As per UGC guidelines an electronic bar code is provided to secure your paper

International Journal for Modern Trends in Science and Technology
Volume 12, Issue 06, pages 385-393.
ISSN: 2455-3778 online
Available online at: http://www.ijmtst.com/vol12issue06.html
DOI: https://doi.org/10.5281/zenodo.20739582

A Cost-Efficient Performance Evaluation Framework for Cloud-Based Software Requirements Classification Using Parameter Efficient Fine Tuning

Sonal | Ashish Kumar Pandey

Department of Computer Science and Engineering , IET Dr. R.M.L. Avadh University , Ayodhya, Uttar Pradesh, India.

To Cite this Article
Sonal & Ashish Kumar Pandey (2026). A Cost-Efficient Performance Evaluation Framework for Cloud-Based Software Requirements Classification Using Parameter Efficient Fine Tuning. International Journal for Modern Trends in Science and Technology, 12(06), 385-393. https://doi.org/10.5281/zenodo.20739582

Article Info
Received: 02 June 2026; Revised: 22 June 2026; Accepted: 26 June 2026.

Copyright © The Authors ; This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

KEYWORDS

Software requirements classification, requirements engineering, cost-efficient machine learning, cloud computing, parameter-efficient fine-tuning, BERT, NLP

ABSTRACT

Cloud-based machine learning has emerged as a promising approach for automating software requirements classification. In most cases, however, fine-tuning transformer-based models is an expensive process that requires substantial computing resources and costs in production in cloud-based systems. This study proposes a cost- efficient performance evaluation framework for cloud-based software requirements classification using parameter-efficient fine-tuning techniques. The framework assesses both classification accuracy and cloud resource efficiency. This study evaluates the performance of Full Fine-Tuning (FFT), Adapter-Based Fine-Tuning and Low-Rank Adaptation (LoRA) under identical cloud deployment conditions using BERT-base model in a GPU-based environment. In the adopted BERT-base configuration, only 0.8% of the total parameters are modified with LoRA while 3.9% are modified with adapter-based tuning. The results further show that these techniques decrease training cost, GPU memory consumption and computational overhead by over 50% and maintain almost the same prediction performance. The accuracy difference between different experiments is within 1%. Overall, the results indicate that parameter-efficient fine-tuning provides a scalable, cost-effective and feasible solution to deploy an industrial-scale software requirements classification system in cloud environment.

ABBREVIATIONS

FFT	Full Fine-Tuning
ABFT	Adapter-Based Fine-Tuning
LoRA	Low-Rank Adaptation
BERT	Bidirectional Encoder Representations from Transformers
PEFT	Parameter-Efficient Fine-Tuning
GPU	Graphics Processing Unit
NLP	Natural Language Processing
NICE	Non Functional Identification and Classification Engineering

1. INTRODUCTION

Software requirement classification is a fundamental task in requirements engineering, enabling automated identification of functional and non-functional requirements, quality attributes, and system constraints. As modern software projects generate large volumes of natural-language requirements, accurate automated classification is essential for improving development efficiency, traceability, and requirement consistency. Recent advances in transformer-based language models, particularly BERT and its variants, have significantly improved performance in requirement classification by capturing contextual semantics beyond traditional machine-learning approaches such as Support Vector Machines and Random Forests [1]–[3]. However, these models typically rely on full parameter fine-tuning, which updates hundreds of millions of parameters and results in high computational cost, large memory consumption, and prolonged training time, factors that become critical in cloud-based deployments due to GPU pricing and scalability constraints [2], [4].

PEFT has emerged as an alternative adaptation strategy in which most pretrained weights remain frozen while only a small subset of task-specific parameters is trained [3], [5], [6]. Methods such as LoRA learn low-rank weight updates, whereas adapter-based approaches introduce lightweight trainable modules within transformer layers [6]–[8]. These techniques substantially reduce training overhead while preserving predictive capability across many NLP tasks.

