Parameter-Efficient Fine-Tuning via Metaheuristic Algorithms for Sentiment Analysis in Foundation Models

Zayaan K. 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) have been growing in usage over the years for various purposes such as text generation, translation and understanding the semantic meaning behind a corpus. Transformers which serve as the underlying backbone of many of these large language models are becoming increasingly harder to finetune for optimal performance through standard techniques such as data augmentation, regularization and discriminative fine tuning (applying different learning rates to each layer of the model), ultimately leading to the model becoming overconfident and having hallucinations due to the vast number of parameters being required to control. We propose LoRA (Low-Rank Adaptation), an algorithm which for a given task [1], adapts a foundation model by freezing the weights and injecting trainable rank decomposition matrices into each layer. We will be using LoRA on the DistilBERT foundation model for the downstream task of text classification on a dataset of tweets representing various emotions within their semantic meanings as well as discussing the various approaches a foundation model can use to find optimal hyperparameters through both evolutionary & metaheuristic algorithms such that overall, we aim to improve the overall confidence score of the foundation model when performing sentiment analysis on a dataset of tweets expressing six different emotions.**

1 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. 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 [2] 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.

2 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 [1] with 175 billion parameters. The need for efficient approaches becomes a requirement in the field of transformer fine-tuning.

Adapter tuning [2] 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 [3] 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) [4] 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 [5] making them impractical for high-dimensional spaces.

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) [6], Differential Evolution (DE) [7], and Particle Swarm Optimization (PSO) [**?**] 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 [] 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.

[CHECK]On the other hand, a study by [] 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.

3 Optimisation Algorithms

3.1 Problem Representation

A chromosome (candidate solution) 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 predefined 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 with a repair mechanism; continuous genes are used as-is, while index-based genes are rounded to the nearest integer to select the corresponding hyperparameter.

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: $\{\text{Attention}, \text{Attention+FFN}\}$

We use index-based encoding to ensure uniform exploration across discrete options. For example, a mutation step size of 5 with value-based encoding would have different effects at

$rank = 4$ (significant change), as opposed to $rank = 24$ (minimal change). Index-based encoding ensures consistency between irregularly spaced out hyperparameter options.

3.2 Algorithm 1: RCGA BLX-α

We propose a Real-Coded Genetic Algorithm (RCGA) for optimising LoRA hyperparameters within a mixed discrete-continuous search space. Unlike standard Binary GAs, which suffer from Hamming cliffs, where small mutations to the bit string can cause drastic changes in parameter values, disrupting the search, RCGA preserves the property of locality. This is crucial for parameters like Rank and Alpha, which possess an ordinal relationship (i.e the categories have a meaningful order but the distance between them is not equal; $r = 4 < r = 8 < r = 16$). 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. We can now describe the genetic operators implementing this approach:

3.2.1 Crossover (BLX-α): To balance exploration and exploitation, we utilised 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.

3.2.2 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 initialisation range of that specific parameter.

3.2.3 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.

3.3 Algorithm 2: SHADE

The comparative study by Sen et al. [8] mentioned in this introduction suggests that Evolutionary Algorithms, particularly Differential Evolution (DE), are an ideal choice for navigating gradient-free search spaces. A notable drawback in standard DE is its sensitivity to the manual selection of its control parameters (Crossover Rate CR and Scaling Factor F), such that initialising sub-optimal values for these parameters can lead to undesired convergence rates. We instead adopted an enhanced variant of DE: Success-History based Adaptive Differential Evolution (SHADE). SHADE improves on standard DE by automatically adapting its control parameters, driven by historical success information (a form of elitism). This is critical for hyperparameter optimisation where the optimal exploration-exploitation balance may shift as the search progresses through the fitness landscape.

Algorithm 1 RCGA with BLX-α Crossover

Input: Population Size N, Generations G, $\alpha = 0.5$
Output: Best Hyperparameter Vector x_{best}
Initialize population P with random real-valued vectors
Evaluate fitness $f(x)$ for all $x \in P$
for $gen = 1$ to G **do**
    $P_{new} \leftarrow \emptyset$
    $x_{best} \leftarrow \text{argmax}_{x \in P} f(x)$
    Add x_{best} to P_{new} `// Elitism`
    while $|P_{new}| < N$ **do**
        Select parents p_1, p_2 via Tournament Selection
        $child \leftarrow$ **empty vector**
        for each gene i **do**
            $d = |p_{1,i} - p_{2,i}|$
            $min = \min(p_{1,i}, p_{2,i}) - \alpha \cdot d$
            $max = \max(p_{1,i}, p_{2,i}) + \alpha \cdot d$
            $child_i \leftarrow \text{Uniform}(min, max)$
            $child_i \leftarrow \text{Clip}(child_i, \text{lower\_bound}_i, \text{upper\_bound}_i)$
        end for
        Apply Random Reset Mutation to $child$ ($prob = 0.1$)
        Add $child$ to P_{new}
    end while
    $P \leftarrow P_{new}$
    Evaluate fitness for all $x \in P$
end for
return x_{best}

