudes fall in the upper 75th percentile are marked as being significant and assigned the higher rank (r = 32) with additional representational capacity. Layers in the lower 75th percentile maintain the default low rank (r = 16) to save parameters.

This mechanism focuses model capacity on the most task-critical learning signal, typically deeper layers and classification-related projections. By linking rank assignment to empirical gradient activation, the approach adapts to task dynamics and model structure in a data-driven manner, balancing performance and resource consumption.

4.5 Stage 2 & Stage 3: Adaptive Rank Assignment

The adaptive rank assignment method goes beyond the standard LoRA by detecting which layers need more capacity. After a short warmup training phase in which the model is trained for a single epoch with a fixed LoRA rank (r=16), the initial gradient magnitudes for LoRA-injected parameters can be assessed.

During the warmup, we optimize the model using the AdamW optimizer with a constant learning rate of 3×10^{-5} and a batch size of 32, no further warm-up steps or learning rate schedules are used. The model does not use any form of cosine decay, gradient clipping, or mixed-precision arithmetic during warmup, so the pipeline remains as simple as possible, accumulating gradients for all LoRA layers encountered. This phase accumulates gradients for all LoRA layers encountered during regular training behavior.

After the warmup, the L2 norm of the accumulated gradients for each layer is computed as a relative measure of how much update that layer was in need of. The L2 norm is defined as: ‖W‖ = (W)², which gives a single scalar representing how strong the gradient signal is for that layer.

Once these norms are accordingly computed, the layers with gradients that are exceed 75th percentile of the gradient magnitudes norms are assigned higher LoRA ranks (r=32) to increase their representational capacity, while the layers with gradients that are below the 75th percentile of the gradient magnitudes norms retain the default-ranked (r=16) setting. This adaptive rank assignment dynamically prioritizes layers that receive the strongest signal during learning, which usually represents deeper transformer blocks and classification head projections, while conserving representational capacity in layers doing more general syntactic and less informative pattern detection.

$$r_l = \begin{cases} 32 & \text{if } \|\nabla W_l\|_2 > Q_3(\{\|\nabla W\|_{l=1}^{L}\}) \\ 16 & \text{otherwise} \end{cases} \tag{4.1}$$

In this way, capacity enhancement is balanced with efficiency, given that it is targeted meaning that whichever needs more capacity must receive it and highlight any performance gains on extremely important minority classes like hate speech, whereas for full capacity enhancement, there is always a big overhead.

4.6 Stage 4: Primary Training

In this primary training phase, for seven epochs, the model is fine-tuned using the adaptive-rank LoRA configurations discovered during warmup. For every epoch, the model preserves the rank initialization scheme of sac adaptive ranks r=32 for the top 25% gradient magnitudes and r=16 for the remaining 75%, while keeping other hyperparameters fixed for a fair comparison.

The AdamW optimizer is used with a learning rate of 3×10^{-5} and a batch size of 32. Only the LoRA-injected parameters in the query and value projections are set to update during this stage, leaving the backbone transformer layers frozen in order to ensure parameters' efficiency.

The training pipeline is run on Google Colab NVIDIA A100 GPUs, which provide the computational power for handling large Transformer models and fast processing.

Chapter 5

Results and Discussion

5.1 Results

5.1.1 Evaluation Framework

Our evaluation uses four major metrics to thoroughly analyze performance on the task of hate speech identification where the classes are imbalanced

- **Accuracy**: Proportion of correct predictions among all predictions ($\frac{TP+TN}{TP+TN+FP+FN}$)
- **Precision**: Proportion of true positives among predicted positives ($\frac{TP}{TP+FP}$)
- **Recall**: Proportion of actual positives correctly identified ($\frac{TP}{TP+FN}$)
- **F1-score**: Harmonic mean of precision and recall ($2 \times \frac{Precision \times Recall}{Precision + Recall}$)

We emphasize macro-averaged scores in an effort to address the highly skewed class balance of our data, particularly significant in the hate speech identification domain where minority class performance is practically significant.

Aside from the classification metrics, we also observe the per-epoch training loss (cross-entropy) in order to see the pattern of the learning. It indicates how the model decreases the mistakes over time and it has a steady convergence.