Despite their success in general NLP applications, the adoption of PEFT in software requirements engineering remains limited. Existing requirement classification studies primarily focus on predictive accuracy and rarely examine deployment feasibility or operational cost in cloud environments [9]–[12],[19]. In particular, systematic evaluation of the trade-off between classification performance and resource utilization such as training time, memory footprint, and compute cost has not been sufficiently explored in the software engineering literature [13], [14].

To address this gap, this work presents a deployment-oriented comparison of fine-tuning strategies under controlled cloud infrastructure. The study evaluates both predictive performance and economic efficiency by analyzing computational resource usage alongside classification accuracy.

The main contributions of this paper are:

1. To design and implement a cost- efficient performance evaluation framework for cloud based software requirements classification using parameter efficient fine tuning.
2. To evaluate and compare Full Fine-Tuning, Adapter-Based Fine-Tuning, and LoRA in terms of predictive performance and cloud resource efficiency.
3. To analyze the trade-off between cloud cost and accuracy.

The remainder of this paper is organized as follows: Section 2 presents related work, Section 3 describes the methodology, Section 4 explains the experimental setup, Section 5 presents the results and discussion and Section 6 concludes the paper.

II. RELATED WORKS

A. Existing Studies

The classification of requirements into functional requirements (FRs) and non-functional requirements (NFRs) is a basic requirement engineering problem that involves the automatic classification of software requirements. In the early studies, the requirement classification was performed using traditional Machine learning and handcrafted linguistic features, statistical representations, and rule-based techniques [12] and [13] and [17]. While these techniques were able to minimize manual work, they did not perform well with respect to the ability to capture semantic meaning in a context, which led to sub-optimal generalization across heterogeneous software projects.

As a result of the deep learning development, a number of researchers turned to using neural network architectures, as well as language models based on transformers, for software requirement analysis. Contextual representations of requirement statements were used to improve classification accuracy using models like BERT and its variants [1, 3, 5]. Such methods proved to be effective in capturing the semantic dependencies and domain-specific terminology, resulting in improved discrimination between functional and non-functional requirements.

Most of the previous studies, however, have concentrated mainly on achieving best predictive performance via the FFT approach, where all the model

parameters are fine-tuned in the training process. Although FFT has a high accuracy of classification, it has high requirements for computing resources, memory, and training time, and it is difficult to be applied in practical applications in resource-limited environments.

The challenges have led to the development of techniques for NLP that focus on overcoming these issues, such as PEFT. Adapter-based methods like Adapters and LoRA only modify a few parameters in the model while preserving the rest of the model weights to remain unchanged [2]–[9]. These methods significantly lower the storage space and complexity of training, while retaining competitive predictive accuracy. In particular, LoRA has become a popular choice because it offers a good balance between efficiency and accuracy, while Adapter-based methods enable incremental learning and multi-task applications [3], [6-7], [14].

B.Comparative Analysis of Previous Approaches

Table 1 summarizes the major approaches reported in the literature for requirement classification and transformer fine-tuning.

Table I: Comparative Analysis of Previous Approaches

Approach	Reference	Advantages	Limitations
Traditional Machine Learning	[12],[13], [17]	Simple implementation, low computational requirements	Limited semantic understanding and poor generalization
Deep Learning Models	[1],[5]	Improved feature extraction and classification performance	Require large training datasets and computational resources
Transformer Models with FFT	[1],[3],[5]	State-of-the-art predictive accuracy and contextual understanding	High memory usage, long training time, and large storage requirements
Adapter-Based PEFT	[6],[8],[14]	Reduced trainable parameters and modular adaptation	Additional adapter modules may increase inference complexity
LoRA-Based PEFT	[3],[7],[13]	Significant reduction in trainable parameters with minimal performance loss	Performance may vary depending on rank selection and task characteristics

The comparative analysis shows that the transformer-based models achieve better results than traditional methods across all tasks related to requirement classification; however, the majority of studies focus on predictive metrics such as accuracy, precision, recall, and F1-score. Limited effort has been made to measure the efficiency parameters of the operation, such as the length of training time, the amount of memory used, parameter reduction and computational cost.

