

# Parameter-Efficient LLM Refinement for Text-based Sentiment Analysis Tasks (X-PERTS)

Jack Henderson Priya Tamilselvan Sakya Kanuparthry Alagappan Swaminathan  
Georgia Institute of Technology

{jhenderson88, ptamilselvan3, skanuparthry6, aswaminathan44}@gatech.edu

## Abstract

*This project investigates two parameter-efficient fine-tuning (PEFT) methods—BitFit and LoRA—for large language models (LLMs), particularly focusing on small-scale models and ultra-low-resource settings. We extended the framework proposed by Acun-Uyan and Elhoushi in their “Efficient LLM Supervised Fine-Tuning” paper, applying BitFit (which only updates the bias terms) and LoRA (which introduces low-rank weight updates for improved memory efficiency). Our comparison evaluates the performance of these methods using Meta AI’s OPT-125M, OpenAI’s GPT-2, and Google’s lightweight ALBERT on two popular sentiment analysis datasets: Sentiment140 and IMDB50K. The models were evaluated based on accuracy, execution time, and memory usage, with the goal of demonstrating the practicality of PEFT techniques in constrained computing environments such as those faced by independent researchers and graduate students.*

## 1. Introduction

This project investigates two parameter-efficient fine-tuning methods—BitFit and LoRA—focusing on lightweight models and low-resource fine-tuning. The task involves applying sentiment classification fine-tuning to three widely used pre-trained language models: Meta AI’s Open Pre-trained Transformer (OPT-125M) [1], OpenAI’s GPT-2 [2], and the lightweight variant of BERT, ALBERT, developed by Google Research and Toyota Technological Institute at Chicago (TTIC) [3].

We explore two model fine-tuning approaches: BitFit [4] and LoRA [5], both of which were proposed by Bilge Acun-Uyan and Mostafa Elhoushi in their “Efficient LLM Supervised Fine-Tuning” paper [6]. These approaches are designed to significantly reduce computational overhead. BitFit is a sparse fine-tuning technique that only updates the bias terms of the model, thus requiring far less computational power. In contrast, LoRA applies low-rank ap-

proximations to the weight update matrix, reducing memory requirements while still capturing essential model updates. These techniques are computationally more efficient compared to traditional fine-tuning methods, such as Vanilla Fine-Tuning [7]—which fine-tunes all model parameters—and Pattern-based Fine-Tuning (PBFT) [8], which adapts models using task-specific input-output pairs.

This research tackles an important challenge in the field of natural language processing (NLP): the high computational cost of fine-tuning large language models (LLMs). By leveraging parameter-efficient fine-tuning methods like BitFit and LoRA, we aim to demonstrate that graduate students, independent researchers, and practitioners can feasibly fine-tune state-of-the-art models even with limited computational resources. In particular, we apply these fine-tuning techniques to the Sentiment140 dataset, a widely used binary text classification dataset containing 1.6 million labeled tweets (positive and negative sentiments) [9], as well as the IMDB50K movie reviews dataset for binary sentiment classification tasks.

The primary goal of this work is to compare the performance of BitFit and LoRA in terms of classification accuracy, training time, and memory usage across different models. Both Sentiment140 and IMDB50K provide rich, real-world datasets for sentiment analysis, allowing us to evaluate how the models perform on a wide range of text-based inputs. For Sentiment140, we focus on a binary classification task where the models are tasked with predicting sentiment labels from tweets. Similarly, for IMDB50K, we apply the same classification approach to movie reviews, aiming to evaluate the models on a different but related sentiment classification task.

To ensure that training remains computationally feasible, we perform stratified sampling on both datasets, selecting subsets of 50K–100K instances from the original datasets. This strategy not only helps reduce the computational burden but also maintains class balance across the sampled data. Furthermore, to evaluate the trade-offs in performance based on model scale, we compare the fine-tuning results across three different models: the relatively smaller

OPT-125M, the medium-sized GPT-2, and the lightweight ALBERT. This comparison will provide insights into how model size influences both performance and resource demands during fine-tuning.

Ultimately, the project aims to provide a comprehensive comparison between BitFit, LoRA, and Prompt Tuning (which we explore as a third approach) by evaluating each method’s performance on Sentiment140 and IMDB50K datasets. This analysis will illustrate the practical applicability of PEFT methods for fine-tuning large language models in resource-constrained environments, without sacrificing performance.