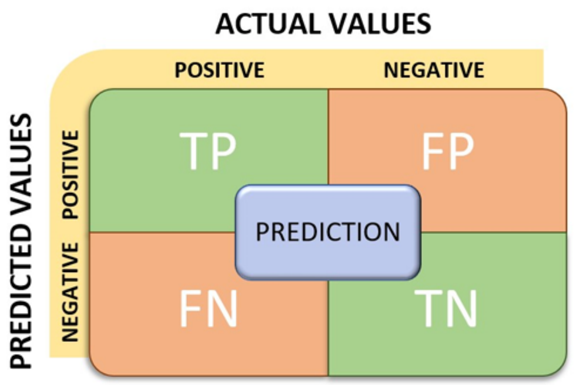

Figure 5.1: Confusion Matrix (adapted from [8])

5.1.2 Model-Specific Performance Analysis

DistilBERT

- Training loss decreased from 0.607 (Epoch 1) to 0.301 (Epoch 7)
- Hate class (Class 0) recall improved from 0.02 to 0.47
- Hate class (Class 0) precision improved from 0.40 to 0.56
- Achieved 90.16% final accuracy with macro F1 of 77.57%

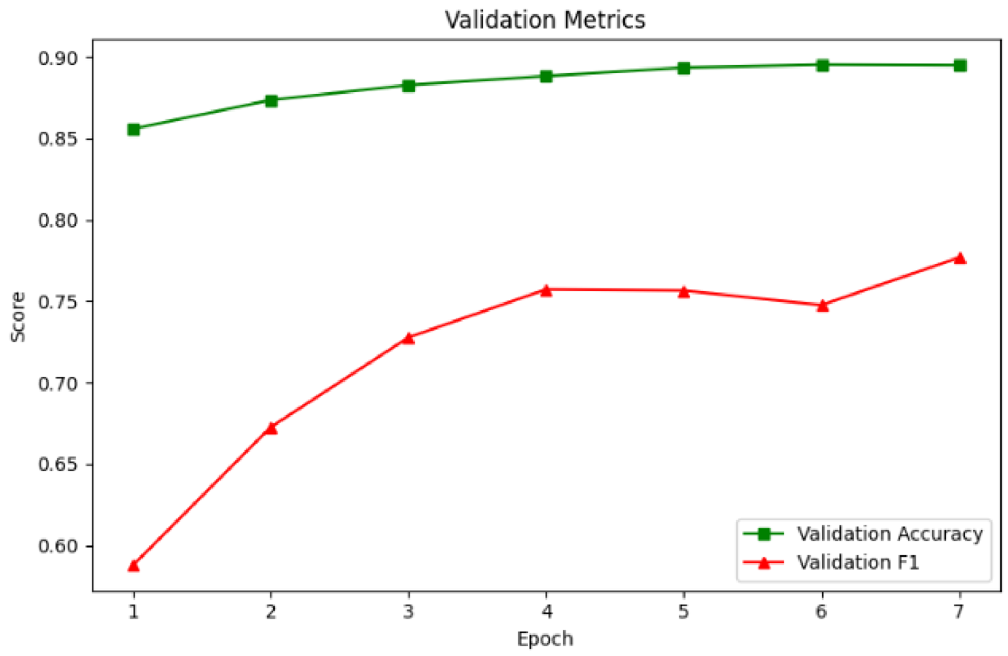

Figure 5.2: DistilBERT training dynamics validation metrics progression. The model demonstrates stable convergence with moderate hate class recall improvement

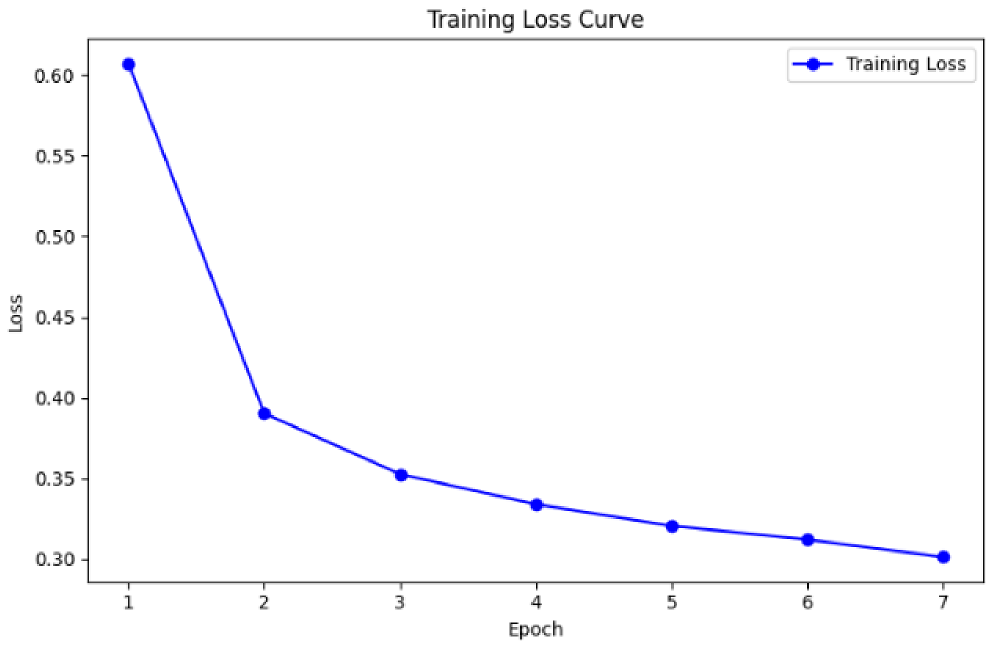

Figure 5.3: DistilBERT training dynamics showing loss decay

RoBERTa

- Training loss decreased from 0.6181 (Epoch 1) to 0.296 (Epoch 7)
- Hate class (Class 0) recall improved from 0.31 to 0.49
- Hate class (Class 0) precision improved from 0.50 to 0.62
- Final test accuracy of 89.35% with macro F1 79.30%