Thoughtful considerations regarding the computational expense of training transformer models (approximately 3-5 minutes per evaluation) were carried out, leaving us with a severely constrained optimisation budget (100 evaluations). However, SHADE's adaptive mechanism allows it to learn from successful parameter combinations, and focus computational resources more effectively compared to random or grid search approaches.

The LoRA hyperparameter optimisation problem presents itself as a mixed continuous-discrete search space. We employ the repair mechanism to map the outputs of SHADE's continuous mutation operators to discrete values. The continuous candidate vectors produced by the mutation operator can be defined as:

$$v_i = x_i + F_i \cdot (x_{pbest} - x_i) + F_i \cdot (x_{r1} - x_{r2}) \tag{3}$$

This approach allows for smooth exploration while respecting discrete constraints.

Furthermore, SHADE maintains an external archive of recently replaced solutions, providing additional diversity for the mutation operator. This is especially valuable in hyperparameter optimisation where similar configurations may perform differently due to local interactions between parameters.

SHADE has demonstrated superior performance on the CEC2014 benchmark suite and various real-world optimisation problems, particularly excelling in problems with limited evaluation budgets; precisely our scenario [9]. Although JADE

Algorithm 2 Success History Adaptive Differential Evolution

Input: $N = 20$ (pop size), $H = 20$ (memory size), $G_{max} = 5$, $arc\_rate = 1.0$, $p = 0.4$
Initialize:
$M_{CR}, M_F = [0.5] \times H$ // historical parameter memories
$Archive = \emptyset, k = 0$
$P = \{\text{random individuals}\}$, evaluate all, track x_{best}
// Main Loop:
for $generation = 1$ **to** G_{max} **do**
    sort population by fitness (descending)
    $p_{num} = \max(2, \lfloor p \times N \rfloor)$
    // generate offspring
    for each individual i **do**
        // sample parameters from memory
        $r = \text{rand\_int}(0, H - 1)$
        $CR_i = 0$ **if** $M_{CR}[r] == -1$ **else** $\text{clip}(\mathcal{N}(M_{CR}[r], 0.1), 0, 1)$
        $F_i = \min(\text{Cauchy}(M_F[r], 0.1), 1.0)$ where $F_i > 0$
        // mutation: current-to-pbest/1
        Select: p_{best} (from top p_{num}), $r1 \neq i$ (from pop), $r2$ (from pop $\cup$ Archive)
        $v_i = x_i + F_i \times (x_{pbest} - x_i) + F_i \times (x_{r1} - x_{r2})$
        // binomial crossover
        $u_i = \text{crossover}(x_i, v_i, CR_i)$
        $u_i = \text{repair}(u_i)$
        $f_{ui} = \text{evaluate}(u_i)$
    end for
    // selection and parameter recording
    $S_{CR}, S_F, \Delta f = []$
    for each individual i **do**
        if $f_{ui} \geq f_i$ **then**
            if $f_{ui} > f_i$ **then**
                $Archive \leftarrow x_i$ (randomly replace if full)
                Record: $S_{CR} \leftarrow CR_i, S_F \leftarrow F_i, \Delta f \leftarrow |f_{ui} - f_i|$
            end if
            $x_i \leftarrow u_i, f_i \leftarrow f_{ui}$
            Update x_{best} if improved
        end if
    end for
    // update memory (weighted Lehmer mean)
    if $|S_{CR}| > 0$ **then**
        $w = \Delta f / \text{sum}(\Delta f)$ // weights from improvement
        $M_F[k] = \Sigma(w \times S_F^2) / \Sigma(w \times S_F)$
        $M_{CR}[k] = -1$ **if all** $S_{CR} = 0$ **else** $\Sigma(w \times S_{CR}^2) / \Sigma(w \times S_{CR})$
        $k = (k + 1) \mod H$
    end if
end for
Output: x_{best}, f_{best}