## 2. Related Works

The effectiveness of PEFT methods, including BitFit and LoRA, has been explored in previous work, such as the study by Acun-Uyan and Elhoushi, which demonstrated that sparse updates like BitFit are highly memory-efficient, though they may diminish model performance on larger datasets. Conversely, LoRA captures low-rank approximations of weight updates, offering a balance between performance and computational efficiency, though it can still lead to higher memory usage. In the context of text classification tasks, several studies, including one by Rust et al., systematically evaluate these techniques in resource-limited scenarios.

Furthermore, the development of Prompt Tuning has emerged as a promising alternative for task-specific adaptation without directly modifying the model’s weights. While effective for low-resource settings, Prompt Tuning still faces challenges in scalability and efficiency for large models, which is where techniques like BitFit and LoRA shine by minimizing the computational footprint.

## 3. Method / Approach

We employ two primary fine-tuning techniques for our experiments: BitFit and LoRA.

**BitFit:** This method involves fine-tuning only the bias terms of a model while keeping all other parameters frozen. This significantly reduces the computational cost and memory usage compared to traditional full fine-tuning.

**LoRA:** This technique introduces low-rank weight matrices to represent parameter updates, which reduces the computational load by constraining the weight updates to a low-rank approximation.

In this study, we applied BitFit and LoRA to three different pre-trained models:

- OPT-125M: A lightweight model by Meta AI.

- GPT-2: OpenAI’s autoregressive transformer model.
- ALBERT: A scaled-down version of BERT with fewer parameters and shared weights.

The models are evaluated on two datasets—Sentiment140 (a dataset for binary sentiment analysis of tweets) and IMDB50K (a dataset for sentiment analysis of movie reviews). We performed fine-tuning on the models using BitFit (which updates only the bias terms) and LoRA (which incorporates low-rank matrix approximations into the weight updates). Each model was evaluated for accuracy, training time, and memory usage.

### 3.1. Baseline

To contextualize the benefits of fine-tuning, baseline metrics (accuracy, training time, and memory usage) were recorded without fine-tuning (i.e., zero-shot inference). This is permissible because the models are pre-trained. The baseline results will help us determine any improvement in performance upon applying fine-tuning for the classification task.

### 3.2. Evaluation Metrics

We evaluate the models using the following metrics:

- Accuracy: The percentage of correctly classified instances.
- Precision: The proportion of true positives among all predicted positives.
- Recall: The proportion of true positives among all actual positives.
- F1-Score: The harmonic mean of precision and recall.
- Training Time and Memory Usage: We monitor the execution time and memory consumption during training to assess the computational efficiency of each method.

## 4. Data

For the experiments, we used the Sentiment140 and IMDB50K datasets:

**Sentiment140:** This dataset consists of 1.6 million tweets, each labeled as either positive or negative sentiment. Given resource constraints, we downsampled the dataset to 50K–100K samples and applied stratified sampling to ensure a balanced class distribution.

**IMDB50K:** A movie review dataset containing 50,000 labeled reviews, also split into training and test sets. We used this dataset to compare model performance across different types of text-based sentiment analysis tasks.

Both datasets were pre-processed to remove noise such as emojis, special characters, and URLs before being tokenized for input into the models. For the Sentiment140 dataset, we applied a similar pre-processing pipeline to clean and structure the data for model training. A sample for both datasets is shown in Figure 1.

Sentiment140	text	user	sentiment
finally set up wireless internet huzzah for tw...	alexwilliams	1	
cleaning the house and packing for my journey ...	LoveandYoga	1	
i broke our site	DjDATZ	0	
3 day long weekend queens bday weekend	toughamber	1	
wrong place at the wrong time always sigh	Nadiahazman	0	
IMDb50k	text		sentiment
This movie is eye-meltingly bad and, sadly, not even unintentionally hilarious. It's just bad.		0	
First a quick 'shut up!' to those saying this movie stinks.		1	
My girlfriend made me watch it. There is nothing positive to say about this film.		0	
This is another Alien imitation and not a very good one at that.		0	
I highly recommend this film...and to watch Stanwyck who is great and beautiful as Lilly.		1	

Figure 1: Data Samples

## 5. Experiments and Results

We conducted a series of experiments to assess the performance of **BitFit** and **LoRA**, two parameter-efficient fine-tuning techniques, alongside **Prompt Tuning**, on three pre-trained large language models: OPT-125M, ALBERT, and GPT-2. These experiments were carried out on two sentiment-classification benchmarks—**Sentiment140** and **IMDb 50 K**. For every model–dataset pair we measured *zero-shot* (pre-finetune) performance and then applied each PEFT strategy, evaluating the resulting models with accuracy, precision, recall, and F1-score (Test-set scores are collected in Table 2; validation metrics and resource usage in Table 1).