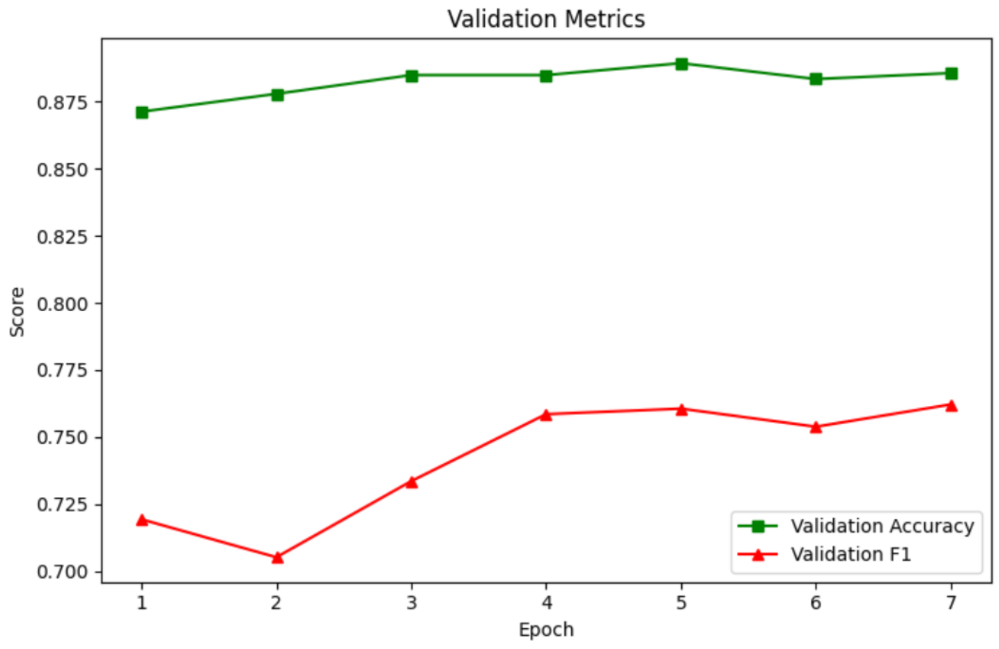

Figure 5.4: RoBERTa validation metrics showing improved hate class precision-recall balance compared to DistilBERT.

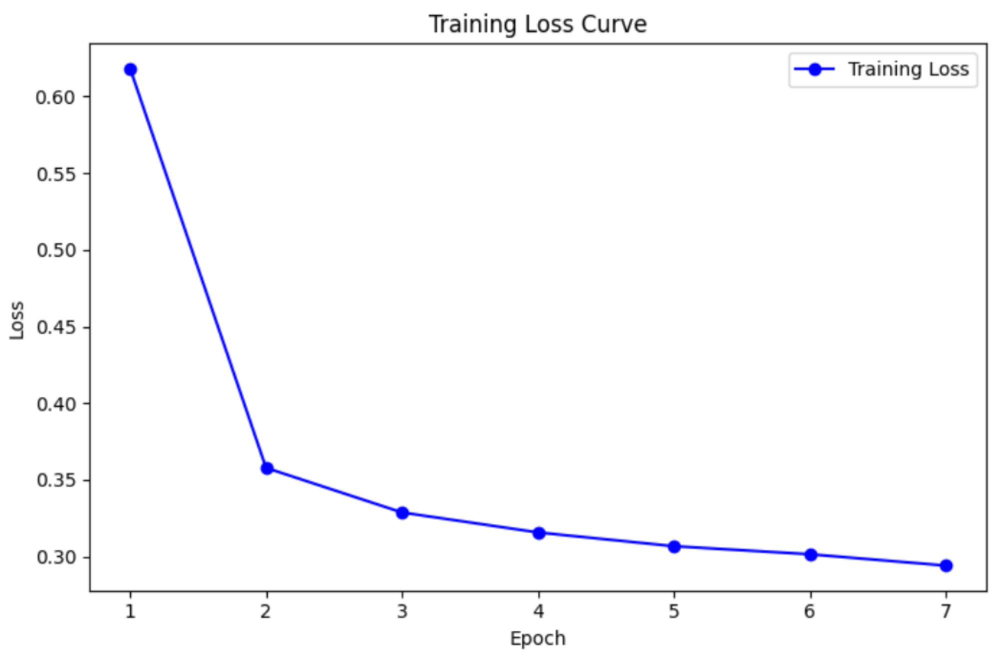

Figure 5.5: RoBERTa training dynamics showing loss decay.

