Parameter-Efficient Fine-Tuning for Sentiment Analysis in Foundation Models

Zayaan Khan
Computer Science
University of Surrey
Guildford, UK
zk00332@surrey.ac.uk

Rita San
Computer Science
University of Surrey
Guildford, UK
rs02004@surrey.ac.uk

Steven Thomas
Computer Science
University of Surrey
Guildford, UK
st01634@surrey.ac.uk

Joel D'Souza
Computer Science
University of Surrey
Guildford, UK
jd01606@surrey.ac.uk

Abstract—**Large language models (LLMs) are increasingly used for text generation and semantic understanding, yet fine-tuning them remains computationally expensive. Low-Rank Adaptation (LoRA) addresses this by freezing pre-trained weights and training low-rank matrices, but its performance relies heavily on hyperparameter configuration. This paper proposes optimizing LoRA hyperparameters using a Real-Coded Genetic Algorithm (RCGA) with BLX-α crossover. We fine-tuned a DistilBERT model on the Emotion dataset, comparing our evolutionary approach against a Random Search baseline. While Random Search achieved a best accuracy of 87.80%, our RCGA method converged to a superior accuracy of 90.55% within 100 evaluations. Robustness testing across multiple seeds confirmed the stability of the evolutionary approach. Our findings indicate that adapting both Attention and Feed-Forward modules with a higher Rank ($r = 16$) is critical for maximizing performance on this sentiment analysis task.**

I. Introduction

Emotion classification in text using sentiment analysis is a core challenge within natural language processing, with this downstream task being applied to mental health monitoring, social media analysis and customer service automation such that being able to accurately understand and classify emotional expressions in short text such as tweets, enables AI models to be able to better respond to user's needs, specifically in critical scenarios such as detecting possible signals of a distressed user on such social platforms [1]. Foundation models have owed their success to sophisticated pre-training objectives and huge model parameters. Foundation models can effectively capture knowledge from massive amounts of labeled and unlabelled data since the rich knowledge is implicitly encoded in the huge number of parameters that benefits a variety of downstream tasks, demonstrated via experimental verification and empirical analysis. Due to the increase in computational power boosted by the wide use of distributed computing devices and strategies, we can further advance the parameter scale of foundation models from million-level to billion-level and in some cases even a trillion-level. However, training these models from scratch or performing full fine-tuning at every layer requires a lot of computational and financial resources with some costing tens of millions of dollars to train [3] which is impractical for resource-constrained environments. A solution to this is using parameter-efficient fine-tuning methods such as LoRA (Low-Rank Adaptation) which allows selective adaptation of pre-trained models with minimal computational overhead through trainable low-rank matrices whilst keeping the base model frozen and therefore reducing the overall number of hyperparameters from the base model.

Although LoRA improves the model's efficiency and the problem of overfitting through reducing the number of hyperparameters, its performance is highly sensitive to these changes within the configuration of hyperparameters such that the choice of rank, alpha scaling factor, learning rate and other training hyperparameters can affect the model's performance. If one resorts to using manual hyperparameter tuning algorithms such as Grid Search then not only is this slow but the search space will not be fully explored and hence this challenge of correctly choosing an efficient hyperparameter fine-tuning method is critical for emotion classification tasks in order for the model to capture linguistic patterns across a range of emotions.

This paper presents our research into using the LoRA fine-tuning algorithm to optimize an emotion classification system built using the foundation model DistilBERT where we utilize the emotion dataset, containing tweets across six distinct emotional categories. We aim to finetune DistilBERT by adapting it's classification head for our task of text classification and then using LoRA combined with metaheuristic and evolutionary optimization algorithms to then evaluate the performance of the model.

II. Literature Review

As larger foundation models are trained every few months, the technique of fine-tuning, which updates all parameters of a pre-trained model, becomes a critical deployment challenge, *e.g.*, GPT-3 [4] with 175 billion parameters. The need for efficient approaches becomes a requirement in the field of transformer fine-tuning.

