# Result Analysis of Low-Rank Adaptation Hyperparameters for PolyCC LLM League on AWS SageMaker

Muhammad Adib bin M Salim[1*], Noor Hayati Binti Basan[1], Lim Chi Zhen[2]

[1]Department of Information and Communication Technology & Politeknik Mersing Johor, Malaysia
[2]PMCare Sdn. Bhd., Malaysia

adibsalim@tvet.pmj.edu.my; noorhayati@tvet.pmj.edu.my; chizhen2001@gmail.com

*Abstract*— **This experiment investigates the optimization of Parameter-Efficient Fine-Tuning (PEFT) for foundations models in a cloud-native environment. Specifically, it analyses the performance of Meta's Llama 3.2 3B Instruct model when adapted to a specialized dataset focused on the Malaysian vocational education landscape, industry partnerships, and institutional roles of polytechnics and community colleges. Using Low-Rank Adaptation (LoRA) on Amazon SageMaker, fifteen experimental iterations were conducted to identify the critical thresholds for learning rates, epoch counts, and rank configurations. The findings indicate that Llama 3.2 3B is highly sensitive to the scaling ratio between LoRA Alpha and Rank. The study identifies that Lim-Model-13, utilizing a 10-4 learning rate and a 4:1 Alpha-to-Rank ratio over 12 epochs, provides the most balanced performance by minimizing evaluation loss (1.0017) while preventing the catastrophic forgetting observed at higher learning rates. A detailed gap analysis further reveals critical boundaries between training fit and generalization, highlighting the risks of aggressive learning in specialized domains.**

*Keywords*— **Llama 3.2, LoRA, AWS SageMaker, Fine-Tuning, PolyCC LLM League**

## I. INTRODUCTION

The PolyCC Large Language Model (LLM) League represents a strategic initiative by the Malaysia Ministry of Higher Education (MOHE) and Jabatan Pendidikan Politeknik & Kolej Komuniti (PolyCC) to train students and lecturers from polytechnics and community colleges to build their own AI tools. In this gamified setting, participants from polytechnics and community colleges are tasked with building fit-for-purpose generative AI solutions through Supervised Fine-Tuning (SFT) and prompt engineering. The core objective is to adapt foundational models to address specific strategies of the PolyCC Artificial Intelligence Reform (PAIR) Agenda.

To address these challenges, a prominent method known as PEFT [1] has emerged as a promising solution. By reducing the number of trainable parameters, PEFT significantly lowers the computational cost of adapting PLMs to specific tasks while maintaining performance comparable to full fine-tuning [2, 3, 4, 5].

A primary challenge in this domain is the efficient adaptation of foundational models like Meta's Llama 3.2 3B Instruct. As we pre-train larger models, full fine-tuning, which retrains all model parameters, becomes less feasible [6]. While this model is designed for edge and mobile environments, its 3.21 billion parameters still require substantial computational resources for full fine-tuning. This experiment explores the technical execution of Low-Rank Adaptation (LoRA) on the Amazon SageMaker platform to customize this model for the Malaysian vocational sector. By leveraging SageMaker JumpStart, the study benchmarks fifteen training iterations (Lim-Model-1 through Lim-Model-15) to identify hyperparameter configurations that yield optimal convergence for educational instruction-following tasks.

## II. THEORITICAL FRAMEWORK

AI models like Llama 3.2 are "generalists". They know a little bit about everything because they were trained on a massive amount of data from the internet. However, to make them useful for a specific task, like answering questions about Malaysian polytechnics, the experiment is using specialized training called PEFT. Fine-tuning helps the model learn the specific jargon, rules, and context of a new field. Because updating every single part of a large AI is very expensive, the experiment uses Parameter-Efficient methods that only change a tiny portion of the model. Many vocational training institutions operate under limited computational resources, making full fine-tuning of large language models impractical. PEFT methods address this limitation by significantly reducing memory and computational requirements without sacrificing performance [7, 8]. This is particularly beneficial in vocational training environments, where institutions often face resource constraints and require frequent updates to reflect evolving industry standards [7]. PEFT emerged as a practical solution for adapting

*Corresponding Author

large language models (LLMs) to domain-specific applications. The approaches aligned with human feedback ensure that LLM outputs remain pedagogically appropriate, which is critical in vocational education where procedural accuracy and safety are essential [9] which such as vocational education.