GPT-2

- Training loss decreased from 0.913 (Epoch 1) to 0.329 (Epoch 7)
- Hate class (Class 0) recall improved from 0.02 to 0.33
- Hate class (Class 0) precision improved from 0.24 to 0.59
- Largest absolute accuracy gain (+17.54%) between first and final epochs
- Final test accuracy of 88.24% with macro F1 74.25%

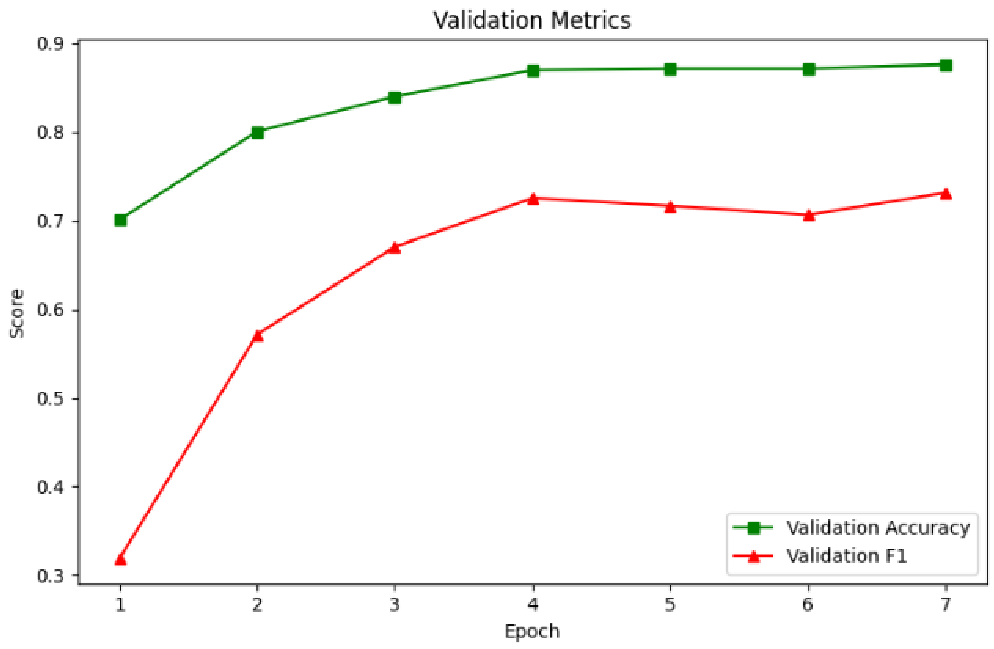

Figure 5.6: GPT-2's training trajectory showing delayed but substantial performance improvements, particularly in offensive class detection.

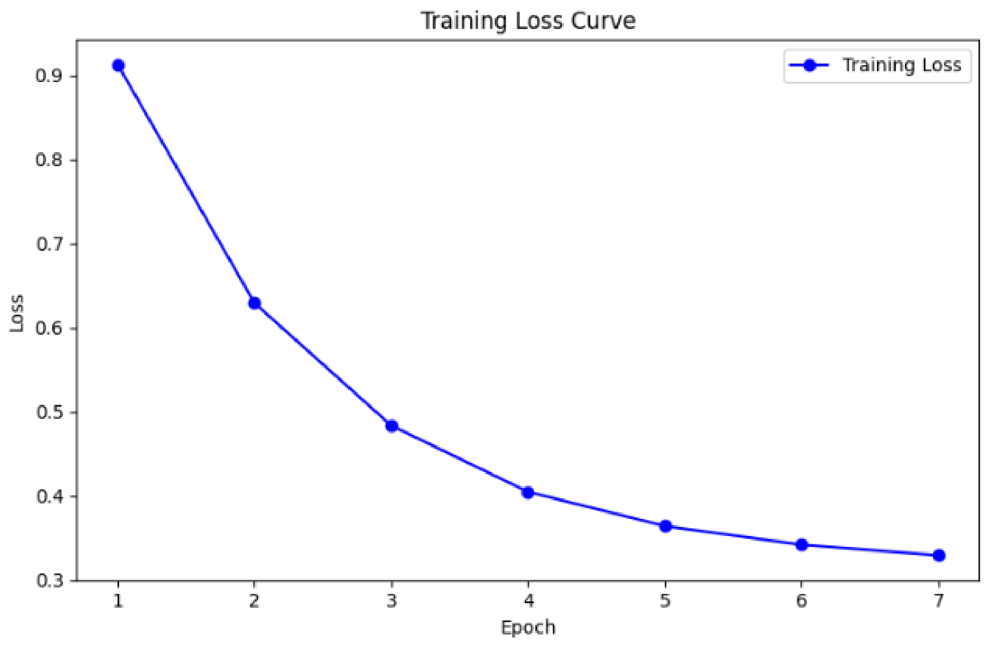

Figure 5.7: GPT-2's training dynamics showing loss decay.

