

# Adaptive Curriculum Learning for Efficient Domain-Specific Fine-Tuning with Parameter-Efficient Strategies in LLMs

G. Srinivasa Raju<sup>1\*</sup>, Dr.A. Sri Krishna<sup>2</sup>, and Dr. Malijeddi Murali<sup>3</sup>

<sup>1</sup>\*Research Scholar, Department of Computer Science and Engineering, Centurion University of Technology and Management [CUTM], Vizianagaram, Andhra Pradesh, India.  
vasuraju1996@gmail.com, <https://orcid.org/0009-0005-9287-1739>

<sup>2</sup>Department of Artificial Intelligence, Shri Vishnu Engineering College for Women, Bhimavaram, Andhra Pradesh, India. [srikrishna.au@gmail.com](mailto:srikrishna.au@gmail.com),  
<https://orcid.org/0000-0002-9134-011X>

<sup>3</sup>Professor, Department of Electronics and Communication Engineering, ACE Engineering College, Hyderabad, India. [muralitejas@gmail.com](mailto:muralitejas@gmail.com),  
<https://orcid.org/0000-0002-8559-0091>

Received: October 29, 2025; Revised: December 27, 2025; Accepted: February 09, 2026; Published: March 31, 2026

## Abstract

The proposed ACL-PEFT-LLM model combines Adaptive Curriculum Learning (ACL) and Parameters-Efficient Fine-tuning (PEFT) to overcome the shortcomings of current models that employ full fine-tuning, which is computationally expensive and impractical in many domain-specific applications. Although PEFT methods such as LoRA, Prefix Tuning, and Prompt Tuning have been designed to optimize only a few model parameters, thereby reducing memory usage and training time by a significant factor, typically lack an adaptive mechanism to adjust task difficulty during training. By comparison, ACL-PEFT-LLM dynamically adjusts the difficulty of training examples based on the model's current performance, enabling it to start with easier tasks and increase in difficulty. This will guarantee that the model learns effectively without being flooded with challenging examples during the initial stages of learning. The ACL-PEFT-LLM model is superior to the other models in both accuracy and computational efficiency. It has an F1 score of 96.2 and a high accuracy of 96.8, indicating strong task-specific performance across a wide range of datasets, including SST-2, SQuAD, and AIME. Besides, it has a high Accuracy-Efficiency Ratio (AER) of 5.12, indicating a positive trade-off between performance and resource consumption. Compared with other methods, LoRA is most efficient in terms of training time and GPU memory usage, with an accuracy of 94.0 and an F1 score of 93.7, but is marginally lower in performance. Other techniques, such as Full Fine-Tuning, Prefix Tuning, and Adapters, are either less accurate or more resource-intensive, whereas ACL-PEFT-LLM is the most efficient mechanism for domain-specific fine-tuning. ACL-PEFT-LLM offers a potent platform that facilitates efficient domain adaptation at minimal resource usage, which makes it suitable for tasks in domains such as medical, legal, and financial, where computing resources are usually scarce. This model has found a very good compromise between maximum performance and computational efficiency such that tasks that belong to a domain can be efficiently performed without overuse of resources.

---

*Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, volume: 17, number: 1 (March - 2026), pp. 870-894. DOI: [10.58346/JOWUA.2026.11.048](https://doi.org/10.58346/JOWUA.2026.11.048)

\*Corresponding author: Research Scholar, Department of Computer Science and Engineering, Centurion University of Technology and Management [CUTM], Vizianagaram, Andhra Pradesh, India.

**Keywords:** Adaptive Curriculum Learning, Parameter-Efficient Fine-Tuning, Large Language Models (LLMs), Domain-Specific Fine-Tuning, Computational Efficiency, Training Efficiency, Model Adaptation.

## 1 Introduction

The fast-evolving developments in Large Language Model (LLM), numerous applications in natural language processing (NLP) have undergone change, yet the size of such models presents considerable challenges in the way of computational resources, such as memory and computation capacity (Wang et al., 2025; Kawamae, 2025). It can be very computationally-intensive and time-consuming to fine-tune these large models on domain-specific problems, like medical diagnoses, legal document analysis, and financial forecasting (Banitalebi-Dehkordi et al., 2023; El-Demerdash & Abdel-Hamid, 2025). To overcome this, parameter-efficient fine-tuning algorithms have been developed to reduce resource usage while preserving or enhancing performance. Adaptive Curriculum Learning (ACL) is one such approach, which is based on the learning of humans, in which easier tasks are acquired first, before more complicated tasks are studied (Wang et al., 2023; Dun et al., 2024). In contrast to conventional models, which are trained on the entire dataset in a fixed sequence, ACL automatically scales the difficulty of training examples, initially easier, before becoming more difficult, according to the model's performance (Lee et al., 2025; Balaji et al., 2022). This ensures that the model does not become overwhelmed by overly elaborate tasks in the initial stages, allowing it to learn better. Other approaches, such as parameter-efficient methods like Low-Rank Adaptation (LoRA), Prefix Tuning, and Prompt Tuning, can fine-tune only a subset of the model parameters, lowering memory requirements and training time without necessarily retraining the full model (Kong et al., 2024; Noviantari & Agustina, 2023). The ACL, together with parameter-efficient fine-tuning strategies, offers an effective solution to fine-tuning the LLMs on domain-specific tasks. This mixed method enables models to progressively evolve to more complex tasks at a lower computational cost, making it applicable to tasks with limited resources (Verma & Kulkarnin, 2025; Fitria, 2024). The dynamic adjustment of the training challenge and the prioritization of the optimal parameter values in this methodology maximize the effectiveness and adaptability of the LLMs in most domains of specialization and guarantee the high efficiency with low computational cost.

### Key Contribution

- Design and implement an adaptive curriculum learning framework that prioritizes domain-specific tasks based on their complexity, beginning with easier tasks and gradually introducing more complex ones to reduce resource consumption.
- Integrate parameter-efficient fine-tuning (PEFT) strategies by implementing adapter layers and low-rank approximation techniques to minimize the number of parameters that are fine-tuned, optimizing memory and computational resources.
- Evaluate the efficiency of the fine-tuning process by measuring training time, FLOPs, and GPU memory usage in comparison to traditional fine-tuning approaches on three domain-specific datasets: medical, legal, and financial data.
- Compare the performance of the domain-adapted models using standard accuracy metrics for each domain, assessing whether the adaptive curriculum approach maintains or improves domain-specific accuracy while significantly reducing computational cost.

This research is followed by the various sections. Section I introduces the topic, Section II explains the literature review, and Section III presents the research gap. Section III explained the proposed

methodology, followed by the overall architecture diagram, the working procedure for parameter-efficient fine-tuning of LLM, Adaptive curriculum learning through parameter-efficient fine-tuning, the data flow diagram, and the proposed algorithm. Section IV explained the results and discussion, followed by the dataset description, performance comparison with various datasets, and metric analysis. Section V explained the main key summary of this research.

## 2 Literature Review

Natural language processing tasks have also seen significant improvements with recent developments in large language models (LLMs) (Hung & Pan, 2025). Historically, models trained on large-scale corpora are further trained on downstream tasks by retraining all their parameters, which is very expensive and usually infeasible when large, domain-specific downstream tasks are involved (Aazami & Fallah, 2016). In response to this, studies have shifted to methods that minimize computational load without degrading task performance (Battou et al., 2011). Among the notable directions are parameter efficient fine tuning (PEFT), only a handful of parameters is manipulated, which reduces the memory usage and training time by a significant factor. Low-rank adaptation, prefix tuning, prompt tuning, and bit-efficient updates have been shown to be effective methods for reducing the number of parameters modified in only a fraction of the model, achieving strong performance (Zhang, 2024). The approaches enable domain adaptation with minimal resources and are thus appealing across a range of applications, such as medical information extraction and legal document analysis. Curriculum learning is another concept emerging from the processes of human learning, in which training starts with simpler tasks and progresses to more complex ones (Zhang & Galaup, 2023).

The adaptive curriculum learning innovation integrates the benefits of curriculum sequencing with dynamically adjusting learning mechanisms to adjust difficulty targets based on model performance (Sappa, 2025). Rather than operating on difficulty-specific training levels, adaptive systems continually vary the examples to be trained on, based on the model's current performance, to choose samples that are nearest to a dynamically changing target difficulty (Yang et al., 2024). This helps the model to avoid getting stuck on extreme samples (that are too easy or too difficult) and provides a more gradual change in difficulty between samples (Shreshthi et al., 2025; Challa & Bright, 2025). Adaptive curriculum learning, coupled with parameter-efficient fine-tuning, has been discussed in recent works to optimize fine-tuning for domain-specific tasks. Within these frameworks, model training takes a series of steps: general domain knowledge, easier domain tasks, and more advanced specialized tasks and enables the model to train on simpler data as it demands more complex examples that are applicable to the target domain (Prottasha et al., 2024). The dynamic variation of the learning rate and difficulty targets can help the model focus on its capacity and avoid unnecessary computation on overly difficult examples in the initial stages.

Adaptive curriculum has been found to improve both accuracy and generalization, particularly when training data span a broad spectrum of difficulty levels (Mariani et al., 2020; Poncette et al., 2020). Comparative analyses with baseline models trained without curriculum structures have demonstrated this. Research comparing various PEFT methods within an adaptive curriculum has also found that not every parameter-efficient strategy performs well across all areas (Xie et al., 2020; Kavitha & Kirubanand, 2025). As an example, low-rank adaptation can be highly successful in terms of performance and moderate resource usage, whereas prefix and prompt tuning can be more curriculum-sensitive when the domain is complex (Goez et al., 2020; Dong, 2021). Although the literature has addressed adaptive curriculum and parameter efficiency independently, more and more is currently being done on the synergy between adaptive curriculum and parameter efficiency, with existing

models that are more efficient (in terms of training time, GPU memory and number of trainable parameters) and more effective (in terms of domain task accuracy and generalization (Horn & Van Niekerk, 2020)). This trend will indicate that further domain adaptation research will keep improving the interaction between difficulty estimation, dynamic task selection and lightweight parameter updates to create robust and scalable fine-tuning frameworks applicable to large pretrained models to specific domains.

### Research Gap

The lack of knowledge in applying Adaptive Curriculum Learning (ACL) to Parameter-Efficient Fine-Tuning (PEFT) to Large Language Models (LLMs) is due to the lack of research on the potential of their integration in domain-specific tasks. Although PEFT methods can lower the computational cost by modifying a few model parameters, and adaptive curriculum learning can enhance model efficiency by increasing difficulty over time, the synergistic combination of both methods is not well studied. The present of literature largely focuses on these methods as individual approaches and does not discuss their dynamic interaction to achieve optimal fine-tuning. Moreover, the responses of various PEFT methods to curriculum development are insufficiently understood, and how real-time difficulty adjustment can further improve the LLM's performance across all areas of specialization remains unclear. This underscores how research in the future would need to formulate frameworks that can effectively integrate these strategies to produce scalable, efficient, and high-performing domain adaptation.

