Fine-Tuning BERT for Word-in-Context (WiC): From Traditional Fine-Tuning to Parameter-Efficient Fine-Tuning with LoRA

Omar Ibrahim / C24029408

Cardiff University
Email: Ibrahimo@cardiff.ac.uk

***Abstract*—This report presents an algorithmic approach to fine-tuning BERT for the Word-in-Context (WiC) task, followed by the advent of an improved version using Low-Rank Adaptation (LoRA). The implementation begins with offset mapping to extract precise token positions, followed by constructing the WiCDataset class for efficient data handling. We then define a custom neural network classifier using PyTorch's nn.Module, train the model, and analyze its performance on the test set. To enhance efficiency, we incorporate LoRA, significantly reducing the number of trainable parameters while improving accuracy. The baseline model, which fine-tunes all of BERT's parameters, achieves 54.21% test accuracy, whereas the LoRA-based model, training only 1.81% of parameters, achieves 65.50%. The results demonstrate that LoRA enables efficient fine-tuning with improved performance.**

I. Introduction

Word sense disambiguation (WSD) is a fundamental challenge in Natural Language Processing (NLP), requiring models to differentiate word senses based on context. The Word-in-Context (WiC) task evaluates a model's ability to determine whether a target word is used with the same sense in two different sentences.

Pre-trained transformer models like BERT (Devlin et al., 2019) have significantly improved contextual word representations, enabling models to adjust word meanings based on sentence context. This report examines the traditional approach of fine-tuning BERT, which involves updating all model parameters for downstream tasks such as WiC. While this method can be effective, it requires excessive computational resources, and it may not necessarily yield high accuracy. To address this challenge, this study explores Low-Rank Adaptation (LoRA) (Hu et al., 2021), a parameter-efficient fine-tuning (PEFT) technique that introduces trainable low-rank matrices into pre-trained layers while keeping most parameters frozen. By comparing both approaches, this study demonstrates that LoRA results in higher accuracy

II. Theoretical Background

A. Word-in-Context (WiC) Task

The WiC task is a binary classification challenge where a model must determine whether a target word has the same meaning across two different sentence contexts. For example:

- Sentence 1: *"The **room** was spacious."*
- Sentence 2: *"There's no **room** for hatred."*

In this example, the word "room" refers to different concepts (physical space vs. capacity), so the correct label is False (different meaning). WiC is a crucial benchmark for evaluating contextual word understanding (Pilehvar and Camacho-Collados, 2019). It is challenging because many words are polysemous (have multiple senses), and only subtle context clues distinguish their senses. Models fed with traditional static word embeddings, such as Word2vec (Mikolov et al., 2013) and GloVe (Pennington, Socher and Manning, 2014), struggle with this task, as they assign a single vector representation to words regardless of context.

B. Fine-tuning Transformers for Contextual Meaning

Modern transformer-based models, such as BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019), excel at modeling contextual meanings, augmented with the attention mechanism.

For the WiC task, a common approach is to feed both sentences into BERT and extract embeddings for the target word in two pragmatically different sentences. These contextualized, more specifically pragmatic, embeddings capture the word's meaning in that specific sentence. Fine-tuning adjusts BERT's parameters such that the relationship between the two embeddings maps to a binary decision: same meaning or different meaning. The classifier often takes as input:

- The concatenation of both embeddings.
- Their element-wise difference.
- Their element-wise product.

Despite its effectiveness, fine-tuning millions of parameters is computationally expensive and risks overfitting, especially in low-resource dataset. Therefore, an alternate approach is presented in this report.

C. Low-Rank Adaptation (LoRA)

Low-Rank Adaptation (LoRA) (Hu et al., 2021) improves fine-tuning efficiency by introducing parameter-efficient adaptations instead of updating all model weights. The key notion is to freeze the pre-trained model's original weights while

injecting small trainable low-rank matrices to approximate task-specific weight updates.

For any weight matrix W in the transformer model (e.g., query and value projection matrices in self-attention layers), LoRA introduces two much smaller matrices A and B (of rank $r \ll \dim(W)$), such that:

$$\Delta W \approx B^T A$$

During training, only A and B are updated, while W remains frozen. This drastically reduces the number of trainable parameters. LoRA enables efficient fine-tuning, saving memory and time while maintaining comparable, sometimes better, performance to full fine-tuning.

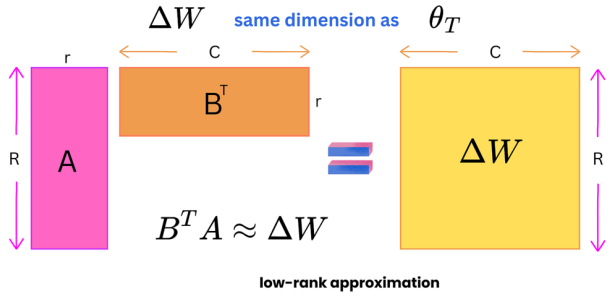

Fig. 1: Low-Rank Adaptation (LoRA) injects two small low-rank matrices (A and B) such that $B^T A \approx \Delta W$, the task-specific weight adjustment. Freezing the original weights θ_T and training only A, B enables efficient fine-tuning with far fewer parameters.