### *A. Low-Rank Adaptation (LoRA)*

Low-Rank Adaptation (LoRA) introduces trainable low-rank matrices into transformer layers, enabling efficient adaptation without modifying the original model weights [10]. LoRA can reduce the number of trainable parameters by 10,000 times and the GPU memory requirement by 3 times [8]. LoRA performs on-par or better than fine-tuning in model quality on RoBERTa, DeBERTa, GPT-2, and GPT-3, despite having fewer trainable parameters, a higher training throughput, and, unlike adapters, no additional inference latency [11, 12].

Low-Rank Adaptation (LoRA)Think of a large AI model as a massive textbook. Full fine-tuning is like trying to rewrite every page in that book, which is slow and consume physical resources. LoRA is like adding "sticky notes" throughout the book with your own updates. You keep the original book frozen and only change a tiny part (less than 1%) of the settings. Mathematically, the update to the AI's brain is handled by two small pieces called Matrix A and Matrix B. The final output $h$ for an input $x$ is calculated using formula:

$$h = W_0 x + \frac{\alpha}{r} BAx$$

In this formula, $W_0$ represents the original frozen knowledge, $r$ (Rank) is the size of the "sticky note", and $\alpha$ (Alpha) is like a volume knob that controls how much the AI listens to the new information. In other words, LoRA reduces the number of trainable parameters by injecting low-rank decomposition matrices [13].

### *B. The AWS Environment*

Amazon SageMaker JumpStart provides a managed hub for foundation models, enabling practitioners to execute fine-tuning in a network-isolated environment. The use of standard precision (bfloat16) instead of 8-bit quantization ensures numerical stability during the weight update phase for distilled architectures like Llama 3.2. By using the No-Code platform that simplifies the fine-tuning of LLM. This no-code approach leverages pre-trained foundation models in AWS, enabling users with limited coding skills to train their own models. This effectively democratizes fine-tuning across all users, making it more accessible and user-friendly [4].

## III. EXPERIMENT METHODOLOGY

The primary goal of this experiment is to identify the best way to teach a general-purpose AI model to become familiar in the PolyCC context by observing the Loss (error rate) and Perplexity (confusion) value. To do this, the experiment is using Domain Adaptation Fine-tuning which is supported by the platform used. Domain adaptation fine-tuning allows pre-trained foundation models to be customized for specific tasks using limited domain-specific data. This process is ideal for incorporating industry jargon or technical terms into the model [5]. This is by conducting a series of controlled tests using various configurations, known as control hyperparameters, which act as the "tuning knobs" for the model's learning process. The methodology followed a cloud-native workflow on Amazon SageMaker, involving data preparation, model initialization, and the execution of fifteen distinct training runs designated and named as Lim-Model-1 through Lim-Model-15. Each run was evaluated to see how well the AI could understand and respond to specific educational topics while maintaining its overall reasoning ability. This structured approach allowed us to identify the point where the model learns most effectively without merely memorizing the training samples.

### *A. Dataset*

The AI was trained on a special dataset with 1,105 examples of questions and answers about Malaysian education. The content of the dataset focuses on the different roles of polytechnics and community colleges, highlighting how polytechnics focus on hands-on technical degrees while community colleges serve a broader range of students with academic transfer programs. It includes details on how these institutions provide job training to help students acquire the specific skills needed to get ready for the workforce. Finally, the data covers industry links, describing how schools work with big companies like IBM and Siemens to design their classes and provide real-world experience. In general, the domain specifically focuses on:

- **Institutional Comparisons:** Distinguishing between applied learning in polytechnics and general academic pathways in community colleges.
- **Workforce Development:** Analyzing regional economic impact and customized training for local industry clusters.
- **Strategic Partnerships:** The role of collaborations with entities like Siemens and IBM in curriculum design.

### B. Training Settings

The training jobs were executed in the us-east-1 region using the JumpStartEstimator class. All models targeted the query and value projection layers (q_proj, v_proj) to isolate the impact of hyperparameters. The experiment is used different settings for each of the 15 attempts. This is done by mainly changed the Learning Rate (how fast the AI tries to learn) and Epochs (how many times the AI reads the entire dataset). Efficiency was managed through data handling settings, where models with longer training times to maintain high data throughput and minimize idle compute time. Fixed seed of 10 is maintained to ensure that the results were reproducible across different runs. The volume of the update was controlled by keeping a consistent 4:1 ratio between LoRA Alpha and Rank (e.g., 64/16), a strategy known to stabilize learning across specialized domains. The specific training setting for the experiment is specified in Table 1.