Adapter tuning [5] is an early approach to address this challenge. The approach inserts 'adapter' modules that have a bottleneck architecture between layers in the Pre-Trained Models (PTMs) and only these modules get updated during fine-tuning. On the other hand, an approach like BitFit [6] updates only the biases in foundation models while freezing the rest of the other modules. These methods, while effective, tend to constrain where and how the adaptation occurs, which can end up either potentially introducing inference latency, or reducing the expressive capacity of the model for complex tasks.

Low-Rank Adaptation (LoRA) [7] offers greater flexibility by allowing selective adaptation of any weight matrix in the model through low-rank decomposition. This is an approach that can address the previous limitations by utilising trainable rank decomposition matrices into the model layers, that are later merged with the original weights post training to eliminate inference overhead while maintaining competitive performance.

LoRA freezes the pre-trained weights W_0 and injects trainable low-rank decompositions such that the adapted weight matrix W is defined as:

$$W = W_0 + \Delta W = W_0 + BA \tag{1}$$

where given that $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$ then $W \in \mathbb{R}^{d \times k}$, such that the rank $r \ll \min(d, k)$, and the update ΔW is scaled by α / r.

This architecture introduces several interdependent hyperparameters:

- **rank** r: Controls adaptation capacity and parameter count.
- **scaling factor** α: Affects the update magnitude.
- **dropout rate**: For regularisation.
- **target modules**: Which layers to adapt.
- **learning rate** and **warm-up ratio**.

These collectively form a complex mixed discrete-continuous optimisation problem that we address through meta-heuristics.

Some traditional methods of hyperparameter optimisation include grid search, random search, and Bayesian optimisation. Both grid search and random search suffer from the curse of dimensionality [8] making them impractical for high-dimensional spaces.

Bayesian optimisation [9] offers an improved sample efficiency by building a probabilistic surrogate model (typically a Gaussian Process) of the objective function. However, the approach faces challenges with mixed discrete and continuous spaces and categorical variables, which are inherent characteristics of LoRA hyperparameter optimisation where parameters like rank and target modules are discrete, while learning rate and dropout are continuous.

Bayesian optimization relies on the notion of proximity and smoothness in a continuous space that is naturally captured by a common Gaussian Process, hence the failure to accurately optimise discrete variables. Additionally, the structural dependency in LoRA (where changing target modules alters the model architecture) would violate the smooth function assumption underlying most Bayesian optimization approaches (where points close to each other in the input space will have similar output values).

For these problems, meta-heuristic algorithms provide an appealing alternative strategy. Because they can naturally handle both discrete and continuous variable types, population-based techniques like the Genetic Algorithm (GA) [10], Differential Evolution (DE) [11], and Particle Swarm Optimization (PSO) [12] are especially attractive. Importantly, they can simultaneously explore several regions of the search space and do not require any assumptions regarding the smoothness or differentiability of the objective function. These features make them appealing for costly black-box optimization problems like hyperparameter tuning in LoRA, where every evaluation requires a complete model training run, which is computationally expensive.

While focused on a Convolutional Neural Network's (CNN) hyperparameters, the recent study by [13] provides a valuable insight. Despite the differences in the base model (DistilBERT in our study), the authors found that the algorithms' efficiency was greatly increased by incorporating domain knowledge about valid architectures into the meta-heuristics, mainly through the design of the architecture encoding (e.g., defining search space boundaries and utilizing modular building blocks). When defining the search space for LoRA hyperparameters in Transformer models, it will be crucial to explore how we can incorporate domain knowledge into the meta-heuristics to ensure that DistilBERT can be fine-tuned efficiently.

On the other hand, a study by [14] compared Differential Evolution (DE), GA, and PSO, and found that DE demonstrated the best performance. The authors concluded that DE's effective ability to balance exploration and exploitation of the search space led to superior optimal solutions, contrasting with PSO's higher risk of premature convergence and GA's slower convergence and limitations for complex hyperparameter configurations.

Overall, it is clear that there are many ways in which parameter-efficient fine-tuning algorithms like LoRA can be applied. Therefore through incorporating knowledge from the existing literature and evaluating performance of population-based methods and evolutionary-based methods, we try to establish an efficient LoRA fine-tuning strategy for the downstream task of analysing the sentiment of sentences from the Emotion dataset.