## 3 Proposed Methodology

### 3.1 Overall Architecture Diagram

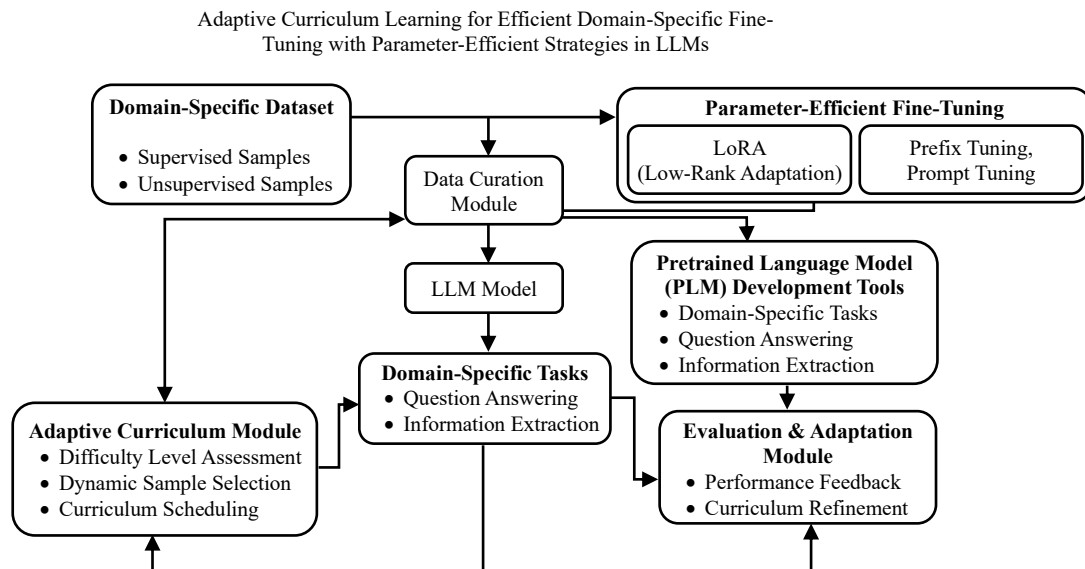


Figure 1: Overall architecture diagram about proposed methodology

The figure 1 diagram is an organized strategy of Adaptive Curriculum Learning to the efficient domain-specific fine-tuning using parameter-efficient strategies in Large Language Models (LLMs). It starts with a Domain-Specific Dataset, which is separated into supervised and unsupervised samples. Such samples are stored in the Data Curation Module, which organizes and prepares the data for the following steps. The main part of the process is the LLM Model, which is fine-tuned by

parameter-efficiently applying strategies such as LoRA (Low-Rank Adaptation) and Prefix Tuning & Prompt Tuning, expected to optimize the model and use less computational resources. The central component of this strategy is the Adaptive Curriculum Module, which dynamically adjusts task difficulty, selects appropriate samples for training, and schedules the curriculum to gradually enhance learning. This module ensures that difficult and easy tasks are presented in the model with equal frequency. Question Answering and Information Extraction are Domain-Specific Tasks that are specific to the domain of interest and are directed by PLM Development Tools. The last phase is the Evaluation and Adaptation Module, which gives constant performance feedback and makes amends on the curriculum according to the model progress. This adaptive system makes sure that the fine-tuning process gets improved over time so that the model can be effective in dealing with a large number of activities hence making the process of learning to be efficient and scalable.

### 3.2 Working Procedure for Parameter Efficient Fine-Tuning of LLM

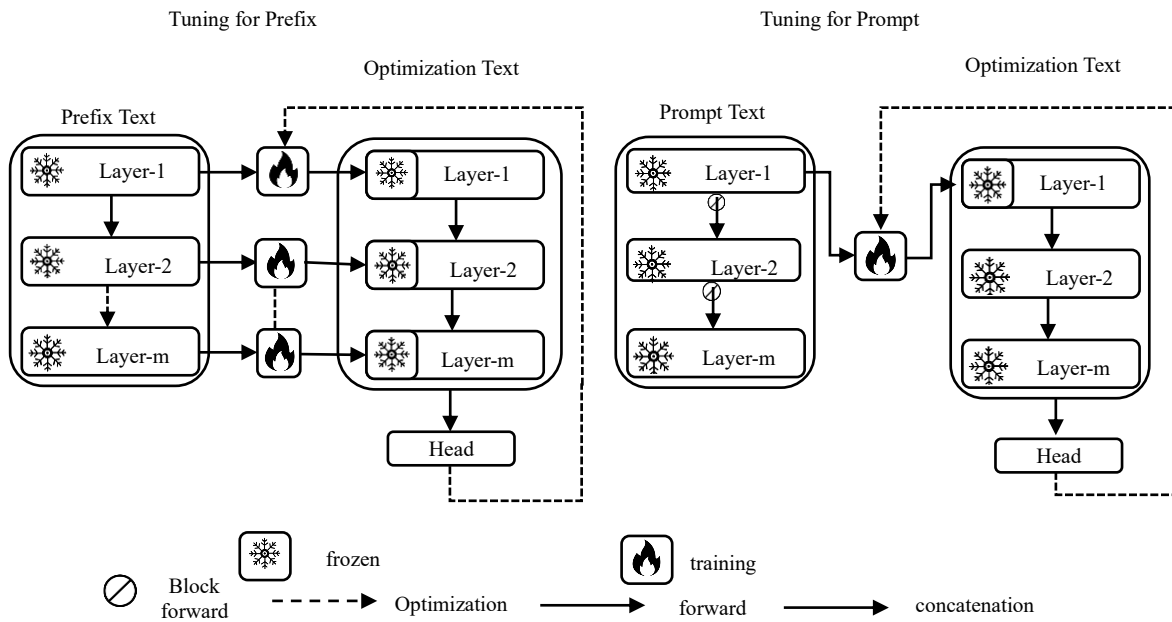


Figure 2: Working procedure for parameter efficient fine-tuning of LLM

Another way to tune the prefix and the Prompt is depicted in the diagram in figure 2. The essence of both approaches is to fine-tune only specific layers of a pretrained model and leave the rest frozen. Tuning for Prefix: The Prefix Text is treated in Tuning for Prefix, in which the layers between Layer-1 and Layer-m are frozen, and a trainable layer is added between the frozen layers and the output head. This optimization would attempt to optimize the trainable layer, as indicated by the dashed red lines, while the frozen layers would not be altered. Concatenation of the optimized layers with the other layers of the model occurs in the forward pass. A similar structure followed in Tuning for Prompt, with Prompt Text, is that in which some layers (Layer-1 to Layer-m) are frozen and others are optimized at the trainable layer. By doing so, the model can be dynamically adjusted to the immediate text, optimizing task-specific layers without retraining the pretrained weights. Both approaches focus on streamlining task-specific fine-tuning of the domain-specific model, optimizing particular components at the cost of efficient adaptation to the task and reduced consumption.

PREFIX and prompt tuning are techniques for relatively small-scale language models to adapt to particular tasks or datasets, with only a small number of updates to the model parameters. These methods

have become popular because of their efficiency and effectiveness, especially in situations where it is computationally cumbersome or even impractical to update the entire model.

### 3.2.1 Prefix Tuning

Prefix tuning involves a sequence of tunable vectors followed by a prefix to the input of each layer with the transfer model. Here is defined the transformer model function  $f$  should connect with the input sequence  $x$  to  $y$ . Output  $y = F(x)$ . Based on the prefix, tuning should be modified with the  $y = F(p \otimes x)$ , from the above  $p$  represents the prefix and  $\otimes$  denoted the concatenation. Mathematically, which is also considered about the transformer model with  $k$  layers perform the transformation of  $cf_l$ . The modified transformation among prefix considers as,

$$F'_k(p_k, x) = F_k(P_k \otimes x) \quad (1)$$

Equation (1) describes the  $p_k$ , is defined as the prefix layer of  $k$ . The prefixes also represent the  $\{P_1, P_2, \dots, \dots, P_K\}$ , which is a learnable parameter with an optimized value during training.

#### Algorithm -1 for Parameter Prefix tuning

*input: pretrained LLM,  $M$  with parameters  $\theta$ , during training*

*Input: prompt text  $p$ , dataset  $\{x_i, y_i\}_{i=1}^N$*

*initialize Trainable parameters  $\Phi$  and  $\xi$*

*for each input  $(x_i, y_i)$  do*

*$h^P \leftarrow M\Phi(p)$*

*$z \leftarrow f\Phi(h^P)$*

*$\gamma_i \leftarrow M\Phi(x_i, Z)$*

*$o_i \leftarrow \xi(\gamma_i)$*

*Compute loss  $L(o_i, y_i)$*

*update  $\Phi$  and  $\xi$  to minimize loss*

*end for*

#### Algorithm Explanation

Parameter Prefix Tuning is an algorithm that adjusts a pretrained Large Language Model (LLM) based on a set of trainable parameters to a task using a prompt text and dataset. Firstly, the fixed parameters  $\theta$  are loaded in the pretrained model  $M$  and trainable parameters  $F$  and  $x_i$  are initialized. To process every training example,  $x_i, y_i$ , the algorithm takes in the prompt-based response  $hp$ , by inputting the prompt  $p$  through the trained model  $M$  with the learnt parameters  $F$ . This output is further processed by taking a function  $f\Phi$  to come up with an intermediate representation  $z$ . The model then predicts the input  $x_i$ , Loss computation Loss is calculated as  $o_i$  and the actual label  $y_i$  and the trainable parameters  $\Phi$  and  $x_i$  are updated to reduce the loss. This is repeated with every single example in the dataset so that the model can optimize its parameters to effectively fit the task at hand and the prefix tuning allows the model to fine-tune to the task but still makes use of the prior knowledge of the model. This tends to minimize the quantity of retraining required by simply optimizing only some layers to the task at hand.