All runs share a common data pipeline. Sentiment140 is down-sampled to 50 k tweets by stratified sampling to preserve class balance; IMDb 50 K uses its full 25 k/25 k train/test split. Tweets are lower-cased, URLs/emojis stripped, and both corpora are tokenised with their respective model vocabularies (max-len 128 for tweets, 256 for reviews). Datasets are then converted to PyTorch Dataset objects with `input_ids`, `attention_mask`, and binary labels.

Fine-tuning is performed on an M2 MacBook Air (with 8-core CPU, 10-core GPU, 8GB RAM, and no CUDA support), to demonstrate feasibility under strict hardware constraints. We use AdamW with learning-rate  $2 \times 10^{-5}$ , batch size 16, and train for up to six epochs. BitFit updates only bias parameters ( $< 0.1\%$  of weights); LoRA injects rank-4 adapters with  $\alpha = 16$ ; Prompt Tuning prepends a 20-token soft prompt. During training we log wall-clock time, peak memory, and per-epoch accuracy/loss. Post-training we compute full test-set confusion matrices (Fig. 2) and learning curves (Fig. 3), which underpin the analysis below.

## 5.1. Sentiment140

### 5.1.1 Pre-finetuned Performance

The first three confusion-matrix panels in Fig. 2—(a) ALBERT, (b) GPT-2, and (c) OPT-125M—highlight how poorly the off-the-shelf models transfer to Twitter data. ALBERT (Fig. 2-a) mis-labels nearly half of the positive tweets as negative (precision<sub>pos</sub> = 0.52, recall<sub>pos</sub> = 0.45), while GPT-2 (Fig. 2-b) collapses into a degenerate solution in which *every* tweet is predicted *positive*, yielding 50 % accuracy but zero practical value. OPT-125M fares slightly better (Fig. 2-c), yet the 1318 false-positives and 1555 false-negatives reveal a clear class-imbalance bias. These numbers are mirrored by the test-set metrics in Table 2 (rows “Baseline”, not shown for brevity).

### 5.1.2 Post-finetuned Performance

**BitFit.** With only bias terms updated, BitFit dramatically re-balances the error distribution: ALBERT’s false-negatives drop from 2742 to 979 and false-positives from 2114 to 1026; GPT-2 jumps from unusable to **0.8007** accuracy, and OPT-125M reaches **0.8282** (Table 2). The learning-rate curves in Fig. 3-a-c show that all BitFit configurations plateau by epoch 3, making it the fastest route to a “good-enough” Twitter classifier.

**LoRA.** Injecting rank-4 adapters halves ALBERT’s residual errors (902 FP, 1440 FN) and boosts accuracy to **0.7658**; GPT-2 climbs to the dataset-best **0.8111** and OPT-125M to **0.7883**. Unlike BitFit, LoRA continues to improve until epoch 6 (Fig. 3-b&c), reflecting the extra capacity afforded by the low-rank matrices.

**Prompt Tuning.** A 20-token soft prompt converges in *half* the wall-clock time of LoRA (Table 1), but inference latency explodes (640–627 s in Table 2). Accuracy peaks 2–3 pp lower, yet early stopping after one epoch still yields a 20 pp gain over baseline, making prompt tuning attractive for rapid iteration.

## 5.2. IMDb 50 K

### 5.2.1 Pre-finetuned Performance

Panels (d)–(f) of Fig. 2 reveal a stronger zero-shot baseline on long-form movie reviews: ALBERT reaches 0.59 accuracy, GPT-2 0.62, and OPT-125M 0.63. Nevertheless, all three exhibit a pronounced *negative-class bias*: 96–97 % of positives are mistakenly labelled negative.

### 5.2.2 Post-finetuned Performance

**BitFit.** Updating fewer than 0.1 % of parameters suffices to correct most of the bias: ALBERT attains 0.8648 accu-

racy, GPT-2 0.8803, and OPT-125M **0.8862** (Table 2). Accuracy curves in Fig. 3-d-f flatten after epoch 3, mirroring the pattern observed on Sentiment140.