In addition, applying the PEFT techniques to the software requirements classification issue is relatively underexplored. Through these observations, there is a need for a holistic evaluation framework that considers both predictive accuracy and computational efficiency.

These observations motivate the need for a comprehensive evaluation framework that simultaneously considers predictive effectiveness and computational efficiency, thereby facilitating the practical application of transformer model in software engineering environments.

III. METHODOLOGY

A. Proposed cost- efficient performance evaluation framework

This framework follows a systematic pipeline designed to balance predictive accuracy with cloud resource consumption. Unlike traditional evaluations that focus solely on F1-scores, this framework treats operational overhead as a primary evaluation dimension.

Figure 1 illustrates the overall workflow of the proposed framework. The process begins with dataset collection and preprocessing, where raw requirement statements are tokenized and formatted for transformer compatibility. The core of the methodology involves training a BERT-base model using three distinct strategies: Full Fine-Tuning (FFT), Adapter-Based Fine-Tuning (ABFT), and Low-Rank Adaptation (LoRA). Each strategy is subjected to a dual-metric evaluation consisting of:

1. Predictive Metrics: Focusing on Accuracy and F1-score to ensure requirements are correctly categorized.

2. Operational Metrics: Analyzing training time, GPU memory allocation, and estimated cloud training cost ($).

The final stage of the workflow involves a Pareto-optimal selection process, identifying the method that provides the highest accuracy gain per dollar of cloud expenditure.

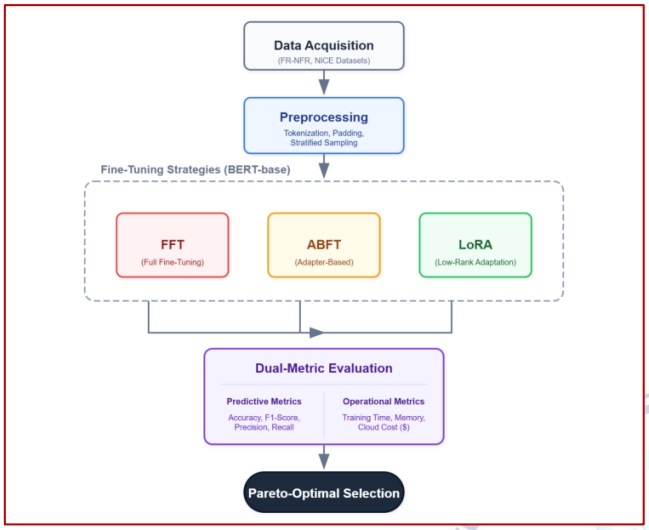

Fig. 1: Proposed Cost- Efficient Performance Evaluation Framework

B. Parameter-Efficient Fine-Tuning (PEFT)

PEFT adapts a pretrained transformer while keeping most backbone weights frozen and updating only a small set of task-specific parameters [2], [3],[20].

Let P = total model parameters and P_t = trainable parameters

Trainable Percentage=$\frac{p_t}{p}$×100

In FFT, P_t=P

In PEFT, $P_t \ll P$.

For a linear layer with weight matrix $W \in R^{d_{out}*d_{in}}$

- FFT update cost: $O(d_{out} * d_{in})$
- LoRA update cost: $O(r(d_{out} + d_{in}))$
- where $r \ll \min(d_{out}, d_{in})$

This significantly reduces computational overhead.

The study evaluates two PEFT strategies: ABFT and LoRA.

i. ABFT

ABFT adapters are lightweight bottleneck layers inserted inside each transformer block after the multi-head attention and feed-forward sublayers.

For hidden representation h:

$$h' = h + W_{up}\sigma(W_{down}h)$$

where, $W_{down} \in R^{d*b}$, $W_{up} \in R^{b*d}$ and b is the bottleneck size ($b \ll d$).