### 3.2.2 Prompt Tuning

Prompt tuning means leveraging the concept of natural language to generate the output combined with a specific task. With the mathematical terms, the pretrained model  $M$  and the objective of the optimal prompt as  $p$  model performance of the task as  $T$ . Which is maximized for the prompt used as input. Finally, the task of  $T$  which is have the set of tasks with specific examples as  $\{(x_i, y_i)\}$  used by the prompt to optimize the following,

$$P^* = \operatorname{argmax}_p \sum_i \log M(y_i | p \otimes x_i) \quad (2)$$

From the above equation (2) describes the objective function maximize the likelihood of correct output  $y_i$  Given the input as  $x_i$  concatenated with the optimal prompt as  $p^*$ . Unlike the prefix tuning, the prompt does not modify with the internal working of the model, which should contain the influence of the output through the input sequences.

#### Algorithm-2 for Prompt Tuning

*input: pretrained LLM,  $M$  with parameters  $\theta$ , during training*

*Input: prompt text  $p$ , dataset  $\{x_i, y_i\}_{i=1}^N$*

*initialize Trainable parameters  $\Phi$  and  $\xi$*

*for each input  $(x_i, y_i)$  do*

*$e_p = E(p) \sim M_\Phi$*

*$e_{x_j} = E(x_i) \sim M_\Phi$*

*$e'_p = g_\gamma(e_p) \in R^{l*d}$*

*compute loss  $L(0_i, y_i)$*

*update  $\gamma$  and  $\xi$  to minimize loss*

*end for*

#### Algorithm Explanation

The algorithm outlines a process for fine-tuning a pretrained Large Language Model (LLM) using a set of domain-specific inputs and a prompt text. Initially, the pretrained model, denoted by  $M$  with parameters  $\theta$ , is used alongside a predefined *prompt text* ( $p$ ) and a dataset of input-output pairs  $x_i, y_i$ . The model is initialized with trainable parameters ( $\Phi$  and  $\xi$ ), which will be adapted during the fine-tuning process. For each training sample, the model generates an embedding  $e_p$  for the prompt  $p$  and an embedding  $e_{x_j}$  for the input  $x_i$ , both processed using the trainable parameters  $\Phi$ . The prompt embedding  $e'_p$  is then passed through a transformation function  $g_\gamma$ , producing a modified embedding  $e'_p$ , which is designed to better suit the task. The model then generates an output from the input embedding and the transformed prompt, and the loss is computed by comparing the predicted output with the true target  $y_i$ . Finally, the algorithm updates the trainable parameters  $\gamma$  and  $\xi$  to minimize the loss, iteratively improving the model's performance on the task at hand. This is done in harmony with all the input-output pairs of the dataset until the model becomes adjusted to the exact task of the particular domain, whilst maintaining the learned knowledge. The approach exploits parameter-efficient strategies for fine-tuning

the model, ensuring that the fine-tuning process targets only specific components of the model and saving on computational cost while still adapting the model to the new task.

### 3.3 Adaptive Curriculum Learning Through Parameter Efficient Fine-Tuning (ACLPF)

The aim to improve the performance of a policy model  $\pi_\theta$  for solving mathematical problems through adaptive curriculum learning. Fine-tuning on problems that are too easy or too hard leads to poor learning outcomes. Instead, the model should be trained on problems whose difficulty is close to the model's current capability. We frame this as an adaptive curriculum learning problem and propose ACL-PEFT-LLM, which adaptively adjusts the target difficulty to keep training problems within a suitable difficulty range. ACLPF is compatible with a variety of RL algorithms (e.g., GRPO, PPO); in this work, we instantiate it with PPO and refer to this variant as ADARFT (PPO). Let  $D$  be a dataset of mathematical problems; each annotated with a precomputed difficulty score. The score can be either human-annotated or model-estimated. The objective is to train a policy  $\pi_\theta$  that improves its problem-solving ability by dynamically adjusting the training curriculum according to the model's current performance. Our proposed algorithm, ADARFT, is shown in Algorithm 3.

#### Algorithm – 3 for Adaptive Curriculum in Efficient Fine Tuning

*input: Data source  $D$  with difficulty scores  $\{d_i\}$ , policy model  $\pi_\theta$ ,  
reward function  $R$ , batch size  $B$ , initial target difficulty  $T$ , step size  $\eta$   
sensitivity  $\alpha$ , target reward  $\beta$ , difficulty bounds  $d_{min}, d_{max}$   
Select Parameter efficient fine tuning (LoRA, prefix tuning, Prompt tuning)  
while training is not finished do  
compute absolute differences from target difficulty  
 $\Delta_t = |d_i - T| \forall i \in \{1, 2, \dots, |D|\}$   
sort and select top  $B$  samples closest to target difficulty  $X \leftarrow \{S_1, S_2, \dots, S_B\}$   
Generate responses using policy model  $G = \pi_\theta(X)$   
Compute average reward:  $R_{avg} \leftarrow \frac{1}{|x|} \sum_{i=1}^{|x|} R(X_i, G_i)$   
Update policy  $\pi_\theta \leftarrow A(\pi_\theta, X, G, R)$   
Update and clip target difficulty  $T' \leftarrow clip(T + \eta \cdot \tanh(\alpha \cdot (R_{avg} - \beta)), d_{min}, d_{max})$   
update sample  $T \leftarrow T'$   
end while*

#### Algorithm Explanation

The Parameter-Efficient Fine-Tuning with Target Difficulty Adjustment algorithm aims to fine-tune a policy model  $\pi_\theta$  using a dataset with difficulty scores and optimize it to a specific difficulty. The procedure starts by determining the absolute differences between the difficulty scores of each sample and the target difficulty  $T$ . The samples that differ the least are selected (batch size  $B$ ) and fed into the policy model to produce responses. The responses are evaluated using the reward function, and the

average reward is calculated. The target difficulty  $T$  is then adjusted based on the average reward and a sensitivity parameter  $\alpha$ , with the new target clipped within predefined difficulty bounds  $d_{min}, d_{max}$  to avoid extreme values. The algorithm uses parameter-efficient fine-tuning methods such as LoRA, Prefix Tuning, and Prompt Tuning to fine-tune only the relevant parts of the model, thereby reducing resource consumption. This process iterates until the training goal is met, allowing the model to adaptively focus on tasks with difficulty levels close to the target, improving task-specific performance while maintaining efficient resource usage.

### 3.3.1 Dynamic Curriculum Sampling

To construct the adaptive curriculum, the target difficulty was defined as  $T$  which also represents the current target as a challenging training level. ACL-PEFT-LLM dynamically adjusts  $T$  based on the model of reward signal to maintain the optimal difficulty level for learning. The algorithm computes the absolute difference between the target and each dataset.  $\Delta_t = |d_i - T| \forall i \in \{1, 2, \dots, |D|\}$ . This batch of the training problem was formed by selecting the  $B$  problem with the smallest values  $\Delta_i$ , producing the batch  $X = \{S_1, S_2, \dots, S_B\}$ . It also ensures the selected problems closest to the current difficulty are focused on in the learning process, making the problem as hard as possible.

### 3.3.2 Policy Update

The selected batch  $X$  used as to train the policy model of  $\pi_\theta$ , which is also generate the responses as  $G = \pi_\theta(X)$ . The reward signal is computed based on the correctness of the model output.  $R_i = 1$ , if the response as correct and  $R_i = 0$  means the response is incorrect. The average reward of the batch should be computed as  $R_{avg} = \frac{1}{|X|} \sum_{i=1}^{|X|} R(X_i, G_i)$ . This policy should be updated with a parameter finetuning followed by *LoRA, prefix tuning, Prompt tuning*

### 3.3.3 Target Difficulty Update

To adapt the curriculum dynamically, the target difficulty is updated based on the average reward. If the model performs well on the current difficulty level (high reward), the target difficulty increases, making the training problems harder. Conversely, if the model performs poorly, the target difficulty decreases. This dynamic update mechanism lies at the core of the ACL-PEFT-LLM curriculum adaptation strategy.

$$T' = clip \left( T + \eta \cdot tanhh \left( \alpha \cdot (R_{avg} - \beta) \right), d_{min}, d_{max} \right) \quad (3)$$

Equation (3) describes the  $\eta, \alpha, \beta$  should represent the hyperparameters.  $\eta$  defined the step size among adjust the target difficulty and  $\alpha$  defined the control sensitivity update and  $\beta$  Is the target reward level supposed to represent the desired outcome of large deviations.  $d_{min}, d_{max}$  should represent the valid ranges. These bounds are manually specified and derived by the training outcomes.