TABLE I
TRAINING JOB SPECIFICATIONS

| **Parameter** | **Specifications** |
|---|---|
| Base Model | meta-textgeneration-llama-3-2-3b-instruct |
| Precision | Bfloat6 (Standard) |
| Batch Size (per device) | 1 to 2 |
| Validation Split Ratio | 0.2 (20% data kept for testing) |
| Int8 Quantization | False (High Precision) |
| LoRA Dropout | 0.05 (Prevents memorization) |
| Data Workers | 1 to 20 (Task Dependent) |
| Target Modules | q_proj, v_proj |

## IV. EMPIRICAL RESULTS

The experiment is identified the results by tracked how well each model did by looking at its Loss (error rate) and Perplexity (predictive confidence-how confused the AI is) in each cycle of tuning with lower numbers are better for both Loss and Perplexity. The experimental runs varied and manipulate the Learning Rate (LR) $1 \times 10^{-5}$ to $1 \times 10^{-3}$ and epoch counts from 6 to 12. Performance was measured training loss via evaluation loss (EVAL LOSS) and evaluation perplexity (EVAL PPL).

Table 2 summarizes the impact of different hyperparameter configurations on the performance of LoRA-based PEFT models which highlight the effectiveness of PEFT techniques, particularly LoRA, in optimizing large language models under varying hyperparameter configurations. The findings demonstrate that careful selection of learning rate, training duration and LoRA-specific parameters (rank and alpha) significantly influences model performance.

TABLE II
HYPERPARAMETERS AND MODEL PERFORMANCE SUMMARY

| **MODEL ID** | **EPOCHS** | **LEARNING RATE** | **RANK** | **ALPHA** | **EVAL LOSS** | **EVAL PPL** | **RUN TIME (S)** |
|---|---|---|---|---|---|---|---|
| Lim-Model-1 | 6 | 0.001 | 16 | 64 | 1.0654 | 2.9819 | 2926 |
| Lim-Model-2 | 8 | 0.001 | 16 | 64 | 1.0785 | 2.9168 | 3749 |
| Lim-Model-3 | 10 | 0.001 | 16 | 64 | 1.0757 | 2.9319 | 4561 |
| Lim-Model-7 | 10 | 2.5e-05 | 16 | 64 | 1.0919 | 2.9798 | 4621 |
| Lim-Model-8 | 12 | 0.00025 | 16 | 64 | 1.2746 | 3.5776 | 5401 |
| Lim-Model-9 | 8 | 5e-05 | 16 | 64 | 1.0756 | 2.9513 | 3748 |
| Lim-Model-10 | 10 | 5e-05 | 16 | 64 | 1.0744 | 2.9283 | 4517 |
| Lim-Model-12 | 6 | 0.0001 | 16 | 64 | 1.1501 | 3.1270 | 2921 |
| Lim-Model-13 | 12 | 0.0001 | 16 | 64 | 1.0017 | 2.9456 | 5409 |
| Lim-Model-14 | 6 | 5e-05 | 8 | 32 | 1.2445 | 3.4712 | 2911 |
| Lim-Model-15 | 6 | 1e-05 | 8 | 32 | 1.2357 | 5.4408 | 1952 |

### A. Impact of Learning Rate and Convergence

Learning rate acted as the primary driver of early convergence [13]. The AI learns fast but makes mistakes or forgets its old knowledge. High rates ($0.001$) produced rapid loss reduction but caused the evaluation loss to elevated early (Models 1-3) [1]. Model 8, using a rate of $2.5 \times 10^{-4}$ over 12 epochs, demonstrated clear

overfitting, achieving the lowest training loss of 0.5261 but suffering a significant spike in Evaluation Loss (1.2746) and Perplexity (3.5776).

### *B. Capacity Constraints: Rank and Alpha*

Models 14 and 15 utilized lower LoRA settings (Rank 8, Alpha 32). These models showed the highest confusion levels (PPL), particularly Model 15 which spiked to 5.4408. This indicates that for complex tasks like comparing technical curricula, the model requires the higher learning capacity provided by a Rank of 16.