(Joint Adaptive Differential Evolution) similarly addresses the limitations of manual parameter tuning found in standard DE, it does not guarantee diversity within the population due to its reliance on a single average. SHADE improves upon this foundation by saving successful parameters to a history list and randomly selects parameter pairs from the list, preventing premature convergence. [10]

4 Experimental Results

4.1 Experimental Setup & Parameter Ranges

We first initialise a population of candidates with random configurations of indices mapped to a specific set of LoRA hyperparameters we decide on. The fitnesses of these individuals are then calculated by obtaining the validation accuracy after training DistilBERT for 3 epochs. The following process outlines the method of evaluation:

1) Decode the index vector x into LoRA hyperparameters via a repair function
2) Load a fresh DistilBERT-base-uncased model
3) Apply LoRA adaptation with decoded hyperparameters and freeze the original weights of DistilBERT
4) Train the LoRA matrices and the final, fully connected layer for 3 epochs on 3,000 samples from the Emotion datasets train set using the AdamW optimiser
5) Evaluate on the full validation set (2,000 samples)
6) Return validation accuracy as fitness (maximisation objective)

All algorithms are initialised with a population of 20 with individuals being evaluated at the start. Offsprings and updates are generated from the starting population and undergo four generational updates, totalling for 100 evaluations. We have chosen these small values (relative to literature) for population and generations as the search space itself has been constrained due to the computational intensiveness of the task at hand. Three epochs are used during training for all algorithms to reinforce fairness, and limit evaluation time. P100 GPUs are used in training for their speed and availability. Finally, the top five configurations are taken from each algorithm and are trained on three unique seeds. The results shown on Table 2 and Table I are from the best performing configuration of each algorithm.

Algorithm Parameter Ranges:

TABLE I
Algorithm Parameters

Algorithm	Parameters
PSO	$w = 0.7$, $c_1 = c_2 = 1.5$
RCGA BLX-α	$\alpha = 0.5$, mut. prob. $< 10\%$
SHADE	$H = 20$, A $= 1.0$, $p_{\text{best}} = 0.4$

For PSO, w controls velocity momentum while c_1 and c_2 are used for cognitive and social learning respectively. In RCGA BLX-α, α defines the exploration range beyond parent bounds, with mutation applied to individual hyperparameters. SHADE adapts its parameters using a history of size H, maintains an archive of inferior solutions scaled by the archive rate, and an

individual from the top p_{best} fraction of the population is used to help create a mutant vector.

5 Results and Key Findings

The experimental set up comprises of 4 metaheuristic algorithms; The baseline 20-trial random search, the population based PSO, and two chosen algorithms, RCGA BLX-alpha and SHADE.

Top Configuration Three Seed Results					
Algorithm	Best Acc.	Mean Acc.	±Std.	Evaluations	Time (min)
Random Search	0.878	0.877	0.003	20	21.04
PSO	0.905	0.893	0.002	100	108.44
RCGA BLX-Alpha	0.906	0.896	0.004	100	108.55
SHADE	0.900	0.897	0.003	100	72.90

Fig. 1. Three seed performance of top LoRA configurations

Best Performing LoRA Configurations						
Algorithm	Learning Rate	Warm Up	Rank	Alpha	Dropout	Target Modules
Random Search	0.0002	0.06	4	64	0.1	Attn+FFN
PSO	0.0002	0.1	4	96	0.0	Attn+FFN
RCGA BLX-Alpha	0.0002	0.0	16	96	0.0	Attn+FFN
SHADE	0.0002	0.0	24	96	0.0	Attn+FFN