### 3.4 Data Flow Diagram of Algorithmic Framework

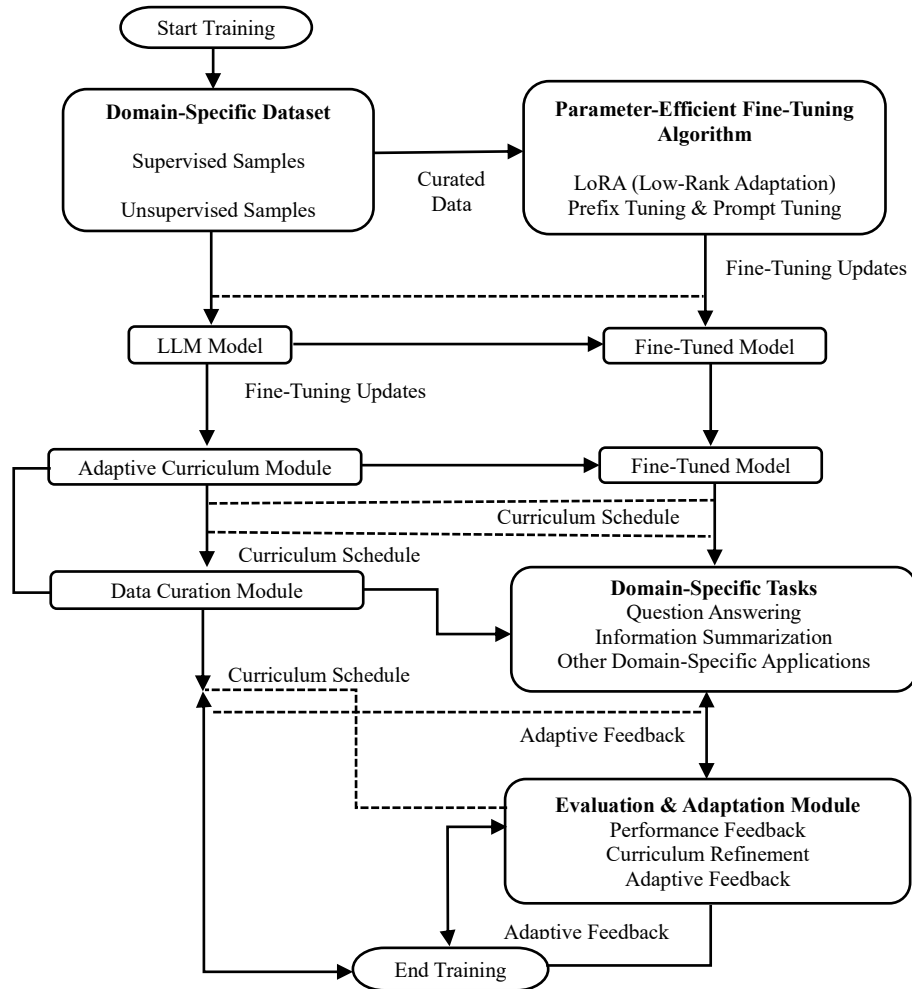


Figure 3: Data flow diagram for algorithmic framework

The diagram in figure 3 shows the organized training of a domain-specific fine-tuning of a Large Language Model (LLM) using a mixture of parameter-efficient fine-tuning and adaptive curriculum learning techniques. The domain-specific dataset is considered the starting point of the process; it includes supervised and unsupervised samples. The data curation module cleans and processes the data, which is then passed to the adaptive curriculum module, which adjusts the difficulty level of tasks based on the model's performance, ensuring a gradual increase in task complexity. It is followed by a parameter-efficient fine-tuning algorithm, whereby techniques such as LoRA (Low-Rank Adaptation), Prefix Tuning, and Prompt Tuning are used to update a small number of the model parameters, resulting in a lower computational cost and at the same time, allowing an efficient adaptation to the particular domain. The fine-tuning updates generate a model tailored to the presented domain-specific tasks, e.g., question answering, information summarization, and other specialized applications. This process provides the model with adaptive feedback, which is applied to improve the model's performance and the fine-tuning process. The model is then tested and refined through performance feedback and curriculum refinement after fine-tuning, so it keeps getting better over time. This enables efficient domain-specific adaptation through parameter-efficient fine-tuning and adaptive curriculum learning, allowing the model to perform complex tasks with minimal resource consumption. All processes are

aimed at achieving optimal fine-tuning of the model and ensuring high performance without compromising computational efficiency.

### 3.5 Algorithm-4 For Proposed Methodology

1. Step 1: Initialize Pre – trained LLM
2. *Initialize model (pre – trained LLM)*
3. *Load tokenizer and pre – trained weights for the model*
4. *step: 2 load domain specific dataset*
5. *load domain specific dataset (supervised and unsupervised samples)*
6. *split dataset into training, validation and test sets*
7. *step: 3 data preprocessing*
8. *for each dataset split*
9. *tokenize and preprocess data using the tokenizer*
10. *apply padding and truncation to ensure consistent input size*
11. *step: 4 define curriculum stages*
12. *define curriculum stages*
13. *stage: 1 general domain knowledge*
14. *stage: 2 domain specific but easier tasks*
15. *stage: 3 advaced and specific domain tasks*
16. *step: 5 Apply parameter efficient fine tuning*
17. *select a parameter efficient fine tuning (PEFT) method*
18. *option: 1 low rank Adaptation (LoRA)*
19. *Option: 2 Prefix Tuning*
20. *option: 3 Prompt Tuning*
21. *step: 6 Adaptive Curriculum learning process*
22. *for each curriculum stage (stage1 → stage 2 → stage 3)*
23. *set learning rate based on stage*
24. *fine tune model using current curriculum stateg data*
25. *train the model for a predicted number of epochs*
26. *evaluate model performance on validation dataset*
27. *if stage = stage 1;*
28. *set learning rate to higher value*
29. *else if stage = stage 2*
30. *set learning rate to medium value*
31. *else if stage = stage 3*

32. *set learning rate to lower value*
33. *Intermediate model performances(accuracy, precision, recall)*
34. *Evaluate model performance on validation data and log results*
35. *step: 7 dynamic curriculum scheduling and refinement*
36. *if model performance improves significantly*
37. *continue with the next stage of curriculum*
38. *else provide feedback to adapt training strategy based on performance*
39. *step: 8 Final evaluation*
40. *Evaluate the model on the test set*
41. *calculate final performance metrics (accuracy, F1 score)*
42. *step: 9 save fine tuned Model*

### 3.5.1 Algorithm Explanation

The proposed algorithm (ACL-PEFT-LLM) presents a systematic procedure for fine-tuning a pre-trained Large Language Model (LLM) using an adaptive curriculum learning strategy and parameter-efficient fine-tuning (PEFT) methods. The second step is to load a dataset for a specific domain, which includes both supervised and unsupervised samples, and then split it into training, validation, and test sets. Next comes data preprocessing, during which each data split is tokenized, and padding and truncation are applied so that all data provided to the model has the same size. The algorithm then outlines the stages of the curriculum, which start with general domain knowledge in Stage 1, proceed to domain tasks and an easier level in Stage 2, and then to more advanced, more specific domain tasks in Stage 3. Step 5: A PEFT approach among Low-Rank Adaptation (LoRA), Prefix Tuning or Prompt Tuning is chosen and is intended to explicitly adjust the model and achieve high accuracy without major model re-training. The learning rate also varies in relation to the stage during the adaptive curriculum learning process: it is higher in Stage 1 to cover more general knowledge, medium in Stage 2 to cover more specific tasks, and less in Stage 3 to cover finer aspects of complex tasks. The performance of the model regarding accuracy, precision and recall is assessed at every level on the validation dataset. In case of an improvement in performance, the training is continued to the next level, otherwise feedback is provided to handle the training strategy. Once the curriculum stages are done, the model is last assessed on the test set, whose performance measures such as accuracy and F1 score are computed. Lastly, the fine-tuned model is stored to be used in the future. This sequential strategy makes the model go through easy to complex tasks and dynamically it readjusts itself through performance at any given level thus maximizing its learning efficiency.

## 4 Results and Discussion

### 4.1 Dataset Description

The AIME (Automated Image-to-Text Model Evaluation) dataset is designed to evaluate a set of automated image-to-text models in a medical context. It includes medical pictures, which are radiology scans and relevant textual commentaries, including diagnostic reports and radiology results. The best use of this multimodal data is in activities such as image captioning and producing radiology reports from medical images, which bridge the gap between image recognition and natural language processing.

The dataset enables models to be specialized in specific medical fields such as radiology or pathology, and is hence especially applicable in Stage 3 of adaptive curriculum learning, where the model is specialized in complex domain-specific tasks. In turn, SQuAD (Stanford Question Answering Dataset) is a popular dataset for evaluating question answering (QA) models, with reading comprehension as its primary focus. It includes more than 100,000 pairs of questions and answers across More than 500 articles, with the aim of measuring a model's capacity to answer factual questions given a specific context paragraph. SQuAD2.0 introduces the feature of unanswerable questions, where not only relevant information needs to be retrieved but also it is necessary to know that no information is provided. This renders SQuAD suitable for Stage 2 of adaptive curriculum learning, where the model advances from responding to simpler questions to more difficult domain-specific ones. Combining the AIME and SQuAD datasets poses a range of challenges for domain-specific fine-tuning of parameter-efficient strategies, exposing the model to a variety of challenges, both general understanding and specialized problem-solving, in fields such as healthcare.

#### 4.2 Difficult Distribution for Different Training Sets Uniform, Skew Difficult, and Skew Easy Contain 10,000 Samples Using the AIME Dataset

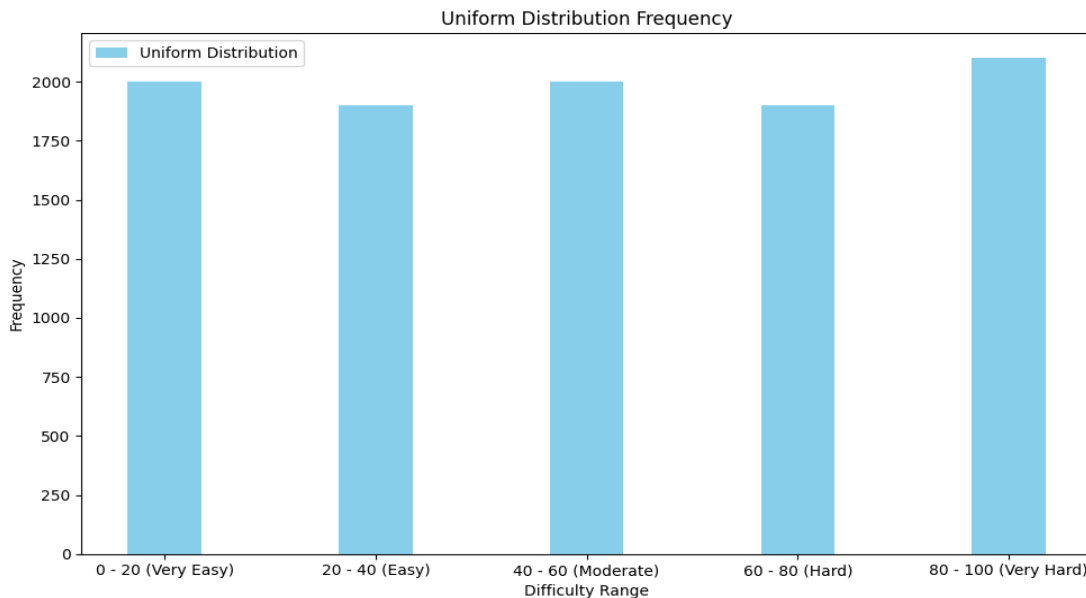


Figure 4: Uniform distribution for different training sets contains 10,000 samples

The Uniform Distribution Frequency of the various difficulty ranges (Very Easy, Easy, Moderate, Hard, and Very Hard) for a training dataset of 10,000 samples from the AIME Dataset is shown in the figure 4. In this distribution, the number of samples in each range of difficulty (0-20 (Very Easy), 20-40 (Easy), 40-60 (Moderate), 60-80 (Hard), and 80-100 (Very Hard)) is equal, and is around 2,000 samples each. This means the samples are well spread across all levels of difficulty, without favoring easier or more challenging tasks. The homogeneity provides the model with an equal amount of challenge during training, which can potentially enhance its capacity to generalize to new levels of difficulty. This data distribution, by ensuring that all difficulty levels are equally represented, is especially helpful when you want to train a model capable of handling a wide range of input complexities (not biased toward a single difficulty level). This arrangement can be advantageous in situations with a high degree of variation in task levels, as the model can adapt to different situations.