III. Implementation Details

A. Dataset and Preprocessing

in the Word-in-Context Dataset (WiC), each row is presented as:

- Two sentences containing a shared target word.
- Character-level indices indicating the start and the end of the target word in the two sentences.
- A binary label (1 if the target word's meaning is the same in both contexts, 0 otherwise).

Before feeding sentences into BERT, the text is tokenized using the tokenizer of BERT (bert-base-uncased). Since BERT splits words into subword tokens, a function is created to precisely locate the span tokens of the target word. This is done using offset mapping, which provides character-to-token position mappings. Using these mappings, a target attention mask is constructed, which is a binary mask highlighting the tokens corresponding to the target word.

The final output of the function would be a binary tensor mapping the tokens that correspond to the taget word with 1, otherwise 0.

1) WiCDataset Class: To efficiently handle data processing and formatting, a custom dataset class, WiCDataset, is implemented using PyTorch's Dataset module. This class automates:

- **Tokenization**: Converts sentences into BERT-compatible tokenized representations.
- **Target Word Mapping**: Identifies the exact position of the target word using offset mapping.
- **Efficient Batch Processing**: Structures data into tensors that can be loaded into the model.

The dataset class has two key components:

- `__init__()`: Loads the dataset and initializes the tokenizer.
- `__len__()`: Returns the total number of samples in the dataset.
- `__getitem__()`: Converts the raw text into tokenized input, attention masks, and target masks

B. Baseline Model: Full Fine-Tuning

The baseline model is a classifier built on BERT Base (bert-base-uncased), a 12-layer transformer with approximately 110M parameters. The model is trained using full fine-tuning, where all parameters are updated.

Architecture: A custom neural network module (WiCClassifier) is implemented to process the contextual embeddings extracted from BERT. The classifier is designed with the following components:

1) BERT Feature Extraction: The bert-base-uncased model is handled, stored as self.bert within the __init__ method.
2) Feature Concatenation: The model constructs a feature representation by concatenating the contextual embeddings of the target word from both sentences.
3) Classification Layer: A fully connected neural network layer with ReLU activation is used to project the concatenated embeddings into a lower-dimensional space before classification. This is done by doubling the linear size of BERT, given two sentences to analyse.

The model is optimized using **AdamW** with:

- **Learning rate:** 2×10^{-5}
- **Batch size:** 32
- **Training epochs:** 3

Despite its effectiveness, full fine-tuning may risk overfitting, especially in a low-source setting. Therefore, LoRA is presented in the next section.

C. LoRA-Based Model: Parameter-Efficient Fine-Tuning

Low-Rank Adaptation (LoRA) is adapted in this task, where most of the parameters of BERT are frozen and small trainable low-rank matrices are introduced in the model.

Instead of updating all weights in BERT's attention mechanism, LoRA injects trainable matrices into the model, and the standard weight update ΔW is approximated using two smaller matrices A and B, such that:

$$\Delta W \approx B^T A \tag{1}$$

where:

- A and B are low-rank matrices** with rank.
- During training, only A and B are updated, while the original weight matrix remains frozen.

For this experiment, LoRA hyperparamteres are fine-tuned as:

- Rank r = 12
- LoRA alpha α = 24
- Dropout = 10%

To improve word sense similarity modeling, the classifier is modified by incorporating interaction-based features. This enhances feature expressiveness, allowing the model to capture semantic similarities and differences between target words in both contexts.

The classifier consists of:

- A 512-unit fully connected layer with ReLU activation.
- Dropout (40%).
- A final binary softmax classification layer.

D. Training and hyperparameters

The LoRA-based model is trained for up to 5 epochs with early stopping. The optimizer and learning rate scheduler are set as follows:

- Learning rate: 2.5×10^{-5} (slightly higher than baseline).
- Optimizer: AdamW with weight decay (0.01).
- Batch Size: 32.

By training only 1.81% of the parameters, the LoRA model achieves higher accuracy than the fully fine-tuned baseline BERT.

IV. Experiments and Comparisons

For this task, a comparison is conducted between the accuracy of the baseline BERT model and the LoRA-augmented BERT model.

A. Trainable Parameters

Unlike a traditional fine-tuning, which trains on all the parameters, LoRA drastically, with the utilization of the two low-rank matrices, reduces the number of the trainable paramteres to 2,016,770, which translates to 1.81%. This confirms the efficiency gains of LoRA: memory and time are greately minized while surpassing the BERT baseline. Table I summarizes this difference. Notably, despite having 98% fewer adjustable parameters, by introducing low-rank matrices into specific layers of a pre-trained model, LoRA focuses on adapting **essential parameters relevant to the task** while preserving the model's general knowledge (Hu et al., 2021).