III. Optimisation Algorithms

A. Problem Representation

The hyperparameter optimisation of LoRA involves a mixed search space consisting of continuous variables (learning rate) and discrete, ordered variables (Rank r, Alpha α, Dropout). To address this, we employed a real-valued vector representation.

The chromosome is represented as a vector $G = [g_1, g_2, \ldots, g_6]$. The learning rate ($g_1$) is encoded directly as a continuous float. The remaining five discrete parameters are encoded as integer indices pointing to a pre-defined list of valid options. For example, if the valid Ranks are $\{2, 4, 8, 16, 24\}$, a gene value of 2.0 corresponds to Rank 8. During fitness evaluation, the vector is decoded: continuous genes are used as-is, while index-based genes are rounded to the nearest integer to select the corresponding hyperparameter.

B. Algorithm 1: Real-Coded Genetic Algorithm (RCGA)

We propose a Real-Coded Genetic Algorithm (RCGA) specifically designed to navigate the mixed discrete-continuous landscape of LoRA fine-tuning. Unlike standard Binary GAs, which suffer from Hamming cliffs—where a single bit flip can cause a drastic change in phenotype, destroying local search progress—RCGA preserves the property of locality. This is crucial for parameters like Rank and Alpha, which possess an ordinal relationship ($r = 4 < r = 8 < r = 16$). By operating in a continuous representation and mapping to the nearest discrete index, the algorithm can exploit local gradients and explore the solution space "between" parents.

1) Crossover (BLX-α): To balance exploration and exploitation, we utilized the Blend Crossover (BLX-α) operator. For two parent genes x_1 and x_2 (where $x_1 < x_2$), the offspring y is sampled uniformly from the interval:

$$[x_1 - \alpha(x_2 - x_1), x_2 + \alpha(x_2 - x_1)] \tag{2}$$

We selected $\alpha = 0.5$, which allows the algorithm to search slightly outside the bounds of the parents, preserving population diversity. For the index-based genes, the resulting floating-point value is clipped to the valid index range and rounded.

2) Mutation: We applied a random reset mutation with a probability of 0.1. If a gene is selected for mutation, it is replaced by a new random value sampled uniformly from the initialization range of that specific parameter.

3) Selection and Elitism: A tournament selection mechanism was used to choose parents for the next generation. To ensure convergence stability, we implemented elitism, where the single best individual from the current generation is copied unchanged to the next generation.

Algorithm 1 RCGA with BLX-α Crossover

```
1: Input: Population Size $N$, Generations $G$, $\alpha = 0.5$
2: Output: Best Hyperparameter Vector $x_{best}$
3: Initialize population $P$ with random real-valued vectors
4: Evaluate fitness $f(x)$ for all $x \in P$
5: for $gen = 1$ to $G$ do
6:     $P_{new} \leftarrow \emptyset$
7:     $x_{best} \leftarrow \text{argmax}_{x \in P} f(x)$
8:     Add $x_{best}$ to $P_{new}$                          ▷ Elitism
9:     while $|P_{new}| < N$ do
10:        Select parents $p_1, p_2$ via Tournament Selection
11:        $child \leftarrow$ empty vector
12:        for each gene $i$ do
13:            $d = |p_{1,i} - p_{2,i}|$
14:            $min = \min(p_{1,i}, p_{2,i}) - \alpha \cdot d$
15:            $max = \max(p_{1,i}, p_{2,i}) + \alpha \cdot d$
16:            $child_i \leftarrow \text{Uniform}(min, max)$
17:            $child_i \leftarrow \text{Clip}(child_i, \text{lower\_bound}_i, \text{upper\_bound}_i)$
18:        end for
19:        Apply Random Reset Mutation to $child$ ($prob = 0.1$)
20:        Add $child$ to $P_{new}$
21:    end while
22:    $P \leftarrow P_{new}$
23:    Evaluate fitness for all $x \in P$
24: end for
25: return $x_{best}$
```

IV. Experimental Results

A. Experimental Setup

Experiments were conducted using the Google Colab environment with a TPU v5e-1 accelerator. To balance computational feasibility with statistical significance within a resource-constrained environment, the DistilBERT model was fine-tuned on a stratified subset of 3,000 samples from the Emotion dataset. This prioritized a higher volume of evaluations (100 per algorithm) and multi-seed robustness testing over a single training run on the full dataset. To account for stochasticity in neural network training, the final robust evaluation involved re-training the top candidate solutions on three distinct random seeds $(42, 43, 44)$ to report the mean accuracy and standard deviation.