### 4.3 Skew Difficult Distribution Frequency Different Training Set Contains 10,000 Samples

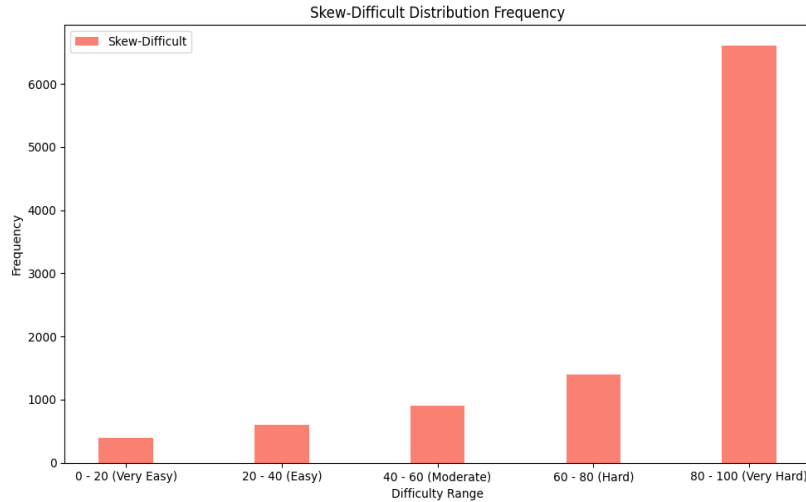


Figure 5: Skew difficult distribution frequency different training set contains 10,000 samples

Figure 5 above shows the Skew-Difficult Distribution Frequency for a training set of 10,000 samples from the AIME Dataset. The bias of this distribution is evident in the difficulty range of Very Hard (80-100), which has more than 6,000 samples. Conversely, other difficulty ranges, namely, Very Easy (0-20), Easy (20-40), Moderate (40-60), and Hard (60-80) are much less represented, and there are fewer than 1,000 samples in each of the former. Such an imbalanced distribution implies that the dataset is mostly composed of hard tasks, thereby exposing the model to many hard examples. As much as this arrangement can help models be trained to perform well on hard-to-solve problems, it poses a problem of underfitting easier tasks, as there are few simpler difficulty samples. When the aim is to optimize the model's performance on high-complexity tasks, the Skew-Difficult distribution is desirable, though care should be taken to prevent the model from focusing on simpler tasks. This kind of data is good when the specialized domain or task is of interest and is, by its nature, challenging and may fail to generalize to easier and more diverse tasks.

### 4.4 Skew Easy Distribution Frequency Different Training Set Contains 10,000 Samples

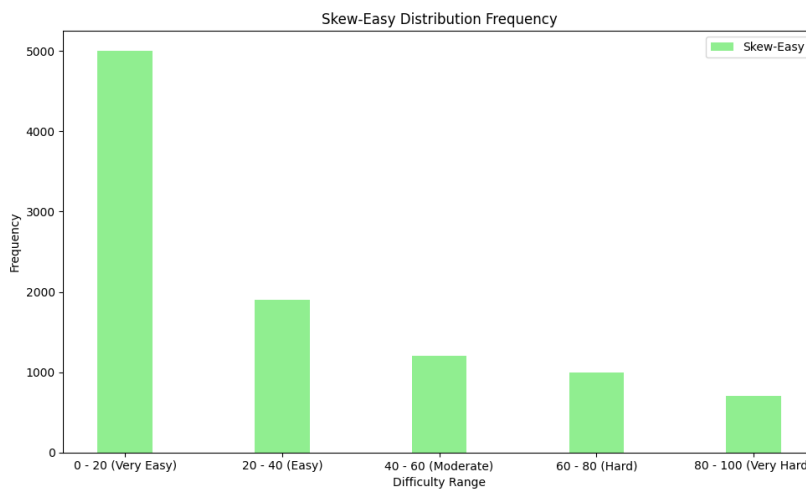


Figure 6: Skew easy distribution frequency different training set contains 10,000 samples

Figure 6 shows the Skew-Easy Distribution Frequency for a training dataset of 10,000 samples from the AIME Dataset. The distribution indicates a large concentration of samples in the Very Easy (0-20) range, with more than 5,000 samples, and in the Easy (20-40) range, with approximately 2,000 samples. There are progressively fewer samples in the Moderate (40-60), Hard (60-80), and Very Hard (80-100) ranges, with fewer than 1,000 in each. The data is also highly biased towards simpler tasks, so that the model will be largely exposed to simpler examples during training. Although such an arrangement is advantageous for maximizing performance on simpler problems, it can cause problems when the model encounters more challenging ones, because it has reduced exposure to such examples. The distribution is especially convenient when optimization for simpler tasks is desired, but may lead to significant generalization failure in more complex situations. To enhance model performance across a wider range of challenges, it may be necessary to balance the dataset by including harder examples.

#### 4.5 Software and Hardware Configuration

Table 1: Software and hardware configuration

Category	Details
Hardware Configuration	
GPU	NVIDIA A100 or V100 GPUs for model training and fine-tuning
CPU	High-end processors like Intel Xeon or AMD EPYC for computational overhead and data preprocessing
RAM	64 GB or more for handling large-scale datasets and model parameters
Storage	SSD storage, typically 1TB or more for storing large datasets and model results
Software Configuration	
Operating System	Linux-based systems (Ubuntu 20.04 or later) for high-performance deep learning tasks
Deep Learning Frameworks	TensorFlow, PyTorch, Hugging Face Transformers for LLM implementation and fine-tuning
Libraries	- CUDA, cuDNN for GPU acceleration - NVIDIA Apex for mixed precision training - Hugging Face Datasets - Optuna or Ray Tune for hyperparameter optimization - scikit-learn, NumPy, Pandas for data manipulation and preprocessing
Training Setup	Parameter-efficient fine-tuning methods like LoRA, Prefix Tuning, or Prompt Tuning for optimizing model resources

Table 1 above shows that it proposes a methodology for fine-tuning large language models (LLMs), which uses a powerful hardware and software setup to maximize computational efficiency and performance. The hardware will be based on NVIDIA A100 or V100 GPUs, which are essential for computationally intensive tasks during model training and fine-tuning. High-end Intel Xeon or AMD EPYC processors support these GPUs by handling the overhead computation and data preprocessing. The system has 64 GB of RAM, which means it can process large datasets and handle large model parameters, and 1 TB or more of SSD storage is provided to access datasets and model outputs quickly. On the software side, the system works with Linux-based operating systems (e.g., Ubuntu 20.04 and above), providing a stable, high-performing environment for deep learning tasks. The deep learning models employed are TensorFlow, PyTorch, and Hugging Face Transformers, which are essential for implementing and fine-tuning LLMs. The setup also includes some of the main libraries, such as CUDA and cuDNN, to run applications on a graphics card, NVIDIA Apex to support mixed-precision training, Hugging Face Dataset to manage data effectively, and Optuna or Ray Tune to optimize hyperparameters. Also, scikit-learn, NumPy, and Pandas are used to process and manipulate data. Most of the parameter-efficient techniques used in fine-tuning are LoRA, Prefix Tuning, and Prompt Tuning, which minimize computational cost while maintaining the same model performance. This architecture

guarantees that the model can learn effectively from domain-specific datasets, including medical, legal, and financial domains, and that resource consumption is manageable.

#### 4.6 Parameter Initialization

Table 2: Parameter initialization

Parameter	Description	Suggested Initialization
Pre-trained Model	Pre-trained Large Language Model (LLM)	Load pre-trained model weights (e.g., GPT-3, BERT, T5)
Tokenizer	Tokenizer for preprocessing text	Load pre-trained tokenizer for the selected LLM
Trainable Parameters	Parameters to be fine-tuned during the process	$\Phi, \eta$ Random initialization or from pre-trained model with low rank values
Curriculum Stages	Stages of the adaptive curriculum learning	Stage 1: General domain knowledge (0-30) Stage 2: Easier domain-specific tasks (30-60) Stage 3: Advanced domain tasks (60-100)
Learning Rate ( $\eta$ )	Rate of learning adjustment for each stage	Stage 1: 0.001 Stage 2: 0.0005 Stage 3: 0.0001
Sensitivity ( $\alpha$ )	Sensitivity to target reward and difficulty adjustments	0.5 (based on empirical results, can be tuned)
Target Reward ( $\beta$ )	Desired performance target for the model	0.8 (can vary based on task complexity and dataset)
Difficulty Bounds	Minimum and maximum bounds for target difficulty	$d_{\min} = 0, d_{\max} = 100$ (task difficulty level range)

The table 2 presents the parameterization of the fine-tuning of the pre-trained Large Language Model (LLM) based on Adaptive Curriculum Learning with parameter-efficient methods. The parameters to be trained ( $\Phi$  and  $\xi$ ) can be randomly or low-rank based on the already trained model in order to effectively fine-tune particular aspects of the model. The learning process is based on a structured curriculum that is split into three stages: Stage 1 is based on general domain knowledge covering the difficulty range of 0-30, Stage 2 is based on easier domain-specific tasks covering the difficulty range of 30-60, and Stage 3 is focused on more advanced domain tasks covering the difficulty range of 60-100. Learning rate ( $\eta$ ) is also modified: Stage 1: 0.001, Stage 2: 0.0005, Stage 3: 0.0001 so that, initially, the model learns very fast and then gradually slows down as it gets more capabilities. The sensitivity parameter ( $s$ ) will be set to 0.5, based on empirical evidence, and can be adjusted to meet specific training requirements, regulating the degree to which the target difficulty varies with model performance. The target reward ( $\beta$ ) will be 0.8, which can be adjusted based on the complexity of the task and data. Lastly, the difficulty limits are set to  $d_{\min} = 0, d_{\max} = 100$ , making the target difficulty feasible for efficient learning. These starting values will allow a systematic way of fine-tuning the model as it reaches a gradual progress of performance at different levels of task difficulty.

#### 4.7 Metric Evaluation

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total Number of Predictions}} = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

The above equation (4) describes the  $TP$  as True Positives,  $TN$  as True Negatives,  $FP$  as False Negatives and  $FN$  as False Negatives.

### 4.7.1 F1 Score

Precision and Recall have a harmonic mean called F1 Score. It also strikes a balance between the two, giving only one measure of how the model performs in equation (5).

$$F1\ Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (5)$$

### 4.7.2 Performance Efficiency Trade Off (AER)

$$AER = \frac{Task\ Score\ (normalized)}{\log(Trainable\ Params * VRAM * Training\ Time)} \quad (6)$$

The average score in the dataset (e.g., accuracy, F1 score) is expressed in the form of equation (6) above. The efficiency metric is called AER (Better), which is the computational efficiency of the model (e.g., memory usage or training time).