5.1.3 Performance Summary

Table 5.1: Final test set performance across the 3 models

Class	**Metric**	**DistilBERT**	**RoBERTa**	**GPT-2**
Hate	Precision	0.55	0.62	0.59
	Recall	0.45	0.49	0.33
	F1	0.50	0.55	0.42
Offensive	Precision	0.94	0.93	0.90
	Recall	0.94	0.94	0.95
	F1	0.94	0.93	0.93
Neither	Precision	0.88	0.88	0.88
	Recall	0.91	0.92	0.88
	F1	0.89	0.90	0.88
Macro Avg	Precision	0.79	0.81	0.79
	Recall	0.77	0.78	0.72
	F1	0.7757	0.7930	0.7425
	Accuracy	0.9016	0.8935	0.8824

RoBERTa had the best macro F1-score (79.30%) of the three, demonstrating better balance in all the classes. It was the best performer in hate class precision (0.62) and thus the best model to classify the minority class. DistilBERT was close to the top, demonstrating the best accuracy (90.16%) and decent performance in offensive class. Slow convergence in the learning hurt GPT-2 but still left it with a big offensive recall (0.95) to its credit. It lagged behind in the hate class performance and shows a lack of autoregressive modeling for this classification task.

5.2 Comparisons

We enhance the core LoRA adjusting the rank of each layer based on its gradient magnitude (Section 4.4). This enables us to assign the most significant importance to the most significant layers. [20], on the contrary, decomposes the weight both into direction and magnitude but maintains the same rank across all layers.

Our investigation compares three parameter-efficient fine-tuning approaches across different transformer architectures with the same hyperparameters setup and same model setup, but the difference between these approaches are that:

- **Adaptive-rank LoRA**: Adjust ranks to r=32 for layers with gradient norms that exceed the 75th percentile of the gradient values.
- **Fixed-rank LoRA**: Constant rank across all layers (r=16).
- **DoRA**: Decomposes pretrained weights into magnitude and direction components; applies LoRA only to the directional part.

5.2.1 DistilBERT Comparison

Adaptive-Rank LoRA performed better than Fixed-Rank and DoRA in all hate class metrics, achieving near a 0.10 improvement over fixed. It achieved the best macro F1 score of 0.776, demonstrating the performance of adaptive rank assignment. DoRA ranked second, with a minor improvement over fixed-rank in precision as well as macro F1.

Table 5.2: DistilBERT configuration comparison (Test Set)

Class	Metric	Adaptive	Fixed	DoRA
Hate	Precision	0.55	0.53	0.58
	Recall	0.45	0.35	0.38
	F1	0.50	0.42	0.46
Offensive	Precision	0.94	0.93	0.92
	Recall	0.94	0.94	0.94
	F1	0.94	0.94	0.93
Neither	Precision	0.88	0.88	0.87
	Recall	0.91	0.93	0.92
	F1	0.89	0.91	0.89
Macro	F1	0.776	0.755	0.761
	Accuracy	0.902	0.900	0.891

5.2.2 RoBERTa Comparison

RoBERTa demonstrated a consistent pattern: The top-performing result for hate had an F1 score of 0.55 and a macro F1 score of 0.793 which was obtained by Adaptive-Rank LoRA. Fixed-rank fell far short of the hate precision and recall. DoRA performed surprisingly nearly as well as Adaptive but wasn't balanced as a whole, again highlighting the necessity of gradient-aware rank adaptation.

Table 5.3: RoBERTa configuration comparison (Test Set)

Class	**Metric**	**Adaptive**	**Fixed**	**DoRA**
Hate	Precision	0.62	0.57	0.56
	Recall	0.49	0.43	0.45
	F1	0.55	0.49	0.50
Offensive	Precision	0.93	0.90	0.92
	Recall	0.94	0.94	0.95
	F1	0.93	0.92	0.93
Neither	Precision	0.88	0.89	0.91
	Recall	0.92	0.85	0.87
	F1	0.90	0.87	0.89
Macro	F1	0.793	0.761	0.772
	Accuracy	0.894	0.878	0.891

5.2.3 GPT-2 Comparison

In all strategies, GPT-2 achieved the highest accuracy and F1 gains between the first and final epoch. The best two were Adaptive and then DoRA, both achieving an accuracy of 0.882. Adaptive method achieved the highest macro F1 score of 0.743 as well. Fixed-rank received the lowest macro F1 score. So, GPT-2 isn't greatly affected by ranking but improves with adaptive tuning while recalling the minority class.