The search space was defined as follows:

- Learning Rate: $[1 \times 10^{-5}, 2 \times 10^{-4}]$
- Rank r: $\{2, 4, 8, 16, 24\}$
- Alpha α: $\{8, 16, 32, 64, 96\}$
- Warmup Ratio: $\{0.0, 0.06, 0.1\}$
- Dropout: $\{0.0, 0.05, 0.1, 0.2\}$
- Target Modules: {Attention, Attention+FFN}

B. Baseline: Random Search

A random search was performed for 20 trials to establish a baseline. The process took approximately 21 minutes (1,273 seconds). The results showed high variance, highlighting the sensitivity of LoRA to hyperparameter configurations.

TABLE I
Baseline Random Search Statistics

Metric	Value
Best Accuracy	87.80%
Mean Accuracy	69.40%
Std. Deviation	10.02%
Total Time	1,255s

The best model from Random Search achieved 88.65% accuracy using Rank 4 and Alpha 64. However, the low mean accuracy (58.90%) indicates that a large portion of the search space yields sub-optimal models, motivating the use of evolutionary algorithms.

C. RCGA Results

The RCGA was run for 5 generations with a population size of 20 (100 total evaluations). The algorithm demonstrated rapid convergence. As shown in Fig. 1, the best accuracy improved from 89.75% in Generation 1 to 90.55% in Generation 5.

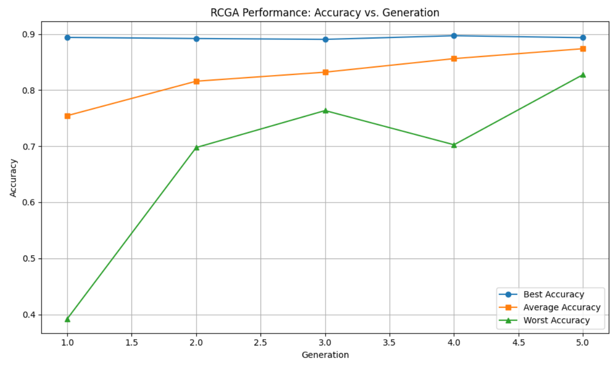

Fig. 1. RCGA Convergence: The algorithm improves accuracy from 89.75% to 90.55% over 5 generations. The narrowing gap between Best and Average indicates population convergence.

The best solution found by RCGA achieved an accuracy of **90.55%**, outperforming the random search baseline by 2.75%. The winning configuration utilised a higher Rank ($r = 16$) and Alpha ($\alpha = 96$) with both Attention and Feed-Forward layers adapted, suggesting that for this specific task, a higher capacity adapter was beneficial.

a) Robustness Verification: To verify that the RCGA result was not a statistical anomaly, the top 5 solutions were re-evaluated across three seeds. The top-ranked solution maintained a high mean accuracy of 89.57% ($\pm 0.33\%$). Interestingly, the 3rd ranked solution from the optimisation run proved to be the most robust, achieving a mean accuracy of 90.37% ($\pm 0.72\%$).

TABLE II
BEST HYPERPARAMETERS FOUND (RCGA VS BASELINE)

Parameter	**Random Search Best**	**RCGA Best**
Learning Rate	1.73×10^{-4}	2.24×10^{-4}
Rank (r)	4	16
Alpha (α)	64	96
Target Modules	Attn + FFN	Attn + FFN
Max Accuracy	**87.80%**	**90.55%**

D. Discussion

The RCGA successfully navigated the search space, outperforming the baseline by 2.75%. A critical insight is the selection of **Target Modules**. Random Search often selected only 'Attention' modules, yielding lower accuracy. The RCGA converged on **Attention + Feed-Forward (FFN)** layers, suggesting that for sentiment analysis, modifying attention alone is insufficient; the model requires capacity updates in the FFN to capture semantic emotional nuances.