Only adapter parameters are updated while pretrained weights remain frozen.

ii. LoRA

LoRA injects trainable low-rank matrices into attention projection layers.

The adapted weight matrix is:

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

where: $W_0 \in R^{d_{out}*d_{in}}$, $B \in R^{d_{out}*r}$ and $A \in R^{r*d_{in}}$ are trainable matrices, r is the rank hyperparameter, α is a scaling factor.

LoRA reduces memory usage and training time while preserving predictive capability.

C. Datasets and Preprocessing

Datasets

This study used a publicly available dataset. The primary dataset is FR–NFR dataset consisting of 1,145 requirements collected from 15 software projects and 5 domains, including 722 FR and 423 NFR instances[15]. NFRs are annotated into quality attributes such as performance, security, usability, and maintainability.

Extended FR–NFR dataset with 6,118 requirements. It is used for larger-scale binary classification evaluation.It contains approximately 3,964 FR and 2,154 NFR instances[16].

The PROMISE-relabeled-NICE dataset contains approximately 622 unique requirements. Since a single requirement may belong to multiple NFR categories, the total number of labeled instances increases to approximately 1,250 for multi-class NFR classification[18].

The inclusion of the Extended FR–NFR and NICE datasets improves robustness, supports external validation, and demonstrates the applicability of the proposed PEFT methods across both binary and multi-class requirement classification scenarios.

Preprocessing

- The preprocessing stage transformed raw requirement statements into a format suitable for BERT-based classification.

- All statements were processed using the pretrained bert-base-uncased tokenizer, which automatically converts text to lowercase.
- The statements were tokenized using the WordPiece method
- Special tokens [CLS] and [SEP] were added at the beginning and end of each sequence, respectively.
- A maximum sequence length of 256 tokens was used.
- Shorter sequences were padded using [PAD] tokens, while longer sequences were truncated.
- Attention masks were also generated, where 1 represents actual tokens and 0 represents padded tokens.
- Stop-word removal was not applied because transformer models rely on complete sentence context for better semantic understanding.
- Finally, the dataset was split into training, validation, and testing sets in a 70:15:15 ratio using stratified sampling, with random_state = 42 to ensure reproducibility.

After preprocessing, the FR–NFR dataset of 1,145 instances was divided into 802 training samples, 172 validation samples, and 171 test samples. Similarly, the Extended FR–NFR dataset (6,118 samples) was split into 4,283 training, 918 validation, and 917 test instances, while the NICE dataset (1,250 samples) was divided into 875 training, 188 validation, and 187 test instances using stratified sampling.

IV. Experimental Setup

In this section, Training Configuration, Experimental Comparison Strategy and Evaluation Metrics are described in details.

A.Training Configuration

Experiments were conducted using Python 3.10 with PyTorch and Scikit-learn libraries on an Ubuntu 22.04 environment. Training was performed on an 8-core Intel Xeon CPU with 32 GB RAM and an NVIDIA Tesla T4 GPU having 16 GB memory. The models were trained using a batch size of 16, learning rate of 2×10^{-5}, AdamW optimizer and 5 epochs. Training time was measured using system timestamps, while peak memory consumption was obtained from maximum GPU allocation during execution. Cloud training cost was estimated using an assumed GPU pricing of $2 per hour for an NVIDIA Tesla T4 instance.

B. Experimental Comparison Strategy

All models were evaluated under the same experimental conditions to ensure a fair comparison. The transformer-based methods, including FFT, ABFT, and LoRA, used the same pretrained BERT model, hyperparameters, preprocessing steps, and dataset splits. Similarly, the SVM and RF baselines were trained and tested on the same data partitions. Therefore, differences in performance and efficiency are mainly due to the fine-tuning strategy rather than external factors.

C. Evaluation Metrics

(i). Predictive Metrics

To evaluate classification effectiveness, standard supervised learning performance measures are used.