Table 5.4: GPT-2 configuration comparison (Test Set)

Class	**Metric**	**Adaptive**	**Fixed**	**DoRA**
Hate	Precision	0.59	0.60	0.56
	Recall	0.33	0.29	0.30
	F1	0.42	0.39	0.39
Offensive	Precision	0.90	0.89	0.91
	Recall	0.95	0.95	0.94
	F1	0.93	0.92	0.93
Neither	Precision	0.88	0.88	0.84
	Recall	0.88	0.86	0.90
	F1	0.88	0.87	0.87
Macro	F1	0.743	0.727	0.730
	Accuracy	0.882	0.876	0.882

5.3 Discussion

The performance on DistilBERT, RoBERTa, and GPT-2 demonstrates adaptive rank assignment LoRA to significantly outperform fixed-rank LoRA and DoRA. These advantages are particularly noticeable for the hate speech class, which is the minority class where we need the most focus.

One main reason to why our approach works so well is the fact that it adapts to the data. Looking at the magnitude of the gradient in each layer during the warm-up period, our approach discovers where the additional learning (rank) is necessary. This prevents issues with static rank methods, which would either not update sufficiently for necessary layers (underfit when the rank is too low) or waste capacity or rank on less informative layers (overfit when the rank was too high). As seen in all the models' results, this rank assignment and allocation translated into the highest F1 scores for the hate class and best overall macro F1 performance.

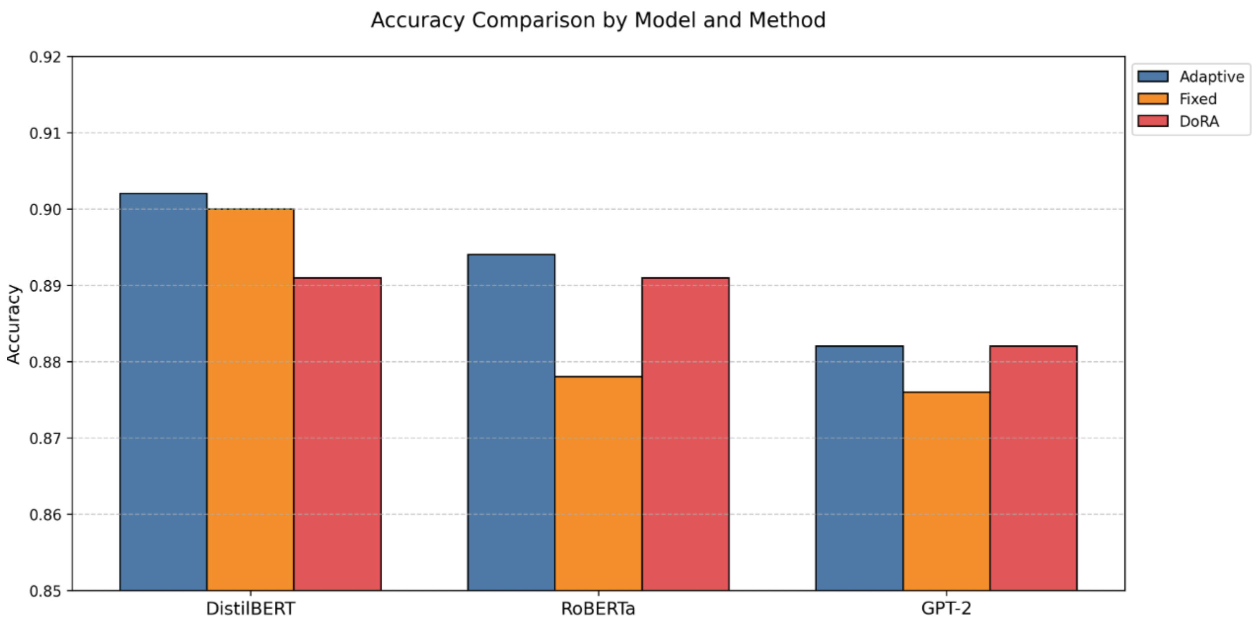

Figure 5.8: Accuracy comparison across models and methods. Adaptive allocation shows consistent performance while fixed-rank excels in RoBERTa due to majority class focus.