## 4.8 Model Performance Based on Different Academic Curriculum

Table 3: Model performance based on different academic curriculum

Model	Setup	Method	GSM8K	MATH 500	Olympiad Bench	Minerva Math	AMC23 (Avg@8)	AIME24 (Avg@8)	Average
Qwen 2.5 Math 1.5B	Skew-Difficult	PPO	69.67	64.60	20.65	12.87	47.50	9.17	37.41
		ADARFT	74.00	66.40	20.36	15.07	55.00	12.08	40.48
		ACL-PEFT-LLM	78.00	68.25	21.25	18.06	56.00	13.08	42.58
	Uniform	PPO	71.95	65.20	21.10	15.81	42.50	6.67	37.20
		ADARFT	74.53	66.20	21.99	14.34	57.50	12.08	41.11
		ACL-PEFT-LLM	76.85	68.45	23.02	16.25	58.25	18.65	43.25
	Skew easy	PPO	72.71	67.40	19.17	13.97	45.00	12.50	38.46
		ADARFT	73.62	66.20	19.91	13.97	55.00	9.17	39.18
		ACL-PEFT-LLM	75.64	68.45	21.25	15.56	58.00	10.15	42.13
Qwen 2.5 7B	Skew-Difficult	PPO	89.69	71.20	23.33	23.53	50.00	11.25	44.17
		ADARFT	90.98	71.40	25.85	22.43	52.50	15.83	46.83
		ACL-PEFT-LLM	92.98	72.25	27.85	24.56	54.56	18.02	48.95
	Uniform	PPO	89.31	72.40	23.63	25.37	42.50	15.00	44.70
		ADARFT	90.14	72.60	24.96	24.26	55.00	14.58	46.92
		ACL-PEFT-LLM	92.16	74.80	26.98	26.46	57.25	16.54	48.99
	Skew easy [22]	PPO	89.39	73.60	23.33	24.26	47.50	13.33	45.07
		ADARFT	90.14	72.60	25.56	23.16	50.00	14.17	45.94
		ACL-PEFT-LLM	92.16	74.65	26.85	24.18	52.25	16.18	47.96

The table 3 shows the results of both models Qwen 2.5 Math 1.5B and Qwen 2.5 7B in various training configurations (Skew-Difficult, Uniform, Skew-Easy) and models (PPO, ADARFT, and

ACL-PEFT-LLM) on several datasets, such as GSM8K, MATH 500, Olympiad Bench, Minerva Math, AMC23 (Avg@8), and AIME24 (Avg@8). On the whole, the outcomes demonstrate that the Skew-Difficult structure is consistently the most successful, especially for the Qwen 2.5 7B model. ACL-PEFT-LLM is the most successful, with an average score of 48.95, followed by ADARFT (46.83) and PPO (44.17). The best performance on the Qwen 2.5 Math 1.5B also occurs with ACL-PEFT-LLM in the Skew-Difficult setup, with an average of 42.58, which is better than PPO (37.41) and ADARFT (40.48). The overall performance is generally lower but significantly high in the Uniform setup, with the performance of ACL-PEFT-LLM rated at 43.25 on average in the case of the 1.5B model and 48.99 on average in the case of the 7B model. ACL-PEFT-LLM's performance drops slightly in both setups, but with the 1.5B model, the Skew-Easy setup achieves 42.13, and with the 7B model, the Skew-Easy setup achieves 47.96. In all configurations and techniques, ACL-PEFT-LLM is always the most dominant in PPO and ADARFT, and the bigger Qwen 2.5 7B model performs much higher scores in all distributions, particularly in the Skew-Difficult configuration. This is to indicate that the size of the model and the training configuration significantly influence the performance, and Skew-Difficult offers more challenging data, which can be used to enhance the performance of the model.

#### 4.9 Efficiency Comparison of Parameter-Efficient Fine-Tuning Techniques

Table 4: Efficiency comparison of parameter-efficient fine-tuning techniques

Techniques	GPU Memory Reduction (%)	Training Time Reduction (%)
ACL-PEFT-LLM	88%	195%
LoRA	85%	175%
Adapters	50%	90%
Prefix/Prompt Tuning	50%	100%
BitFit [21]	25%	45%

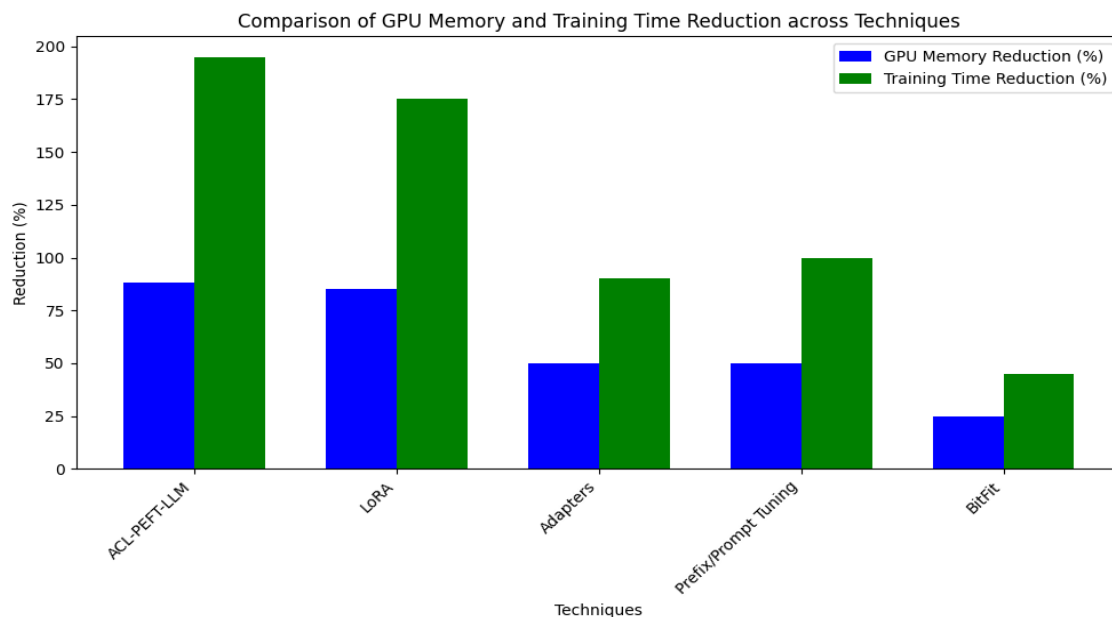


Figure 7: Comparison of GPU memory and training time reduction across techniques

In figure 7 and table 4 compare the results of the five methods in terms of GPU Memory Reduction (%) and Training Time Reduction (%) to include ACL-PEFT-LLM, LoRA, Adapters, Prefix/Prompt

Tuning, and BitFit. The findings reveal that ACL-PEFT-LLM offers the best outcome of reduction in both the GPU memory (88) and training time (195) which demonstrates that it is the most efficient methodology in regard to the memory consumption as well as training time. LoRA comes next with an 85% reduction in the memory of the GPU and a 175% reduction in training time, which also proves to be highly beneficial. Moderate declines in both measures are observed with Adapters and Prefix/Prompt Tuning, with the former reducing its GPU memory by 50% and the former reducing its training time by 90% and 100% respectively. Finally, BitFit is least affected by reduction in that it has only 25% reduction in memory of the GPU and 45% reduction in the time of training. Altogether, these findings indicate that the methods such as ACL-PEFT-LLM and LoRA are quite efficient in optimizing the use of GPS memory and training time whereas BitFit has rather less advantages in comparison.

#### 4.10 Resource Efficiency Trade Off

To compare the training cost, memory consumption, and trainable parameter referred in the above table 5 and figure 8

Table 5: Resource efficiency trade off

Method	Trainable Params (%)	Peak VRAM (GB)	Training Time (hrs.)	Inference latency (ms)
ACL-PEFT-LLM	99%	48.2	7.2	210
Full FT	98%	46.2	6.5	220
LoRA	83%	11.8	2.1	240
Adapters	31%	14.3	2.8	250
Prefix Tune	12%	9.7	1.6	270
BitFit [20]	9%	8.2	1.3	230

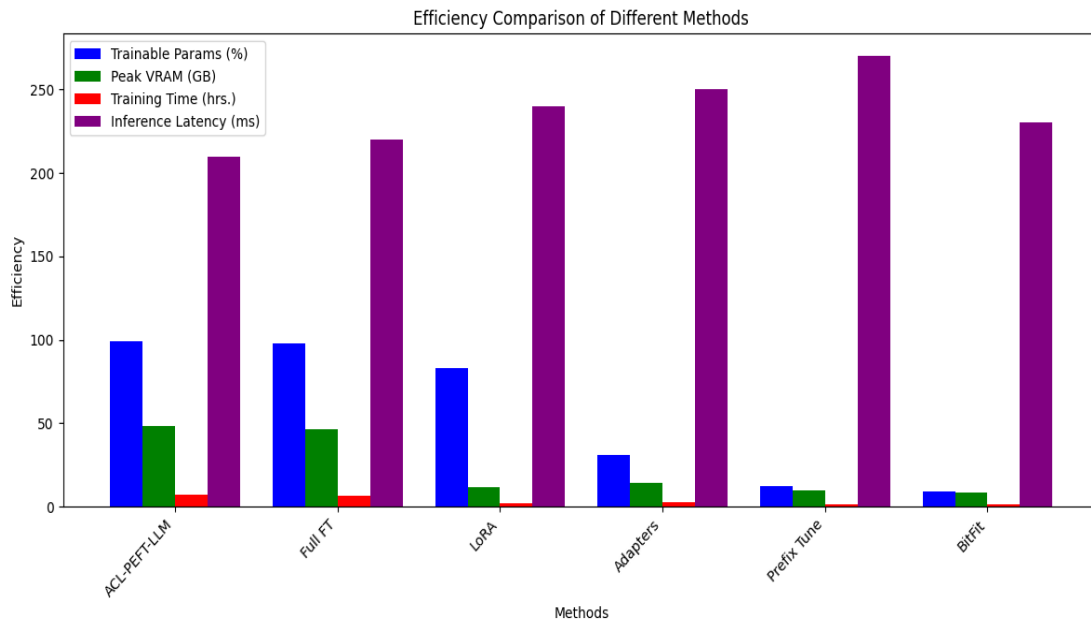


Figure 8: Energy comparison of different methods

The comparison of the resource efficiency of various methods in terms of the trainable parameters, peak VRAM utilization, training time, and inference latency is provided in the figure 8 and table 5.