- **Accuracy** measures the proportion of correctly classified requirements among all predictions.

$$\textbf{Accuracy} = \frac{\textbf{TP} + \textbf{TN}}{\textbf{TP} + \textbf{TN} + \textbf{FP} + \textbf{FN}}$$

- **Precision (P)** indicates how many requirements predicted as a specific class actually belong to that class.

$$\textbf{P} = \frac{\textbf{TP}}{\textbf{TP} + \textbf{FP}}$$

- **Recall (R)** measures the model's ability to identify all relevant instances of a class.

$$\textbf{R} = \frac{\textbf{TP}}{\textbf{TP} + \textbf{FN}}$$

- **F1-Score** is the harmonic mean of precision and recall, providing balanced evaluation under class imbalance.

$$\textbf{F1 Score} = 2 * \frac{\textbf{P} * \textbf{R}}{\textbf{P} + \textbf{R}}$$

Where: TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives.

(ii). Operational Metrics or efficiency metrics

This metrics evaluate deployment feasibility in cloud environments.

- **Training Time** : It is computed as the difference between the end time and the start time of the model training process

$$T_{train(minutes)} = t_{end} - t_{start}$$

- **Peak Memory Usage**: maximum GPU memory consumed during training.

$$M_{peak} = \max M(t), t \in [0, T_{train}]$$

- **Estimated Cloud Training Cost**: this cost was computed by multiplying the total training time in hours with the hourly cost of the GPU instance (P_{gpu}).

$$C_{train} = \frac{T_{train(minutes)}}{60} \times P_{gpu}$$

V. RESULTS AND DISCUSSION

This section presents the experimental results obtained from the proposed approach and provides a comprehensive analysis of its performance. The experimental findings are first presented, followed by a detailed analysis and discussion of the results.

A. Classification Performance Comparison

Table II compares traditional machine learning methods, FFT, and the proposed PEFT methods in terms of predictive performance and cloud efficiency.

Table II: Performance and Cloud Efficiency Comparison

Method	Accuracy (%)	F1-Score	Training Time (min)	Peak Memory (GB)	Estimated Cloud Training Cost ($)
SVM	78.2	0.76	12	2	0.4
RF	80.1	0.78	15	4	0.5
FFT	91.0	0.90	120	16	4.0
ABFT	90.0	0.89	55	9	1.8
LoRA	90.5	0.90	45	7	1.5

The transformer-based models improve accuracy by approximately 10–13% compared to traditional machine learning methods. FFT achieves the highest accuracy of 91.0%, while LoRA attains nearly the same performance with 90.5% accuracy. The difference in accuracy between PEFT methods and FFT remains below 1%, indicating that most task-specific knowledge can be captured through partial parameter updates.

In addition, PEFT methods substantially reduce computational cost. Compared to FFT, ABFT reduces the estimated cloud training cost from $4.0 to $1.8, while LoRA further reduces it to $1.5, making it the most cost-efficient approach among the transformer-based models.

B. Training Efficiency and Resource Utilization

The efficiency metrics demonstrate substantial computational savings using PEFT methods.

- Training time is reduced by approximately 54% for Adapter-Based Fine-Tuning and 62.5% for LoRA compared to Full Fine-Tuning.
- Peak memory usage decreases from 16 GB in FFT to 9 GB for Adapter-Based Fine-Tuning and 7 GB for LoRA.
- Estimated cloud training cost is reduced from $4.0 in FFT to $1.8 for Adapter-Based Fine-Tuning and $1.5 for LoRA, corresponding to cost reductions of approximately 55% and 62.5%, respectively.

These findings confirm that updating only a small subset of model parameters can significantly reduce computational resource usage without causing major performance loss.