**LoRA.** Rank-4 adapters deliver the best trade-off on the larger GPT-2 backbone, peaking at **0.8883** accuracy, while remaining competitive for ALBERT (0.8324) and OPT-125M (0.8554). Loss curves drop monotonically across all batch sizes, indicating stable optimisation.

**Prompt Tuning.** Although it converges fastest, prompt tuning levels off roughly two accuracy points below LoRA on every model. For hyper-parameter sweeps or few-epoch prototypes, it remains attractive; for final deployment, BitFit or LoRA is preferable.

### 5.3. Cross-Dataset and Cross-Model Comparison

- **Error symmetry.** Across both datasets the confusion-matrix diagonals widen progressively from Baseline → BitFit → LoRA (Fig. 2), showing that PEFT methods reduce *both* error types instead of shifting bias.
- **Learning dynamics.** BitFit saturates by epoch 3, LoRA keeps improving to epoch 6, and Prompt Tuning hits diminishing returns after the very first epoch (Fig. 3).
- **Text length matters.** IMDb reviews are easier than tweets; every method achieves its highest absolute scores on IMDb, and the learning curves are smoother, reflecting richer sentiment cues in longer texts (Tables 1–2).
- **Resource trade-offs.** Prompt Tuning delivers the steepest loss drop per epoch but suffers inference latencies 15–600 × larger than BitFit/LoRA; BitFit is the most RAM-efficient (as low as 1.94 GB); LoRA reaches the best final accuracy on the deepest backbone (GPT-2 IMDb, Table 2) at the cost of ≈20 % extra memory (Table 1).

## 6. Conclusion

We set out to determine whether *parameter-efficient* fine-tuning techniques can deliver competitive sentiment-classification performance on consumer-grade hardware.

The experiments conclude with the following conclusions:

1. **BitFit is surprisingly competitive.** It delivered the top test accuracy on *four* of the six model–dataset pairs—ALBERT/Sentiment140 (0.7995), ALBERT/IMDb50 K (0.8648), OPT/Sentiment140 (0.8282) and OPT/IMDb50 K (0.8862)—despite updating only bias parameters.

Validation-set			F1	Acc	Run[s]	Inf[s]	Mem[GB]
ALBERT	S140	BitFit	0.7995	0.7995	3735	57	1.94
ALBERT	S140	LoRA	0.7651	0.7658	4097	54	1.94
ALBERT	S140	Prompt	0.6937	0.6941	1951	1951	1.94
ALBERT	IMDb	BitFit	0.8647	0.8648	4736	87	2.27
ALBERT	IMDb	LoRA	0.8324	0.8324	6141	86	2.27
ALBERT	IMDb	Prompt	0.7358	0.7358	2081	2081	2.27
GPT-2	S140	BitFit	0.8003	0.8007	4434	77	5.90
GPT-2	S140	LoRA	0.8110	0.8111	6469	78	5.90
GPT-2	S140	Prompt	0.7541	0.7541	1911	1911	5.90
GPT-2	IMDb	BitFit	0.8802	0.8803	3679	57	7.41
GPT-2	IMDb	LoRA	0.8882	0.8883	4653	54	7.41
GPT-2	IMDb	Prompt	0.7646	0.7646	1966	1966	7.41
OPT125M	S140	BitFit	0.8282	0.8282	3571	48	4.30
OPT125M	S140	LoRA	0.7877	0.7883	4773	45	4.30
OPT125M	S140	Prompt	0.7931	0.7931	1797	1796	4.30
OPT125M	IMDb	BitFit	0.8862	0.8862	3599	49	3.83
OPT125M	IMDb	LoRA	0.8554	0.8554	4785	14310	3.83
OPT125M	IMDb	Prompt	0.7854	0.7854	1817	1817	3.83

Table 1: Full validation-set results for all model–dataset–method combinations.