Fig. 2. Best LoRA hyperparameter configurations chosen by algorithms

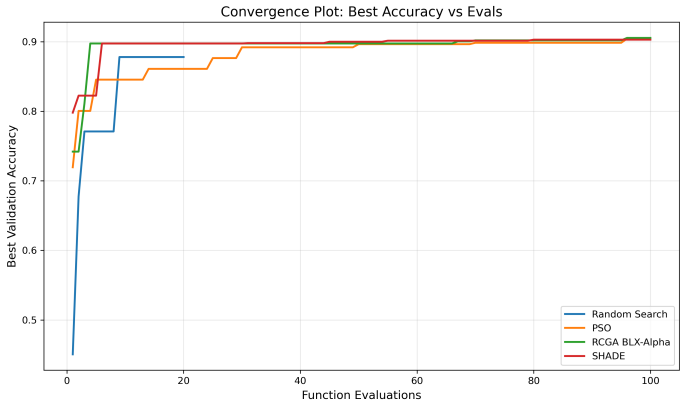

Fig. 3. Best Accuracy of Algorithm vs. Objective Evaluations

Convergence Analysis: All four algorithms converged to values for learning rate (0.0002) and target modules (attention + feedforward), suggesting that these represent ideal optima within the search space. Similarly, the meta-heuristic methods identified alpha=96 and dropout=0.0 as optimal, while Random Search resolved at suboptimal values (alpha=64, and dropout=0.1). This is likely due to its limited smaller evaluation budget of 20 evaluations, compared to 100 for the meta-heuristic algorithms. The most notable difference occurred in the rank parameter. SHADE and RCGA BLX-Alpha discovered higher rank values of 24 and 16 respectively, which could correlate with their marginally better performance. PSO converged to rank=4, suggesting that its velocity based updates were less effective at escaping the local region compared to the mutation based exploration of SHADE and RCGA. All metaheuristic methods rejected dropout regularisation. This could potentially be because the 3 epoch training loop for evaluations is too short for overfitting to occur; leading to the benefit of dropout regularisation not being able to manifest.

The clearest connection from these results was that the more trainable parameters that are available, the better the model performs. This is shown in the two best metaheuristic algorithms converging to a higher rank, and all top configurations of the four algorithms selecting both attention heads and feed forward layers as the target modules for LoRA's application, greatly increasing the number of trainable parameters. Another interesting find is the scaling factor alpha being the highest possible value. This shows that the LoRA adapted weights having a very high influence compared to DistilBERTs pre-trained knowledge (weights), is crucial for greater performance.

Convergence Speed and Search Space Analysis: Figure 3 shows all the population achieving accuracy $>$0.85 within the 20 initial evaluations. This rapid convergence can be attributed to 2 factors:

1) The constrained discrete search space increasing the probability of sampling good configurations randomly.
2) AdamW's robustness as the training optimiser pushing the accuracy ceiling higher, allowing many configurations to perform relatively better than if a worse optimiser like SGD was used for training.

The level of convergence suggests that for similarly constrained LoRA fine-tuning problems, a smaller evaluation budget, with simply more random restarts may be a more cost effective approach than an extended metaheuristic search with several generational updates.

Computational Efficiency: SHADE achieved comparable RCGA BLX-Alpha; 0.897 vs. 0.896 in 33% less time less time; 72.90 vs. 108.55 minutes. This makes our implementation of SHADE the most efficient metaheuristic method in this experiment. However, the 3-5$\times$ computational cost of any metaheuristic algorithm over Random Search yielded only a $\sim$2% accuracy improvement. This raises questions about the cost-effectiveness of these algorithms. To scrutinise further, if we go back to Figure 3, for both SHADE and RCGA BLX-Alpha, the optimal parameters are found by sheer luck in the initial random population generation phase, further reinforcing the point that if for metaheuristic methods, random restarts with checkpoints of best solutions or simple Random Search with more trials may just be the better approach for problems with constrained search spaces like ours.

6 Multi-objective Extension

In this section, we explore the use of multi-objective evolutionary algorithms to determine the optimal trade-off boundary between validation accuracy and model complexity (measured by trainable parameters) for the LoRA adaptation of DistilBERT. We opted for the Non-dominated Sorted Genetic Algorithm II (NSGA-II), proposed by Deb et al. [11], as it provides computational efficiency in bi-objective problems through its fast non-dominated sorting mechanism ($O(MN^2)$). Furthermore, its elitist strategy means that optimal hyperparameter configurations are not lost through stochastic selection