ACL-PEFT-LLM makes use of 99 percent of the trainable parameters and model peaks at 48.2 GB of VRAM, training it in 7.2 hours and incurring a latency of 210 ms inference, and is one of the most resource-hungry. Full FT is close behind with 98% of the parameters, 46.2 GB of peak VRAM and a slightly lower training time (6.5 hours) and inference latency (220 ms). Adapters and Prefix Tune also produce significantly reduced trainable parameters of 31% and 12 each with comparatively low training times (2.8 hours and 1.6 hours, respectively) and higher inference latencies (250 ms and 270 ms, respectively). BitFit is the least trainable with 9 percent, 8.2 GB of VRAM, 1.3 hours of training time and 230 ms of inference latency. The comparison between these strategies indicates that LoRA and Adapters lower VRAM consumption and training time but at the expense of a little greater inference latency. However, ACL-PEFT-LLM and Full FT are more comprehensive in the training provide, at the cost of much more resources, in both VRAM and training time.

#### 4.11 Task Performance Comparison of Various Dataset

Table 6: Task performance comparison of various dataset

Method	SST-2 Accuracy	SQuADF1	SQuADEM	CNN/DMROUGE-L
ACL-PEFT-LLM	96.8%	91.5%	90.2%	75.8%
Full FT	94.6%	89.2%	83.1%	41.3%
LoRA	93.7%	88.4%	82.5%	40.7%
Adapters	93.1%	87.9%	81.6%	40.2%
Prefix Tune	91.5%	86.1%	79.4%	39.3%
BitFit [20]	90.2%	83.6%	77.4%	37.1%

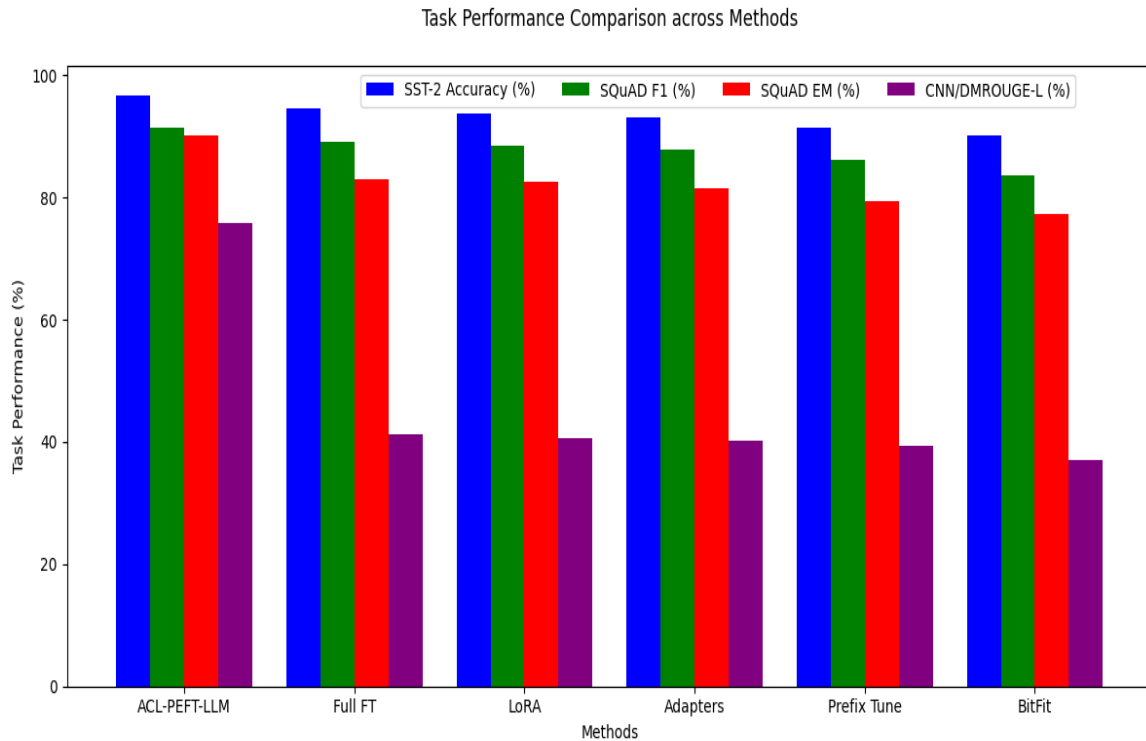


Figure 9: Task performance comparison across the various methods

The comparison of task performance of various methods, on various datasets, such as SST-2 Accuracy, SQuAD F1, SQuAD EM, and CNN/DMROUGE-L, is given in the figure 9 and table 6. ACL-PEFT-LLM has the best performance, 96.8% SST-2 Accuracy, 91.5% SQuAD F1, 90.2% SQuAD EM, and 75.8% CNN/DMROUGE-L, which implies that it is the best among all the methods. Full FT is the next with 94.6% in SST-2 Accuracy and 89.2% in SQuAD F1, yet the results are much lower in SQuAD EM (83.1%) and CNN/DMROUGE-L (41.3%), which indicates that it is not so efficient in these domains. LoRA and Adapters are similarly performing, though LoRA is higher in the Accuracy of SST-2 (93.7) and the F1 of SQuAD (88.4), however, lower in SQuAD EM (82.5) and CNN/DMROUGE-L (40.7). Prefix Tune and BitFit have a worse overall performance with Prefix Tune scoring 91.5% in SST-2 Accuracy and 86.1% in SQuAD F1, and significantly lower on SQuAD EM (79.4% in Prefix Tune) and CNN/DMROUGE-L (39.3% in Prefix Tune and 37.1% in BitFit). Overall, ACL-PEFT-LLM has the best results at the time of task performance, especially on SST-2 Accuracy and SQuAD F1, whereas BitFit has the poorest ones, especially on SQuAD EM and CNN/DMROUGE-L. It illustrates the complexity-performance trade-off of models, where more complex models such as the ACL-PEFT-LLM can work better on all the tasks, whereas simpler models, such as the BitFit, can demonstrate fewer abilities.

#### 4.12 Accuracy Efficiency Ratio

Table 7: Accuracy efficiency ratio

Method	Avg Task Score	AER (Better)
ACL-PEFT-LLM	0.99	5.12
LoRA	0.91	4.82
Adapters	0.89	4.12
Prefix Tune	0.85	4.68
BitFit	0.81	5.10
Full FT [21]	0.94	2.01

The table 7 provides the comparison of the performance of various methods according to two metrics: Average Task Score and AER (Better). The best Avg Task Score of 0.99 and AER (Better) of 5.12 of ACL-PEFT-LLM shows a high level of performance in terms of working with the tasks and efficiency in managing the data. LoRA has a good Avg Task Score of 0.91 and AER (Better) of 4.82 with only modest performance though slightly less efficient than ACL-PEFT-LLM. Adapters and Prefix Tune have fairly good performance in terms of Avg Task Scores of 0.89 and 0.85, respectively, though AER (Better) scores are significantly lower (4.12 in both cases), indicating a lower level of efficiency than both LoRA and ACL-PEFT-LLM. BitFit has a comparable Avg Task Score of 0.81 and AER (Better) of 5.10 that is similar to ACL-PEFT-LLM, but its performance is relatively low as compared to the others in terms of task score. Full FT records the lowest Avg Task Score of 0.94 but extremely low AER (Better) of 2.01 implying that it is scoring higher in tasks, but its effectiveness in dealing with tasks is far much less than the other techniques. Overall, it can be seen that ACL-PEFT-LLM is the most balanced and efficient technique and has higher task performance and efficiency than Full FT, which has a good task score and lower efficiency.

### 4.13 Ablation Study Analysis

Table 8: Ablation study results for ACL-PEFT-LLM and comparison with alternative models and configurations

Configuration	Accuracy	Training Time (hrs.)	GPU Memory (GB)	F1 Score	Inference Latency (ms)
Full Model (ACL-PEFT-LLM)	96.8%	7.2	48.2	96.2%	210
No Curriculum Learning	93.5%	7.5	50.1	93.0%	220
No PEFT (Full Fine-Tuning)	94.6%	12.0	75.0	94.1%	240
No Adaptive Difficulty Adjustment	94.2%	8.0	50.5	93.5%	230
LoRA Only	94.0%	2.1	11.8	93.7%	240
Prefix Tuning Only	91.5%	1.6	9.7	91.0%	270
Prompt Tuning Only	92.0%	2.5	12.0	91.5%	260

The table 8 of ablation study compares the performance of ACL-PEFT-LLM using different configurations and it is possible to note the trade-off between the accuracy, training time, GPU memory, F1 score and the inference latency. The Full Model (ACL-PEFT-LLM) has the best accuracy (96.8) and F1 score (96.2) and also has a reasonable training time (7.2 hours) and GPU memory consumption (48.2 GB), which denotes a balanced performance-efficiency trade-off. By contrast, No Curriculum Learning leads to minor accuracy (93.5%) and F1 score (93.0%) loss because of the absence of a systematic development of tasks, whereas No PEFT (Full Fine-Tuning) has 94.6% accuracy, but takes much longer (12 hours) and uses more memory on the GPU (75.0 GB), indicating that the method is rather inefficient. The No Adaptive Difficulty Adjustment setting demonstrates a slightly deteriorated performance with accuracy of 94.2 and F1 score of 93.5, indicating that the dynamically changing the difficulty of a target is helpful in fine-tuning. The main benefit of LoRA Only is its efficiency, which requires only 2.1 hours of training and 11.8 GB of GPU memory, but has 94.0% of accuracy and 93.7% of F1 score, demonstrating that with finely-tuning, resource consumption can be significantly lower and still have high performance. The most efficient training time (1.6 hours) and GPU memory (9.7 GB), however, comes with less accurate (91.5) and F1 score (91.0), which shows the trade-off between efficiency and performance. Prompt Tuning Only balances efficiency on training and model performance, but at a slightly lower accuracy and F1 score than LoRA. In general, ACL-PEFT-LLM is the most suitable solution because of its high accuracy, F1 score, and computational efficiency, and LoRA is the most efficient in terms of computational power.

## 5 Conclusion

The ACL-PEFT-LLM model is an effective model that combines Adaptive Curriculum Learning (ACL) and Parameter-Efficient Fine-Tuning (PEFT) approaches to address the limitations of computational inefficiency in domain-specific fine-tuning of Large Language Models (LLMs). This mixed method allows the model to evolve over time to complicated tasks by using a minimum number of resources. Using both ACL, which raises and lowers task difficulty according to the changing model performance, and PEFT techniques such as LoRA, Prefix Tuning, and Prompt Tuning, the model drastically lowers the amount of GPU memory and training duration and yet still performs well. ACL-PEFT-LLM is more efficient and can reach higher accuracy and F1 score (96.8 and 96.2 respectively) than other models such

as LoRA, Prefix Tuning, or Adapters though are more efficient. The model also provides an Accuracy-Efficiency Ratio (AER) of 5.12, the value that reflects the tradeoff between the performance and the computational efficiency of the model. ACL-PEFT-LLM is more scalable and efficient than Full Fine-Tuning (Full FT) which is resource-intensive due to the fact that Full FT requires fine-tuning of domain specific models. The main strengths of ACL-PEFT-LLM are its high performance, its ability to resourcefully utilize resources, versatility to domain-specific tasks, and ability to scale to resource-constrained settings. The next step in the evolution of this method might be the optimization of more LLM architectures, the development of new PEFT approaches, and the introduction of multi-task learning and continuous learning methods that will allow for the optimization of the fine-tuning process and make the model more adaptable to a greater variety of real-world tasks.