Final test-set			Acc	F1-M	F1-W	Inf[s]
ALBERT	S140	BitFit	0.7995	0.7995	0.7995	9.8
ALBERT	S140	LoRA	0.7658	0.7651	0.7651	0.8
ALBERT	S140	Prompt	0.6941	0.6937	0.6937	640.5
ALBERT	IMDb	BitFit	0.8648	0.8647	0.8647	21.9
ALBERT	IMDb	LoRA	0.8324	0.8323	0.8324	21.2
ALBERT	IMDb	Prompt	0.7358	0.7358	0.7358	668.6
GPT-2	S140	BitFit	0.8007	0.8003	0.8003	19.6
GPT-2	S140	LoRA	0.8110	0.8110	0.8110	19.0
GPT-2	S140	Prompt	0.7541	0.7541	0.7541	627.0
GPT-2	IMDb	BitFit	0.8802	0.8802	0.8802	10.1
GPT-2	IMDb	LoRA	0.8882	0.8882	0.8882	9.8
GPT-2	IMDb	Prompt	0.7646	0.7646	0.7646	347.4
OPT125M	S140	BitFit	0.8282	0.8282	0.8282	8.6
OPT125M	S140	LoRA	0.7883	0.7877	0.7877	7.9
OPT125M	S140	Prompt	0.7931	0.7931	0.7931	310.0
OPT125M	IMDb	BitFit	0.8862	0.8861	0.8862	15.5
OPT125M	IMDb	LoRA	0.8554	0.8554	0.8554	5202
OPT125M	IMDb	Prompt	0.7854	0.7854	0.7854	585

Table 2: Comprehensive test-set results (F1-M = macro-F1, F1-W = weighted-F1).

2. **LoRA shines on larger backbones.** For GPT-2, LoRA edged out BitFit on both datasets (0.8111 vs. 0.8007 on Sentiment140 and **0.8883** vs. 0.8803 on IMDb), confirming that low-rank adapters better exploit deeper representations.
3. **Prompt Tuning is fastest to train but slowest at inference.** Training runtime dropped by **45–60 %** compared with BitFit (e.g. ALBERT/IMDb 2081 s vs. 4736 s), yet inference time ballooned to **309–669 s** compared with ≤22 s for the other PEFT methods, making prompts ill-suited for production scoring.
4. **Laptop-level hardware suffices.** Peak memory never exceeded **7.41 GB** (GPT-2) and was as low as **1.94**