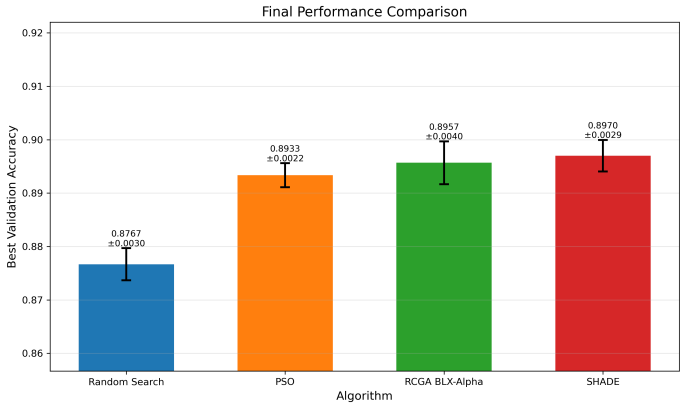

Fig. 4. Comparison of Final Performances on Three Seeds

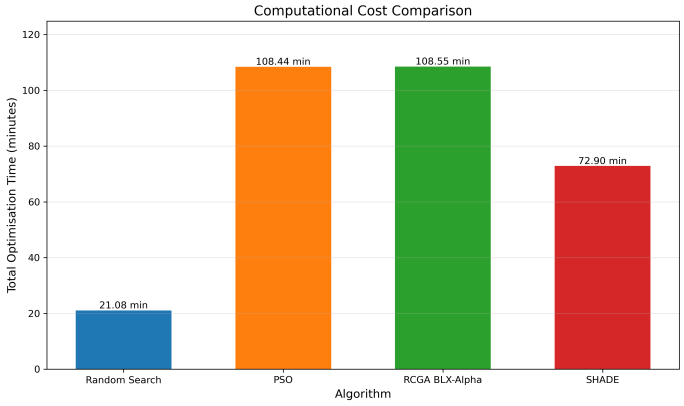

Fig. 5. Training Time of All Algorithms

processes, but are rather passed onto the next generation. This ensures monotonic improvement in the Pareto front.

Another Pareto front-based multi-objective algorithm worth discussing is the Strength Pareto Evolutionary Algorithm 2 (SPEA2), which utilises a k-th nearest neighbour approach to guide the search process. Similarly, SPEA2 demonstrates elitism much like NSGA-II, but by employing a fixed-size external archive system. The archive is updated per generation by ensuring non-dominant solutions are kept to be used to create new offspring. While SPEA2 is a strong candidate for approaching the LoRA hyperparameter optimisation problem, it is more strongly suited for problems with 3 or more objectives–evident from Zitzler et al's paper when comparing SPEA2 and NSGA-II. As the result of the additional nearest neighbour calculations, that is, the calculations of the euclidean distances between each individual before sorting them, we face an increase computational complexity of $O(M^3)$ (or in some cases, $O(M^2 logM)$) where M is the population size. [12]

We use the same LoRA hyperparameter configurations as those in the other algorithms discussed in sections 3, with the evolutionary parameters also unchanged (population size $P = 20$ and generations $G = 5$), matching the exact search space and evaluations to ensure fairness. While the chosen single-objective algorithms focused on maximising validation accuracy, the complexity of the resultant neural architectures did not contribute to selection pressure. In other words, these algorithms favour models with higher accuracy, regardless of over-parametrisation. For example, one LoRA configuration could produce an accuracy of 85.0% with $rank = 4$ and $\#trainable\_params = 600,000$, resulting in low training cost. Another configuration could produce an accuracy of 85.1% with $rank = 24$ and $\#trainable\_params = 2,000,000$, in which the training cost has significantly increased, but this cost is insignificant to single-objective algorithms and it will only select the model configuration with higher accuracy ($85.1\% > 85.0\%$). Therefore, we can justify the need to introduce parameter efficiency as a primary optimisation goal alongside accuracy.

Figure 6 presents the Pareto front procured from the NSGA-II optimisation process, plotting validation accuracy against the number of trainable parameters.

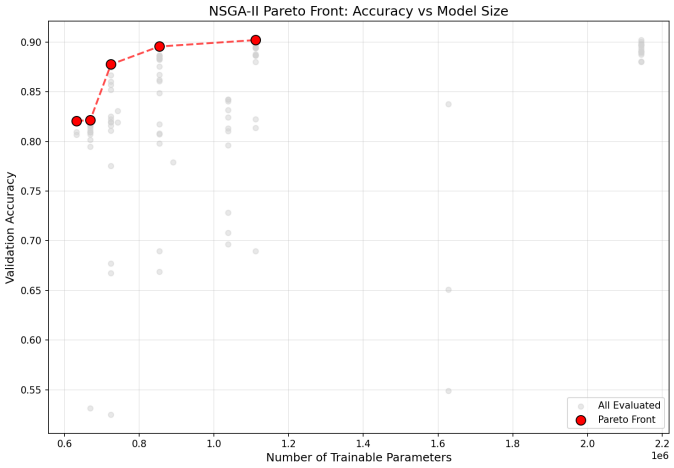

Fig. 6. NSGA-II: Pareto Front

NSGA-II Pareto Front Solutions				
Solution	Rank	Trainable Params	Accuracy	Target Modules
1	2	0.63M	82.05%	Attn
2	4	0.67M	82.15%	Attn
3	2	0.72M	87.75%	Attn+FFN
4	4	0.85M	89.55%	Attn+FFN
5	8	1.11M	90.20%	Attn+FFN