### *C. Capacity Constraints: Rank and Alpha*

Run time scaled linearly with study time (epochs). The fastest run, Model 15 (1.952$s$), was also the least accurate. The longest run, Model 13 (5.409$s$), provided the highest accuracy, showing that the extra compute time was a necessary investment for a trusted result. Papers in this format must not exceed ten (10) pages in length. Papers for initial consideration may be submitted in either .doc or .pdf format. Final camera-ready versions should consider reviewers' suggested amendments.

## V. Optimal Setup

The success of Lim-Model-13 indicates that a balanced configuration using a moderate learning rate ($1 \times 10^{-4}$) over a longer training duration (12 epochs) is superior for domain adaptation and expected providing the most accurate and trusted result within this experiment. It achieved the lowest evaluation error as of 1.0017 with a highly stable perplexity of 2.9456. This was possible because it read the data 12 times at a moderate learning speed, allowing it to internalize the domain without forgetting its core reasoning skills.

### *A. Analysis of Model Tiers*

i. **Optimal Generalization:** Lim-Model-13 achieved the best (lowest) Eval Loss of 1.0017. This setup utilizes 12 epochs at a moderate learning speed, ensuring the highest predictive accuracy across the experiment domain.

ii. **Highest Confidence:** Lim-Model-2 achieved the lowest Perplexity (2.9168), though its error rate was higher than Model 13. This indicates the model was most certain in its next-word predictions, likely due to the higher learning rate facilitating strong early feature extraction.

iii. **Efficiency Gap:** Lim-Model-15 finished training fastest (1952$s$) but failed to converge, resulting in an Eval PPL of 5.4408. The high perplexity signifies an unreliable model that struggles with domain-specific instruction following.

### *B. Stability and Retention*

Homogenization provides powerful leverage but demands caution, as the defects of the foundation model are inherited by all the adapted models downstream [14]. In homogenous datasets from the chosen domain, higher learning rates often lead to the model "memorizing" noise [11, 14]. The stability of Lim-Model-13 suggests that maintaining a conservative update schedule allows the model to learn new context patterns while preserving the general logic distilled into the model architecture. By maintaining a moderate update intensity, the model avoids the memorization of training noise seen in Model 8, which exhibited high training accuracy but poor validation performance.

### *C. Gap Analysis of Performance Metrics*

A critical component of identifying trusted results is analysing the "Generalization Gap"—the difference between a model's performance on its training data and its performance on unseen evaluation data.

i. ***The Overfitting Gap:*** Lim-Model-8 presents the most significant gap in the study. While it achieved a Train Loss of 0.5261, its Eval Loss was 1.2746, representing a massive gap of 0.7485. This indicates that the combination of a high epoch count (12) and a relatively high learning rate (0.00025) for this dataset size caused the model to "memorize" specific sentences rather than learn the logic of the PolyCC curriculum.

ii. ***The Generalization Gap:*** Lim-Model-13, the top performer, maintains a healthy gap of 0.1761 (Train Loss: 0.8256, Eval Loss: 1.0017). This gap is small enough to demonstrate that the model is learning generalized patterns that apply to new data, rather than just copying patterns from the training set.

iii. ***The Capacity Gap:*** A clear gap in capability was observed between models using Rank 16 and those using Rank 8. Models 14 and 15 showed a "Confusion Gap," where their evaluation perplexity (PPL) was significantly higher (3.4 to 5.4) compared to the Rank 16 cohort (~2.9). This identifies a structural gap where lower ranks lack the "storage space" to hold the complex nuances of the learning domain.

## VI. CONCLUSION

The experimental results from the PolyCC LLM League show that fine-tuning Meta's Llama 3.2 3B Instruct on AWS SageMaker depends heavily on careful hyperparameter tuning. The results show that model performance is sensitive to learning rate and training duration. Among all configurations, Lim-Model-13 achieves the best performance with the lowest evaluation loss and perplexity, indicating superior generalization ability. In contrast, models trained with very low learning rates or higher epoch counts do not necessarily yield better results, suggesting diminishing returns with extended training.

Additionally, the experiment gap analysis indicates that a learning rate of $10^{-4}$ combined with a LoRA rank of 16 produces the most reliable performance, as it minimizes the difference between training and evaluation results. With these settings, the model converges steadily over 12 epochs, achieving low evaluation error while maintaining strong generalization within the target domain.