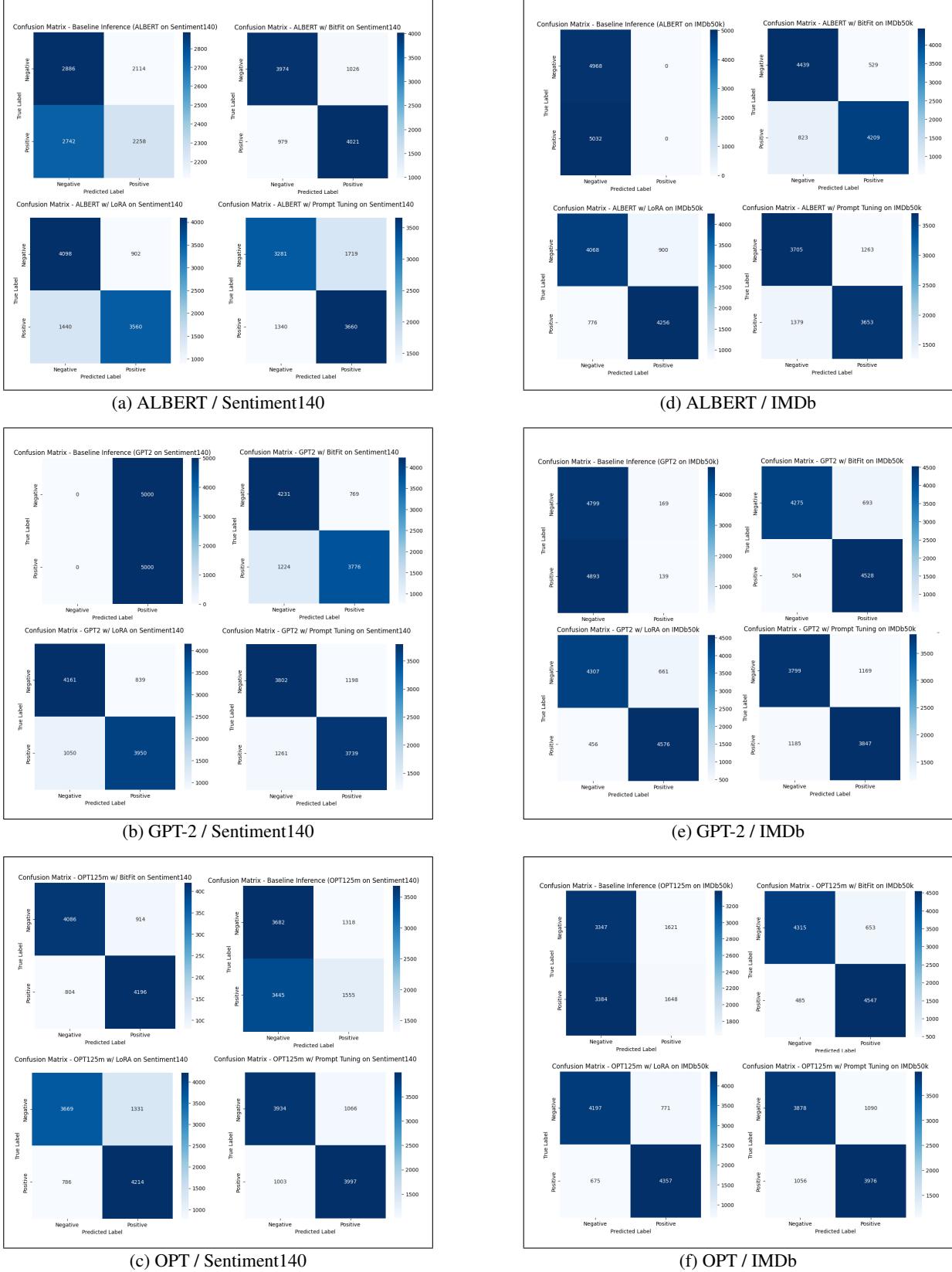


Figure 2: Confusion matrices before and after PEFT. Left column shows Sentiment140 results; right column shows IMDb results. Each panel stacks Baseline, BitFit, LoRA, and Prompt-Tuning results for the indicated model–dataset pair.