## References

- [1] Aazami, M., & Fallah, M. (2016). The effect of the hidden curriculum on learning English lesson from the English teachers point of view in amol high schools. *International Academic Journal of Social Sciences*, 3(1), 24–33.
- [2] Balaji, R., Logesh, V., Thinakaran, P., & Menaka, S. R. (2022). E-Learning Platform. *International Academic Journal of Innovative Research*, 9(2), 11–17. <https://doi.org/10.9756/IAJIR/V9I2/IAJIR0911>
- [3] Banitalebi-Dehkordi, A., Amirkhani, A., & Mohammadinasab, A. (2023). EBCDet: Energy-based curriculum for robust domain adaptive object detection. *IEEE Access*, 11, 77810–77825. <https://doi.org/10.1109/ACCESS.2023.3298369>
- [4] Battou, A., El Mezouary, A., Cherkaoui, C., & Mammass, D. (2011). An adaptive learning system architecture based on a granular learning object framework. *International Journal of Computer Applications*, 32(5), 18–26.
- [5] Challa V., & Bright, A. O. (2025). Low-Resource Fine-Tuning of LLMs for Domain-Specific Tasks. *Universal Research Reports*, 12(4), 45–56. <https://doi.org/10.36676/urr.v12.i4.1621>
- [6] Dong, H. (2021). Adapting during the pandemic: A case study of using the rapid prototyping instructional system design model to create online instructional content. *The Journal of Academic Librarianship*, 47(3), 102356. <https://doi.org/10.1016/j.acalib.2021.102356>
- [7] Dun, J., Wang, J., Li, J., Yang, Q., Hang, W., Lu, X., ... & Shi, J. (2024). A trustworthy curriculum learning guided multi-target domain adaptation network for autism spectrum disorder classification. *IEEE Journal of Biomedical and Health Informatics*, 29(1), 310–323. <https://doi.org/10.1109/JBHI.2024.3476076>
- [8] El-Demerdash, A. A., & Abdel-Hamid, N. B. (2025). Metaheuristic-Driven Hyperparameter Optimization for Bert in Sentiment Analysis. *Archives for Technical Sciences*, 2(33), 176–192. <https://doi.org/10.70102/afts.2025.1833.176>
- [9] Fitria, T. N. (2024). Using Canva in creating and designing e-Module for english language teaching. *SAGA: Journal of English Language Teaching and Applied Linguistics*, 5(2), 111–122. <https://doi.org/10.21460/saga.2024.52.194>
- [10] Goez, H., Lai, H., Rodger, J., Brett-MacLean, P., & Hillier, T. (2020). The DISCuSS model: Creating connections between community and curriculum—A new lens for curricular development in support of social accountability. *Medical Teacher*, 42(9), 1058–1064. <https://doi.org/10.1080/0142159X.2020.1779919>
- [11] Horn, J. G., & Van Niekerk, L. (2020). The patchwork text as assessment tool for postgraduate law teaching in South Africa. *Obiter*, 41(2), 292–308. <http://dx.doi.org/10.17159/obiter.v41i2.9152>
- [12] Hung, L. C., & Pan, H. J. (2025). Innovative approach to ESD integration into school-based curriculum development modules for elementary schools. *Sustainability*, 17(4), 1427. <https://doi.org/10.3390/su17041427>

- [13] Kavitha, K. P., & Kirubanand, V. B. (2025). Performance evaluation of transfer learning VGG16 in handwritten text using word beam search and language model. *Journal of Internet Services and Information Security*, 15(2), 536–551. <https://doi.org/10.58346/JISIS.2025.12.037>
- [14] Kawamae, N. (2025, August). Knowledge-aligned domain shift tuning for efficient adaptation in large language models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2* (pp. 1128-1138). <https://doi.org/10.1145/3711896.3737013>
- [15] Kong, Y., Nie, Y., Dong, X., Mulvey, J. M., Poor, H. V., Wen, Q., & Zohren, S. (2024). Large language models for financial and investment management: Applications and benchmarks. *Journal of Portfolio Management*, 51(2), 162–210. <https://doi.org/10.3905/jpm.2024.1.645>
- [16] Lee, J., Stevens, N., & Han, S. C. (2025). Large language models in finance (finllms). *Neural Computing and Applications*, 37(30), 24853-24867. <https://doi.org/10.1007/s00521-024-10495-6>
- [17] Mariani, A., Pellegrini, E., & De Momi, E. (2020). Skill-oriented and performance-driven adaptive curricula for training in robot-assisted surgery using simulators: a feasibility study. *IEEE transactions on biomedical engineering*, 68(2), 685-694. <https://doi.org/10.1109/TBME.2020.3011867>
- [18] Noviantari, I., & Agustina, D. A. (2023). Development of teaching modules on independent curriculum implementation. In *Social, Humanities, and Educational Studies (SHES): Conference Series* (Vol. 6, No. 1, pp. 465-470). <https://doi.org/10.20961/shes.v6i1.71154>
- [19] Poncette, A. S., Glauert, D. L., Mosch, L., Braune, K., Balzer, F., & Back, D. A. (2020). Undergraduate medical competencies in digital health and curricular module development: mixed methods study. *Journal of medical Internet research*, 22(10), e22161. <https://doi.org/10.2196/22161>
- [20] Prottasha, N. J., Mahmud, A., Sobuj, M. S. I., Bhat, P., Kowsher, M., Yousefi, N., & Garibay, O. O. (2024). Parameter-efficient fine-tuning of large language models using semantic knowledge tuning. *Scientific Reports*, 14(1), 30667. <https://doi.org/10.1038/s41598-024-75599-4>
- [21] Sappa, A. (2025). Assessing the Impact of Large Language Models on the Scalability and Efficiency of Automated Feedback Mechanisms in Massive Open Online Courses. *Indian Journal of Information Sources and Services*, 15(2), 275–286. <https://doi.org/10.51983/ijiss-2025.IJISS.15.2.35>
- [22] Shreshthi, A. J., Spanò, S., Di Nunzio, L., La Cesa, R., Re, C. V. M., & Cardarilli, G. C. (2025). Parameter-Efficient Fine-Tuning of LLMS for Low-Resource Deployment in Real-World Applications. *International Journal of Intelligent Systems and Applications in Engineering*, 13(1), 566–572.
- [23] Verma, A., & Kulkarnin, R. (2025). Adaptive Learning Algorithms for Differentiated Instruction in Mathematics Education. *International Academic Journal of Science and Engineering*, 12(1), 50–53. <https://doi.org/10.71086/IAJSE/V12I1/IAJSE1209>
- [24] Wang, A., Islam, M., Xu, M., & Ren, H. (2023). Curriculum-based augmented fourier domain adaptation for robust medical image segmentation. *IEEE Transactions on Automation Science and Engineering*, 21(3), 4340-4352. <https://doi.org/10.1109/TASE.2023.3295600>
- [25] Wang, L., Chen, S., Jiang, L., Pan, S., Cai, R., Yang, S., & Yang, F. (2025). Parameter-efficient fine-tuning in large language models: a survey of methodologies. *Artificial Intelligence Review*, 58(8), 1-64. <https://doi.org/10.1007/s10462-025-11236-4>
- [26] Xie, Z., Ling, H. Y., Kim, N. H., & Van De Panne, M. (2020, December). Allsteps: curriculum-driven learning of stepping stone skills. In *Computer Graphics Forum* (Vol. 39, No. 8, pp. 213-224). <https://doi.org/10.1111/cgf.14115>
- [27] Yang, Y., Zhou, J., Wong, N., & Zhang, Z. (2024, June). Loretta: Low-rank economic tensor-train adaptation for ultra-low-parameter fine-tuning of large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)* (pp. 3161-3176). <https://doi.org/10.18653/v1/2024.naacl-long.174>

- [28] Zhang a, H., & Galaup ab, M. (2023). Reflection on the construction and impact of an adaptive learning ecosystem. *Revue internationale des technologies en pédagogie universitaire*, 20(2), 125-138. <https://doi.org/10.18162/ritpu-2023-v20n2-10>
- [29] Zhang, Q. (2024). Harnessing artificial intelligence for personalized learning pathways: A framework for adaptive education management systems. *Applied and Computational Engineering*, 82(1), 167-172.

## Authors Biography



**G. Srinivasa Raju** received his M. Tech degree in Information Technology from SRKR Engineering College, Bhimavaram, in 2020. He is currently working as an Assistant Professor in the Department of Information Technology at SRKR Engineering College, Andhra Pradesh, India, and has been serving in this role since 2021. He has contributed to research through publications in journals and conferences. His research interests include Computer Science Information Systems, with a focus on Web Technologies and emerging areas in computing.



**Dr.A. Sri Krishna** currently working as a Professor and HoD of Artificial Intelligence Department. He has 11 years of teaching and 4 years of research experience. He received his Ph.D. degree in Computer Science and Engineering from Andhra University, in 2015. He worked as JRF(P) & SRF for the Privacy-Preserving Data Publication project, sponsored by DST, at Andhra University from 2010 to 2014 and developed Tools and algorithms for PDP. His main research area is on Data Privacy and Predictive analytics under Data Engineering. In this area, he has published twelve journals and three conference papers during and after his Ph.D. He is an editorial board member in the journal of information, and he acts as a reviewer for Springer international conferences. Under his supervision, one student was awarded a Ph.D. degree in Data Privacy in Big Data fields. He also completed PG Diploma in Data Science from IIIT Bangalore. He is currently working on building Machine Intelligence models to monitor the aqua culture using IoT and Machine Learning



**Dr. Malijeddi Murali** received his Ph.D. from Andhra University in Electronics and Communication Engineering. He has over 25 years of academic and administrative experience and currently serves as Professor, with prior roles including Principal, Vice-Principal, and Dean (R&D and Skill Development). His research interests include VLSI design, antenna systems, wireless communications, IoT, and machine learning. He has published extensively in reputed journals and conferences and has guided six Ph.D. scholars. He is the Chief Investigator of the AICTE Chip-to-Startup (C2S) initiative and Chief Mentor of the AICTE IDEA Lab. He has received multiple awards for academic excellence and leadership.