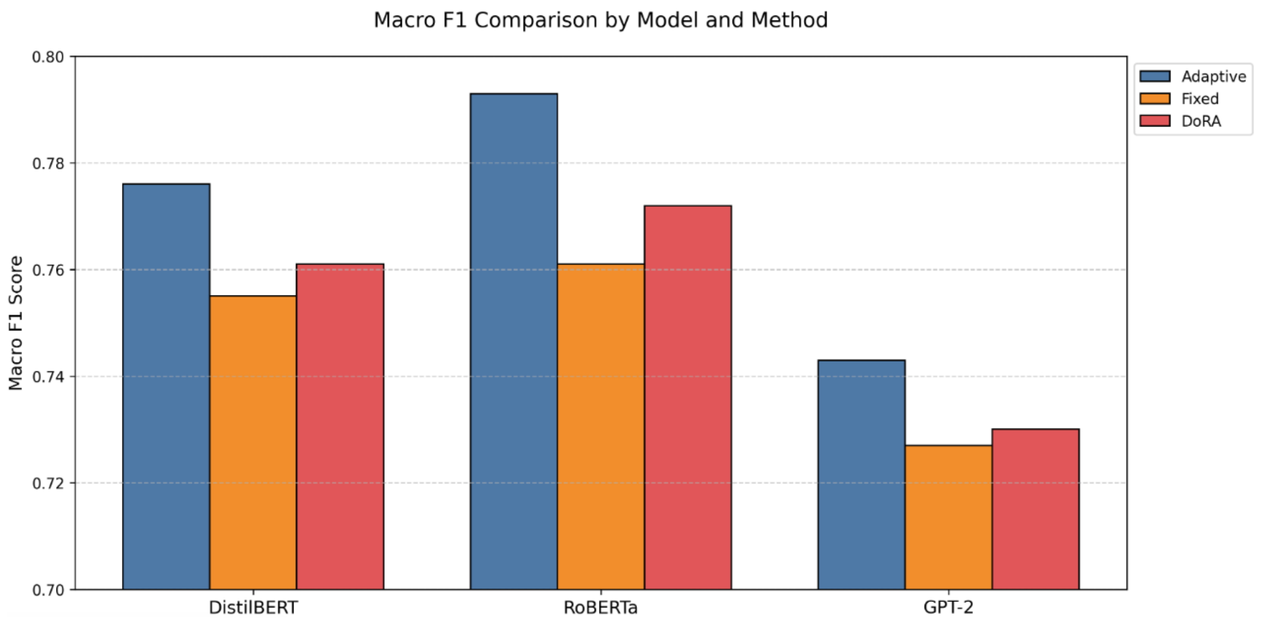

Figure 5.9: Macro F1 scores reveal adaptive methods' superior balance, particularly for DistilBERT and GPT-2. DoRA shows RoBERTa-specific benefits.

DoRA is an improvement over fixed-rank LoRA since it retains the weight magnitude while allowing changes in alternate directions. This minor modification is useful but lacks the adaptive assignments of the rank to layers that are most important. Our findings indicate that although DoRA performs well, its fixed-rank nature makes it less capable of fitting certain tasks in all layers.

Ultimately, our approach maintained high accuracy and good overall performance in all configurations without increasing the number of trainable parameters. This indicates that our approach is effective. Reallocating existing capacity is often more effective than adding more parameters, especially in constrained training environments.

Chapter 6

Conclusion, Recommendations and Future Work

This thesis examined the parameter-efficient fine-tuning (PEFT) of cyberbullying identification using adaptive rank assignment in LoRA. It adapts the rank of the LoRA matrices according to layer-wise magnitudes of the gradients over an early warmup period. This enables the model to distribute the capacity of the model where it can be utilized best, avoiding the primary drawback of the conventional fixed-rank LoRA.

In the background, we went over the fundamentals of transformer models, BERT, DistilBERT, RoBERTa, and GPT-2. We went over the parameter-efficient fine-tuning approaches, especially LoRA and DoRA. This laid the groundwork for what adaptive rank assignment might be utilized to further advance PEFT.

We considered the earlier hate speech and cyberbullying detection methods, including fully fine-tuning approaches (BERT, SBERT, HateBERT) and parameter-efficient approaches (e.g. LoRA) employed in earlier hate speech work. We did not employ DoRA on the cyberbullying or hate speech detection at hand but later added it as another PEFT method, while adding GoRA even later as another adaptation method. We drew inspiration from the gradient-adaptive ranking assignment of GoRA but employed a simpler, more practical approach with a single warmup epoch to implement discrete ranking changes rather than dynamic.

With one warmup epoch, we identified critical layers by observing their gradient magnitudes and giving them a superior ranking (r=32) and others a default ranking (r=16). The approach settled the model's performance and the amount of computing it did by adapting the model to the task.

The results proved our adaptive-rank LoRA was superior in all the experiments, particularly the minority hate speech classification. Overall performance and certain statistics supported the process beating the fixed-rank LoRA process as well as the DoRA process. The improvements in this instance were made with the efficient use of parameters, proving how the adaptive methods with actual indicators in the actual world are a superior and effective method compared to others.