Fig. 7. Pareto optimal LoRA configurations from NSGA-II

The red dots represent Pareto optimal solutions also known as non-dominated solutions, which are characterised by having no further possibility of improving an objective without harming the other objectives [13] which, in this case, is the optimal trade-off between validation accuracy and number of trainable parameters. The grey dots on the other hand, represent the Pareto dominated solutions which were discovered during the optimisation process. These solutions are considered sub-optimal as they are overcome by non-dominated (Pareto front)

solutions which achieve equal or higher accuracies with the same or fewer amount of trainable parameters.

The front path displays a steep vertical ascent as the number of trainable parameters increase, until the path halts at a validation accuracy 90.2% with 1.1M trainable parameters. The steepest ascent occurs when the parameters are < 0.72M, suggesting that the model is constrained by capacity. Evidently from the results, an increase of model size from 0.67M to 0.72M (8% increase) training parameters provided a gain in accuracy of 82.15% to 87.75% (5.6% increase). This "High-Gain, Low-Cost" behaviour demonstrates that LoRA configurations targeting only attention layers (shown in Table 7 [solution 1 & 2]) underfit the emotion dataset regardless of rank. The jump from 82.15% to 87.75% when adding FFN modules, $solution_2 \rightarrow solution_3$, suggests that adapting the feedforward layers is critical for this task.

The knee point becomes visible at approximately 0.72M parameters, representing the most efficient trade-off in the search space as it approaches the peak accuracy. At this point the model size is roughly a third smaller than the largest Pareto solution.

As the front path exceeds the knee point, the curve plateaus, converging to the maximum accuracy. During this stage, it can be observed that achieving a final 2% gain in accuracy requires a costly increase of nearly 53% in trainable parameters.

This analysis suggests that the LoRA hyperparameter optimisation problem benefits significantly from multi-objective optimisation algorithms. In contrast to single-objective optimisation algorithms, which have no mechanism to penalise model complexity and tend to select high-capacity configurations for marginal accuracy gains, multi-objective optimisation algorithms can penalise high model-complexity. This strategy ensures smooth convergence towards the 'knee point' of the Pareto front, eventually achieving optimal efficiency by avoiding the selection of over-parametrised LoRA configurations. Ultimately, treating the number of training parameters and validation accuracy as conflicting objectives, introduces a dimension of efficiency to the model selection process that would otherwise be ignored by single-objective algorithms.

REFERENCES

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[2] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," 2019. [Online]. Available: https://arxiv.org/abs/1902.00751

[3] E. B. Zaken, S. Ravfogel, and Y. Goldberg, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," 2022. [Online]. Available: https://arxiv.org/abs/2106.10199

[4] 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," 2021. [Online]. Available: https://arxiv.org/abs/2106.09685

[5] E. Keogh and A. Mueen, *Curse of Dimensionality*. Boston, MA: Springer US, 2017, pp. 314–315. [Online]. Available: https://doi.org/10.1007/978-1-4899-7687-1_192

[6] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Cambridge, MA, USA: The MIT Press, 1992.

[7] 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.

[8] S. Dhar, A. Sen, A. Bandyopadhyay, N. Jana, A. Ghosh, and Z. Sarayloo, "Differential evolution algorithm based hyper-parameters selection of convolutional neural network for speech command recognition," in *Proceedings of the 15th International Joint Conference on Computational Intelligence*. SCITEPRESS - Science and Technology Publications, 2023, p. 315–322. [Online]. Available: http://dx.doi.org/10.5220/0012251500003595

[9] R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 1658–1665.

[10] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *2013 IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 71–78.

[11] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[12] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," *TIK-Report*, vol. 103, 07 2001.

[13] Cenaero, "Page title," Cenaero.be (via Internet Archive), Feb. 2020, [Online]. Available: https://web.archive.org/web/20200226003108/https://www.cenaero.be/Page.asp?docid=27103.

Due to the nature of the components of this coursework being inseparable, all the group members have agreed to contribute equally in all components. Therefore, each group member has had a significant impact on all graded sections of this work.