C. Cost vs. Accuracy Trade-Off

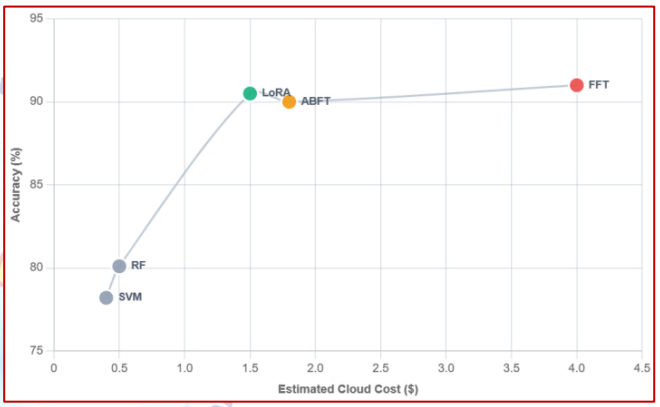

Fig. 2: Trade-off analysis between model accuracy and estimated cloud training costs

The comparative analysis presented in Fig.2 illustrates the trade-off between predictive performance and computational cost across the evaluated methods. Full Fine-Tuning achieves the highest accuracy (91.0%), but it also requires the longest training time (120 minutes), highest memory usage (16 GB), and highest estimated cloud cost ($4.0), which limits its practicality for large-scale or resource-constrained deployments.

In contrast, LoRA and Adapter-Based Fine-Tuning achieve performance close to FFT while requiring substantially lower training time, memory, and cost. LoRA attains 90.5% accuracy with only 45 minutes of training time, 7 GB memory usage, and an estimated cost of $1.5, making it the most cost-efficient transformer-based approach.

Traditional machine learning methods such as SVM and RF remain computationally inexpensive, but their lower predictive accuracy makes them less suitable for

complex requirement classification tasks. Overall, parameter-efficient fine-tuning provides the best balance between performance and operational cost.

D. Efficiency Gain over Full Fine-Tuning

Table III indicates that the LoRA approach provides the most favorable balance between predictive performance and operational cost. LoRA achieves an accuracy of 90.5%, which is only 0.5% lower than Full Fine-Tuning, while reducing the estimated cloud training cost from $4.0 to $1.5, corresponding to a cost reduction of 62.5%. The adapter-based approach also demonstrates strong efficiency, reducing the cost from $4.0 to $1.8, which represents a 55% reduction, with only a 1.0% decrease in accuracy.

These findings confirm that PEFT methods, especially LoRA, can substantially improve computational efficiency without significantly affecting classification performance.

Table III: Efficiency Gain over FFT

Method	Accuracy Drop vs FFT	Cost Reduction
Adapter	−1.0%	55%
LoRA	−0.5%	62.5%

DISCUSSION

This section interprets the experimental findings and analyzes their implications for cost- efficient cloud-based SRC.

A. Classification Performance

The experimental results indicate that PEFT methods achieve performance comparable to FFT. As shown in Table II, Full Fine-Tuning attains an accuracy of 91.0% (F1 = 0.90), while LoRA achieves 90.5% (F1 = 0.90) and Adapter-Based Fine-Tuning achieves 90.0% (F1 = 0.89). Thus, the maximum performance difference between PEFT and FFT is only 1.0% in accuracy and 0.01 in F1-score, confirming that updating only a small subset of parameters is sufficient for adapting transformer models to requirement classification tasks.

In contrast, traditional machine learning methods such as SVM (78.2%) and RF (80.1%) perform substantially worse, highlighting the importance of contextual semantic representations learned by transformer architectures for capturing requirement intent and quality attributes. These findings suggest that PEFT retains most of the representational power of FFT while significantly reducing the number of trainable parameters.

B. Resource Utilization and Cost Analysis

Significant differences emerge when operational efficiency is considered. Full Fine-Tuning requires 120 minutes of training time, 16 GB of peak GPU memory, and an estimated cloud training cost of $4.0. In comparison, the PEFT methods require considerably fewer computational resources, as shown in Table IV.