For future participation in the league, it would be beneficial to explore automated hyperparameter sweeps within these stable ranges to further improve response quality. In addition, future work should prioritize the careful selection and curation of training examples to enhance the model's consistency and reliability. Overall, the findings suggest that dataset quality plays a more critical role than dataset size [15, 16]. Model performance scales predictably with compute, data, and parameters.


## ACKNOWLEDGEMENT

The authors thank MOHE, JPPKK, and the AWS team for providing the gamified framework of the PolyCC LLM League.


## DECLARATION OF GENERATIVE AI USAGE

During the preparation of this work, the authors used ChatGPT to construct writing ideas and Llama 4 Scout in order to verify data consistency and structure mathematical notation. The authors declare that they reviewed and edited the final output as needed and take full responsibility for the content of the published article.

## REFERENCES


[1] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," in Proc. Int. Conf. Mach. Learn. (ICML), 2019, pp. 2790–2799.

[2] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in Proc. Annu. Meeting Assoc. Comput. Linguistics (ACL) and Int. Joint Conf. Natural Lang. Process. (IJCNLP), 2021, pp. 4582–4597.

[3] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in Proc. Int. Conf. Learn. Representations (ICLR), 2022.

[4] L. Xu, H. Xie, S.-Z. J. Qin, X. Tao, and F. L. Wang, "Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment," IEEE Trans. Pattern Anal. Mach. Intell., 2026, doi: 10.1109/TPAMI.2026.3657354.

[5] K. Randhi, A. K. Sannasi, and A. Tarcar, "Leveraging AWS services for efficient LLM fine-tuning," Persistent Systems Blog, Dec. 24, 2024. [Online]. Available: https://www.persistent.com/blogs/leveraging-aws-services-for-efficient-llm-fine-tuning/

[6] E. J. Hu et al., "LoRA: Low-Rank Adaptation of Large Language Models," arXiv preprint arXiv:2106.09685, 2021.

[7] A. Aghajanyan, S. Gupta, and L. Zettlemoyer, "Intrinsic dimensionality explains the effectiveness of language model fine-tuning," arXiv preprint arXiv:2012.13255, 2020.

[8] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for NLP," arXiv preprint arXiv:1902.00751, 2019.

[9] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al., "Training language models to follow instructions with human feedback," arXiv preprint arXiv:2203.02155, 2022.

[10] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," arXiv preprint arXiv:2106.09685, 2021.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," arXiv preprint arXiv:1706.03762, 2017.

[12] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," arXiv preprint arXiv:2106.09685, 2021.

[13] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," arXiv preprint arXiv:2001.08361, 2020.

[14] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji, A. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. Krass, R. Krishna, R. Kuditipudi, A. Kumar, F. Ladhak, M. Lee, T. Lee, J. Leskovec, I. Levent, X. L. Li, X. Li, T. Ma, A. Malik, C. D. Manning, S. Mirchandani, E. Mitchell, Z. Munyikwa, S. Nair, A. Narayan, D. Narayanan, B. Newman, A. Nie, J. C. Niebles, H. Nilforoshan, J. Nyarko, G. Ogut, L. Orr, I. Papadimitriou, J. S. Park, C. Piech, E. Portelance, C. Potts, A. Raghunathan, R. Reich, H. Ren, F. Rong, Y. Roohani, C. Ruiz, J. Ryan, C. Ré, D. Sadigh, S. Sagawa, K. Santhanam, A. Shih, K. Srinivasan, A. Tamkin, R. Taori, A. W. Thomas, F. Tramèr, R. E. Wang, W. Wang, B. Wu, J. Wu, Y. Wu, S. M. Xie, M. Yasunaga, J. You, M. Zaharia, M. Zhang, T. Zhang, X. Zhang, Y. Zhang, L. Zheng, K. Zhou, and P. Liang, "On the opportunities and risks of foundation models," arXiv preprint arXiv:2108.07258, 2021.

[15] C. Bhavaraju, "Fine-tuning Llama 3.2 3B for AML: Lessons from AWS AI League," Medium, 2025.

[16] M. Haque, "Early Detection and Reduction of Memorization for Domain Adaptation and Instruction Tuning," TACL, vol. 13, pp. 1–15,2025.