Future work will be towards enhancing the way we assign ranks using GoRA's ranking mechanism. Our approach assigns larger ranks to gradient norms above certain thresholds for the layers, and GoRA GoRA dynamically changes the ranks of the layers on a continuous scale for each training epoch and can be even more detailed. Using the changes in ranks across a space can improve the model even further by enabling it to adapt to changing task requirements during training, not only layer importance.

Bibliography

[1] T. Tita and A. Zubiaga, “Cross-lingual hate speech detection using transformer models,” *https://arxiv.org/pdf/2111.00981*, 2021.

[2] A. Das *et al.*, “Analysis and detection of multilingual hate speech using transformer based deep learning,” *https://arxiv.org/abs/2401.11021*, 2024.

[3] K. Guo *et al.*, “An investigation of large language models for real-world hate speech detection,” *https://arxiv.org/pdf/2401.03346*, 2024.

[4] L. G. G. Charpentier and D. Samuel, “Not all layers are equally as important: Every layer counts bert,” *https://arxiv.org/pdf/2311.02265*, 2023.

[5] Gutiérrez-Batista *et al.*, “Improving automatic cyberbullying detection in social network environments by fine-tuning a pre-trained sentence transformer language model,” *https://link.springer.com/article/10.1007/s13278-024-01291-0*, 2024.

[6] Verma *et al.*, “Benchmarking language models for cyberbullying identification and classification from social-media texts,” *https://aclanthology.org/2022.lateraisse-1.4/*, 2022.

[7] Sihab-Us-Sakib *et al.*, “Cyberbullying detection of resource constrained language fromsocial media using transformer-based approach,” *https://www.sciencedirect.com/science/article/pii/S2949719124000529*, 2024.

[8] D. Ottoson, “Cyberbullying detection on social platforms using large language models,” *https://www.diva-portal.org/smash/record.jsf?pid=diva2*

[9] *M. E.-K. CHUNG, "Developing a fine-tuned transformer model to detect social media hate speech texts,"* http://eprints.utar.edu.my/6896/*, 2024.*

[10] *Mozafari* et al.*, "Hate speechdetection and racial bias mitigation in social media based on bert model,"* https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0237861*, 2020.*

[11] *A. G. d'Sa* et al.*, "Classification of hate speech using deep neural networks,"* https://hal.science/hal-03101938/*, 2020.*

[12] *A. Glazkova* et al.*, "Fine-tuning of pre-trained transformers for hate, offensive, and profane content detection in english and marathi,"* https://arxiv.org/abs/2110.12687*, 2021.*

[13] *B. Ogunleye and B. Dharmaraj, "The use of a large language model for cyberbullying detection,"* https://www.mdpi.com/2813-2203/2/3/38*, 2023.*

[14] *H. B. Zia* et al.*, "Improving zero-shot cross-lingual hate speech detection with pseudo-label fine-tuning of transformer language models,"* https://doi.org/10.1609/icwsm.v16i1.19402*, 2022.*

[15] *S. Mukherjee* et al.*, "Application of transformer-based language models to detect hate speech in social media,"* https://doi.org/10.47852/bonviewJCCE2022010102*, 2021.*

[16] *F. zahra El-Alami* et al.*, "A multilingual offensive language detection method based on transfer learning from transformer fine-tuning model,"* https://doi.org/10.1016/j.jksuci.2021.07.013*, 2022.*

[17] *B. Wei* et al.*, "Offensive language and hate speech detection with deep learning and transfer learning,"* https://arxiv.org/abs/2108.03305*, 2021.*

[18] *Y. Zhao, "An exploration of fine-tuning bert for hate speech analysis,"* https://trepo.tuni.fi/bitstream/handle/10024/155728/Zhao

[19] A. Nasir *et al.*, "Llms and finetuning: Benchmarking cross-domain performance for hate speech detection," *https://arxiv.org/abs/2310.18964*, 2023.

[20] S.-Y. Liu *et al.*, "Dora: Weight-decomposed low-rank adaptation," *https://arxiv.org/pdf/2402.09353*, 2024.

[21] haonan he *et al.*, "Gora: Gradient-driven adaptive low-rank adaptation.," *https://arxiv.org/pdf/2502.12171*, 2025.

[22] A. Vaswani *et al.*, "Attention is all you need," *https://arxiv.org/pdf/1706.03762*, 2023.

[23] J. Devlin *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," *https://arxiv.org/pdf/1810.04805*, 2019.

[24] V. Sanh *et al.*, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *https://arxiv.org/pdf/1910.01108*, 2020.

[25] Y. Liu *et al.*, "Roberta: A robustly optimized bert pretraining approach," *https://arxiv.org/pdf/1907.11692*, 2019.

[26] T. B. Brown *et al.*, "Language models are few-shot learners," *https://arxiv.org/pdf/2005.14165*, 2020.

[27] V. B. Parthasarathy *et al.*, "The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities," *https://arxiv.org/abs/2408.13296*, 2024.