Table IV: Resource Utilization and Cost Analysis

Method	Training Time	Memory	Cost	Cost Reduction vs FFT
Adapter	55 min	9 GB	$1.8	55%
LoRA	45 min	7 GB	$1.5	62.5%

LoRA provides the highest efficiency, reducing training time by approximately 62.5%, memory usage by about 56%, and cloud training cost by 62.5%, while maintaining nearly identical predictive performance. Adapter-Based Fine-Tuning shows slightly higher resource usage than LoRA, but still reduces training cost by more than half and offers modular extensibility, which can be beneficial in multi-task or incremental learning environments.

The cost vs. accuracy trade-off shown in Fig. 2 demonstrates that PEFT methods operate near the Pareto-optimal frontier, offering large efficiency gains with minimal loss in predictive performance.

C. Limitations and Challenges

Despite the promising findings, several limitations remain. First, the experiments rely on publicly available datasets, which may not fully represent industrial variability such as domain-specific terminology, changing requirement styles, and organizational writing practices. As a result, performance in real-world environments may differ.

Second, the cost model is based on fixed GPU pricing and does not include storage, networking, or cloud-provider-specific billing policies. Therefore, the reported costs should be interpreted as relative measures of efficiency rather than exact deployment expenses.

Third, the study focuses on single-label classification

using a single transformer backbone. Future work should investigate PEFT in multi-label, hierarchical, cross-domain, and continual learning scenarios to further evaluate its generalizability.

Overall, the findings demonstrate that PEFT methods provide classification performance close to FFT while reducing cloud training cost by approximately 55–62.5%, making them practical and economically viable solutions for real-world software requirement classification systems.

VI. Conclusion

This paper presented a cost- efficient comparative evaluation of FFT and PEFT methods for cloud-based software requirement classification. The study compared FFT, ABFT, and LoRA under identical deployment conditions using transformer models. Experimental results show that PEFT methods achieve performance close to Full Fine-Tuning, with LoRA obtaining 90.5% accuracy (F1 = 0.90) and Adapter-Based Fine-Tuning achieving 90.0% accuracy (F1 = 0.89), closely matching the 91.0% accuracy of FFT.

At the same time, PEFT substantially improves operational efficiency. LoRA reduces training time by approximately 62.5%, peak memory usage by about 56%, and estimated cloud training cost by 62.5%, while Adapter-Based Fine-Tuning reduces cost by approximately 55%. These findings confirm that updating only a small subset of parameters is sufficient for effective transformer adaptation in software requirement classification tasks.

The study demonstrates that PEFT enables scalable and economically viable deployment of transformer-based models in practical requirements engineering workflows. By jointly evaluating predictive performance and operational cost, this work bridges the gap between research-oriented accuracy evaluation and real-world deployment feasibility.

Future research will investigate multi-label and hierarchical requirement classification, continual learning under evolving requirements, cross-domain generalization, and explainable AI integration to improve transparency and practitioner adoption.

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

References

[1] S. Eltahier, O. Dawood, and I. Saeed, "BERT fine tuning for software requirement classification: Impact of model components and dataset size," Information, vol. 16, no. 11, p. 981, Nov. 2025.

[2] L. Wang, S. Chen, L. Jiang, S. Pan, R. Cai, S. Yang, and F. Yang, "Parameter-efficient fine-tuning in large language models: A survey of methodologies," Artificial Intelligence Review, vol. 58, Art. no. 227, 2025.

[3] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," arXiv preprint arXiv:2106.09685, Jun. 2021.

[4] S. Nwaiwu, "Parameter-efficient fine-tuning for low-resource text classification: A comparative study of LoRA, IA³, and ReFT," Frontiers in Big Data, 2025.

[5] G. Xanthopoulou, M. Siavvas, I. Kalouptsoglou, D. Kehagias, and D. Tzovaras, "Software Requirements Classification: From Bag-of-Words to Transformer," 2024.