Furthermore, the algorithm converged on a high scaling factor. The LoRA update rule is $\Delta W = \frac{\alpha}{r} BA$. Standard heuristics suggest $\alpha \approx r$ (scaling factor ≈ 1). However, our optimal solution ($r = 16, \alpha = 96$) yields a scaling factor of $\frac{96}{16} = 6$. This implies that the pre-trained weights required aggressive adaptation magnitudes to shift from general language modelling to specific sentiment classification. The RCGA's ability to locate this configuration at the edge of the search space validates the use of BLX-α crossover for boundary exploration.

V. CONCLUSION AND FUTURE WORK

In this study, we investigated the efficacy of evolutionary algorithms for hyperparameter optimisation of LoRA-tuned Transformer models. We established a baseline using Random Search, which yielded a best accuracy of 87.80% and a mean accuracy of 69.40%. While Random Search provided a competitive best solution, the variability in results highlights the complexity of the search space and the inefficiency of unguided search.

We implemented a Real-Coded Genetic Algorithm (RCGA) with BLX-α crossover to address the mixed discrete-continuous nature of the problem. The RCGA demonstrated rapid convergence, improving the best solution to **90.55%** within just 5 generations. Robustness testing across multiple random seeds confirmed that the solutions found by RCGA were stable, with the top-performing models consistently achieving $\approx$90% accuracy. The results highlight that adapting both Attention and Feed-Forward modules, combined with a higher Rank ($r = 16$) and Alpha ($\alpha = 96$), was key to maximizing performance on the Emotion dataset.

Future work could extend this research by applying these techniques to larger foundation models (e.g., Llama-3 or Mistral) to verify if these hyperparameter patterns hold at scale. Additionally, further analysis into the correlation between specific LoRA target modules and linguistic tasks could provide heuristics to narrow the search space before optimisation begins, potentially reducing the computational cost of the evolutionary search.

APPENDIX

Contribution Statement

- **Zayaan Khan**:
- **Rita San**:
- **Steven Thomas**:
- **Joel D'Souza**:

REFERENCES

[1] ScienceDirect, "Article PII S2666651021000231," [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651021000231. [Accessed: 2024].

[2] IBM, "Tuning with LoRA," IBM Watsonx Documentation. [Online]. Available: https://www.ibm.com/docs/en/watsonx/w-and-w/2.1.0?topic =tuning-lora-fine.

[3] M. H. Baig *et al.*, "LoRA-XS: Low-Rank Adaptation with Extremely Small Number of Parameters," *arXiv preprint arXiv:2405.21015*, 2024. [Online]. Available: https://arxiv.org/abs/2405.21015.

[4] T. Brown *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901. Available: https://arxiv.org/abs/2005.14165.

[5] N. Houlsby *et al.*, "Parameter-efficient transfer learning for NLP," in *International Conference on Machine Learning (ICML)*, 2019, pp. 2790–2799. Available: https://arxiv.org/abs/1902.00751.

[6] E. B. Zaken, S. Ravfogel, and Y. Goldberg, "BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022. Available: https://arxiv.org/abs/2106.10199.

[7] E. J. Hu *et al.*, "LoRA: Low-Rank Adaptation of Large Language Models," in *International Conference on Learning Representations (ICLR)*, 2022. Available: https://arxiv.org/abs/2106.09685.

[8] E. Keogh and A. Mueen, "Curse of Dimensionality," in *Encyclopedia of Machine Learning and Data Mining*, Springer, 2017. Available: https://link.springer.com/rwe/10.1007/978-1-4899-7687-1_192.

[9] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, vol. 25, 2012. Available: https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

[10] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

[11] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[12] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," *Soft Computing*, vol. 22, pp. 387–408, 2018.

[13] N. Babaeva *et al.*, "Hyperparameters optimization of Convolutional Neural Networks using Parameter-Setting-Free Harmony Search," in *IEEE Congress on Evolutionary Computation*, 2021.

[14] S. Sen *et al.*, "A comparative study of DE, GA and PSO applied to complex hyperparameter optimization," *Applied Soft Computing*, 2022.