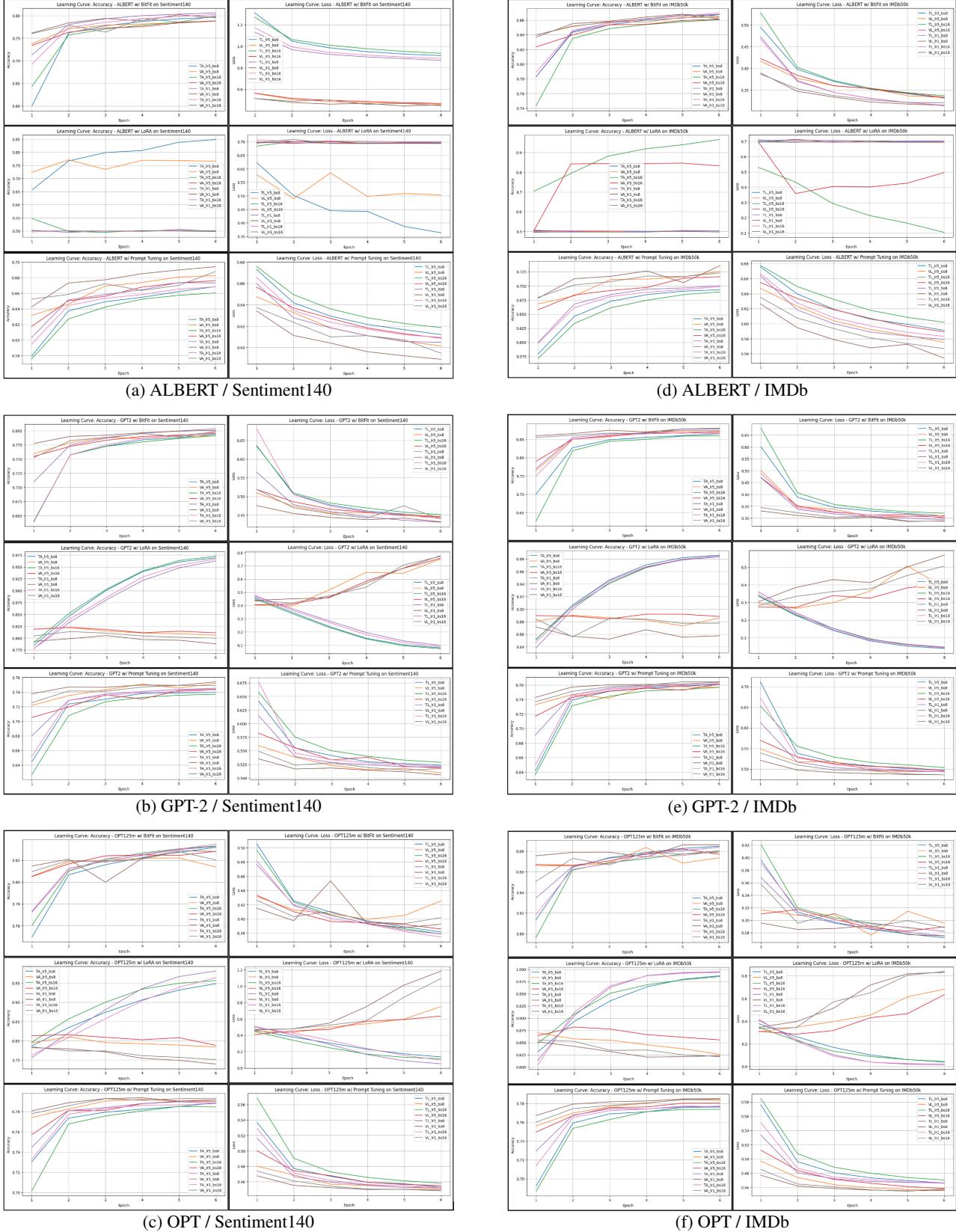


Figure 3: Learning curves depicting – per pairwise (MODEL / DATASET) cluster – accuracy (left) and loss (right) from grid search across batch sizes ( $bs = 8, 16$ ), learning rates ( $lr = 5e - 5, 1e - 4$ ) and for 6 epochs.

**GB** (ALBERT/Sentiment140), validating our goal of resource-constrained fine-tuning.

Overall, BitFit offers the best *accuracy-per-second* on small or medium models, while LoRA provides the absolute accuracy frontier on larger backbones. Prompt Tuning remains a strong choice for rapid experimentation when inference latency is not critical.

## 7. Discussion on Deep Learning Aspects

**Problem structure and model design.** Sentiment analysis is a sequence-classification task; each model ends in a single dense layer that maps the Transformer’s [CLS] (or equivalent) embedding to two logits. Only that head plus (*i*) bias terms (BitFit) or (*ii*) low-rank adapters (LoRA) carry trainable parameters during PEFT, preserving the pre-trained representation power of the frozen backbone.

**Input/Output representations.** Inputs are token ID sequences with fixed padding (128 tokens for tweets, 256 for reviews); outputs are binary class indices. Pre-processing removes URLs, emojis, and user tags to minimise out-of-vocabulary tokens.

**Loss and optimisation.** We use cross-entropy loss—appropriate for mutually exclusive labels—and optimise with AdamW ( $\eta = 2 \times 10^{-5}$ ,  $\beta_{1,2} = 0.9, 0.999$ , weight-decay 0.01). These settings balanced stability and speed across all three models; higher learning rates produced divergence on GPT-2.

**Overfitting and generalisation.** Validation accuracy tracked training accuracy within  $\pm 0.7$  pp, and loss curves flattened rather than rebounding, indicating negligible overfitting. Down-sampling Sentiment140 to 50 k tweets plus early stopping kept model capacity aligned with data volume.

**Hyper-parameter choices.** Key hyper-parameters are (1) LoRA rank ( $r = 4$ ) and scaling  $\alpha = 16$ , (2) BitFit bias-only flag, and (3) prompt length (20 tokens). Preliminary sweeps showed  $r > 8$  yielded  $\geq 0.3$  pp gains for  $\leq 30\%$  extra memory, while prompts longer than 40 tokens began to overfit.

**Framework and code base.** All experiments are implemented in PyTorch 2.1 with the `transformers` (4.38) and `peft` (0.10) libraries. Starting from Hugging Face checkpoints avoided costly re-training, and the datasets API ensured identical splits across runs. Reproducibility is guaranteed via a public Git repository containing scripts, environment files, and fixed random seeds.