[6] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E. V. Chioasca, and R. T. Batista-Navarro, "Natural language processing for requirements engineering: A systematic mapping study," ACM Computing Surveys, vol. 54, no. 3, pp. 1–41, 2021.

[7] J. Frattini, M. Unterkalmsteiner, D. Fucci, and D. Méndez Fernández, "NLP4RE Tools: Classification, Overview and Management," in Handbook on Natural Language Processing for Requirements Engineering, A. Ferrari and G. Ginde, Eds. Cham, Switzerland: Springer, 2025, pp. 357–380.

[8] Gu, Jiancheng & Yuan, Jiabin & Cai, Jiyuan & Xianfa, Zhou & Fan, Lili, "La-LoRA: Parameter-efficient fine-tuning with layer-wise adaptive low-rank adaptation", Neural networks : the official journal of the International Neural Network Society. 194. 108095. 10.1016/j.neunet.2025.108095.

[9] Justine Nakirija, "Enhancing Software Requirements Classification: Integrating Recurrent Neural Networks and Natural Language Processing for Managing Structural Complexity", Int J Intell Syst Appl Eng, vol. 12, no. 4, pp. 3110–3121, 2024.

[10] S. Lei, Y. Hua, and S. Zhihao, "Revisiting Fine-Tuning: A Survey of Parameter-Efficient Techniques for Large AI Models," Preprints, 2025.

[11] Rizwana Batool, Ayesha Naseer, Ayesha Maqbool, Maemoona Kayani, "Automated categorization of software security requirements: an NLP and ML based approach", Vol. (0123456789), Requirements Engineering, 2025.

[12] Y. Ding, L. Han, N. Zhang, X. Liu, D. H. Yang, and H. Chen,"Parameter-Efficient Fine-Tuning of Pre-trained Language Models:A Survey," arXiv preprint arXiv:2303.15647, 2023

[13] O. Razuvayevskaya, B. Wu, J. A. Leite, F. Heppell, I. Srba, C. Scarton, K. Bontcheva, and X. Song, "Comparison between parameter-efficient techniques and full fine-tuning: A case study on multilingual news article classification," PLOS ONE, vol. 19, no. 5, Art. no. e0301738, May 2024, doi: 10.1371/journal.pone.0301738.

[14] G. Pu, A. Jain, J. Yin, and R. Kaplan, "Empirical Analysis of the Strengths and Weaknesses of PEFT Techniques for LLMs," presented at the Workshop on Understanding Foundation Models, ICLR 2023.

[15] F. Dalpiaz, D. Dell'Anna, F. B. Aydemir, and S. Çevikol, "Requirements classification with interpretable machine learning

and dependency parsing," in Proc. 27th IEEE Int. Requirements Engineering Conf. (RE), Jeju Island, South Korea, 2019, pp. 142–152.

[16] T. Hey, J. Keim, A. Koziolek, and W. F. Tichy, "NoRBERT: Transfer learning for requirements classification," in Proc. IEEE Int. Requirements Engineering Conf. (RE), Zurich, Switzerland, 2020, pp. 169–179.

[17] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, "Parameter-Efficient Fine-Tuning for Large Models: A Comprehensive Survey," arXiv preprint arXiv:2403.14608, Mar. 2024.

[18] G. Rejithkumar and P. R. Anish, "NICE: Non-functional requirements identification, classification, and explanation using small language models," in Proc. 47th Int. Conf. Software Engineering (ICSE), Ottawa, ON, Canada, 2025.

[19] A. Ferrari and G. Ginde, "NLP for Requirements Engineering Overview," in Handbook on Natural Language Processing for Requirements Engineering. Cham, Switzerland: Springer, 2025.

[20] Y. Xiong, Z. Liu, X. Wang, H. Zhang, and Y. Fu, "LoRaDA: Low-Rank Direct Attention Adaptation for Efficient LLM Fine-Tuning," in Findings of the Association for Computational Linguistics: EMNLP 2025, 2025, pp. 12638–12655.