B. Training Loss Progression

Both models started with a high loss of approximately 0.68. For the baseline model, loss drastically decreased over 3 epochs, starting from 0.67 and reaching 0.43 by the final epoch, while, however, the validation accuracy remained slightly unchanged. This suggest that the model was overfitting.

For the LoRA model, the loss declined less rapidly:

- Within the **first epoch**, loss dropped to 0.69.
- By **epoch 5**, loss reached 0.55,

The loss curves reflect that the LoRA model improved more quickly and achieved a lower final loss than the baseline, indicating a better fit to the training data. As a deduction, reducing the trainable parameters efficiently converge to a task-specific solution while reducing overfitting.

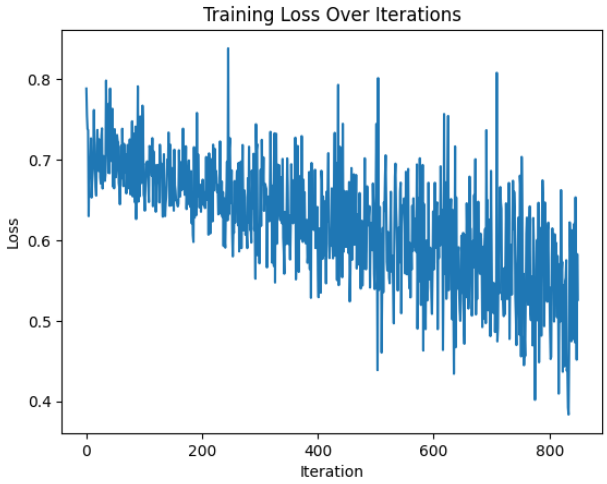

Fig. 2: Training loss progression for the LoRA model over iterations. The loss gradually declines, reaching 0.55 by epoch 5. This indicates that reducing trainable parameters allows the model to efficiently converge to a task-specific solution while mitigating overfitting.

C. Side-by-Side Results

Table I compares compares the the output of the Accuracy metric across the two models. it is transparent that the LoRA-adapted model outperformed the baseline model, resulting in a state-of-the-art result.

TABLE I: Comparison of Baseline vs. LoRA Model

Model	Trainable Params	% of Total	Test Accuracy
Baseline BERT	111M	100%	54.21%
LoRA BERT	2.01M	1.81%	**65.50%**

V. Results and Discussion

A. Key Findings

The experiments demonstrate a clear benefit of using LoRA for the WiC task. By introducing only a small number of parameters, the LoRA-oriented model achieved a higher accuracy. This improvement suggests that for the task of word-in-context, with a low-resource setting, fine-tuning the whole model may risk overfitting and low accuracy.

LoRA facilitated the fine-tuning of BERT, utilizing its pretrained linguistic knowledge, by just just applying a small yet effective adjustment. This allowed the model to capture a task-specific knowledge, the WiC task, while preserving BERT's linguistic understanding.

B. Summary

Overall, LoRA demonstrated its effectiveness in fine-tuning BERT for the WiC dataset by achieving a higher accuracy of 65.50% compared to 54.21% in the baseline, while requiring only 1.81% of the total parameters to be updated. Additionally, it significantly reduced training costs by enabling 2–3× faster convergence with lower computational overhead. The improved performance can be attributed to better generalization, likely due to the regularization effect of freezing most of BERT's parameters, which helped prevent overfitting while maintaining strong task-specific learning.

These results suggest that LoRA should be considered a primary alternative to full fine-tuning, particularly when working with **large pre-trained models on resource-limited setting.

VI. References

References

[1] Camacho-Collados, J. & Pilehvar, M.T. (2019). *WiC: The Word-in-Context Dataset for Evaluating Context-Sensitive Meaning Representations*. NAACL.

[2] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv preprint arXiv:1810.04805.

[3] Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). *LoRA: Low-Rank Adaptation of Large Language Models*. [online] Available at: https://arxiv.org/abs/2106.09685 [Accessed 5 March 2025].

[4] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. arXiv preprint arXiv:1301.3781. Available at: https://arxiv.org/abs/1301.3781 [Accessed 5 March 2025].

[5] Pennington, J., Socher, R., and Manning, C.D. (2014). *GloVe: Global Vectors for Word Representation*. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. Available at: https://aclanthology.org/D14-1162 [Accessed 5 March 2025].

[6] Benveniste, D. (2024). *Fine-Tuning LLMs: From A to Z!*. The AiEdge. [online] Available at: https://newsletter.theaiedge.io/p/fine-tuning-llms-from-a-to-z [Accessed 5 March 2025].