## 8. Team Contributions

In this project, each team member contributed to different aspects of the research, from data collection and pre-processing to implementing and fine-tuning models. Table 1 shows a detailed breakdown of each team member’s contributions:

Contributions by Team Member	
Henderson, J.	<ul style="list-style-type: none"> <li>- Spearheaded project team formation</li> <li>- Spearheaded early-term comms</li> <li>- Actively dialogued project ideas</li> <li>- Spearheaded writing and revision of proposal</li> <li>- Revised and added to codebase established by Priya (all documentation, pre-processing, hyperparameter exploration, visualization) for baseline model.</li> <li>- Spearheaded writing and revision of project update</li> <li>- Adapted codebase to include Prompt Tuning as well as IMDb50k, 3x-ing our scope pre-project update.</li> <li>- Ran experiments and generated and analyzed results</li> <li>- Assisted in writing and refinement of final project paper</li> </ul>
Kanuparth, S.	<ul style="list-style-type: none"> <li>- Actively dialogued project ideas</li> <li>- Assisted in revision of proposal</li> <li>- Assisted in revision of project update</li> <li>- Revised and added to codebase (refining/optimizing training workflow with post-project update 3x scope increase.)</li> <li>- Conducted experiments, generated and analyzed results</li> <li>- Spearheaded writing and refinement of final project paper</li> </ul>
Swaminathan, A.	<ul style="list-style-type: none"> <li>- Actively dialogued project ideas</li> <li>- Assisted in revision of proposal</li> <li>- Assisted in revision of project update</li> <li>- Helped run experiments and generate visualizations for final report</li> <li>- Helped consolidate file structure from experiments of test logs, outputs, distributed testing and organization.</li> <li>- Assisted in writing and refinement of final project paper</li> </ul>
Tamilselvan, P.	<ul style="list-style-type: none"> <li>- Actively dialogued project ideas</li> <li>- Assisted in writing and revision of proposal</li> <li>- Spearheaded feedback process seeking advice/direction from course staff and conveying back to rest of group</li> <li>- Assisted in writing and review of project update</li> <li>- Spearheaded mid-to-late-term comms/leadership</li> <li>- Spearheaded starter code to evaluate baseline model/results for midterm proposal/update</li> <li>- Provided support to other team members in their refinement of the base model/testing environment</li> </ul>

Figure 4: Contributions by Team Member

## 9. References

- [1] B. Liu, J. Chan, J. Bradbury, et al., "OPT: Open Pre-trained Transformer Language Models," arXiv preprint arXiv:2205.01068, 2022. [Online]. [Accessed: Mar. 18, 2025].
- [2] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," OpenAI, 2019. [Online]. [Accessed: Mar. 18, 2025].

- [3] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soriciut, "ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations," Proc. of the 8th International Conference on Learning Representations (ICLR), 2020. [Online]. [Accessed: Mar. 18, 2025].
- [4] E.B. Zaken, S. Ravfogel, and Y. Goldberg, "BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models," arXiv preprint arXiv:2106.10199, Jun. 2021. [Online]. [Accessed: Feb. 11, 2025].
- [5] 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. [Online]. [Accessed: Feb. 11, 2025].
- [6] B. Acun and M. Elhoushi, "Efficient LLM Supervised Fine Tuning," proposal, 2025. [Online]. [Accessed: Feb 10, 2025].
- [7] M. Mosbach, T. Pimentel, S. Ravfogal, D. Klakow, and Y. Elazar, "Few-shot Fine-tuning vs. In-context Learning: A Fair Comparison and Evaluation," arXiv preprint arXiv:2305.16938. [Online]. [Accessed: Feb. 11, 2025].
- [8] A. Askell, Y. Bai, A. Chen, D. Drain, D. Ganguli, T. Henighan, et al., "A General Language Assistant as a Laboratory for Alignment," arXiv preprint arXiv:2112.00861. [Accessed: Feb. 11, 2025].
- [9] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," CS224N Project Report, vol. 1, no. 2009, p. 12, 2009. [Online]. [Accessed: Feb. 12, 2025].
- [10] X. Zhang, M. Lewis, and D. Zettlemoyer, "OPT: Open Pretrained Transformer Language Models," arXiv preprint arXiv:2205.01068, 2022. [Online]. [Accessed: Feb. 11, 2025].
- [11] R. Rust et al., "How Effective is Parameter-Efficient Fine-Tuning? A Systematic Benchmark," arXiv preprint, 2023.
- [12] Lester, B., Al-Rfou, R., and Constant, N. "The Power of Scale for Parameter-Efficient Prompt Tuning." arXiv preprint, arXiv:2104.08691, 2021
- [13] P. TamilSelvan, J. Henderson, S. Kanuparth, and A. Swaminathan, "X-PERTS: Parameter-Efficient Refinement for Text-based Sentiment Analysis Tasks," GitHub repository, 2025. [Online]. Available: <https://github.com/Priya-753/X-PERTS>