DS-GA-1012: Natural Language Understanding and Computational Semantics, Spring 2025

Efficient Fine-Tuning with BitFit?

Aditeya Baral
N19186654

Problem 1: Setup

Problem 1b: Understand BERT Inputs

The tokenized inputs for BERT include the following three components,

1. `input_ids`
    - This is a **sequence of token IDs representing the input text** (and other special tokens like `[CLS]` and `[SEP]` in the input) where each token ID is mapped to token in the vocabulary.
    - BERT uses WordPiece tokenization, meaning words may be split into smaller sub-word units. **Each token in the vocabulary is converted into a unique identifier and mapped to an integer ID** called the `input_id`.
    - This allows the model to create an index and **look up embeddings** for each token from the embedding layer.
2. `token_type_ids`
    - BERT traditionally takes a pair of sentences as input separated by the `[SEP]` token for its pre-training tasks.
    - Also known as segment IDs, these IDs **indicate which part of the input belongs to which sentence segment**.
        - In tasks involving sentence pairs (for example, next sentence prediction), tokens from the first sentence are usually assigned a `0` and tokens from the second sentence a `1`.
        - If the task involves only a single sentence, this is usually a sequence of all zeros.
    - These IDs help the model **differentiate between two segments** when processing paired inputs.
3. `attention_mask`
    - The attention mask **indicates to the model which tokens should be attended to and which shouldn't** - for example, padding tokens added to reach a uniform length while processing multiple inputs in a batch.
    - Tokens corresponding to actual content are marked with a `1`, while any tokens that shouldn't be attended to are marked with a `0`.
    - This ensures that the model **does not incorporate the unnecessary tokens into its attention calculations**, which is vital for maintaining the accuracy of the Transformer's self-attention mechanism.

Problem 1c: Understand BERT Hyperparameters

To fine-tune the $\text{BERT}_{\text{tiny}}$ model on the GLUE benchmark, hyperparameter tuning was performed using a grid search approach. The process followed the methodology as follows,

- An exhaustive grid search was conducted over a predefined set of hyperparameter values across combinations of **learning rates and batch sizes**.

Batch Sizes	8, 16, 32, 64, 128
Learning Rates	3e-4, 1e-4, 5e-5, 3e-5

Table 1: Hyperparameter search space for batch sizes and learning rates.

- For each task, the best fine-tuning hyperparameters were selected from Table 1 after training each model for **4 epochs using a grid-search over all combinations of hyperparameter values**. The goal was to identify the combination that yielded the best performance on the validation set for each GLUE task.
- Performance was evaluated using **task-specific metrics** like F1 score and accuracy for classification (e.g., MRPC). The best configuration was chosen based on which hyperparameter set maximized the corresponding evaluation metric for a given task on its validation set.
- This allowed the models to achieve their **optimum performance on the GLUE benchmark** by finding the most effective hyperparameters tuned to each task.

Problem 3: Run Experiment

Problem 3a: Train Models

	Validation Accuracy	Learning Rate	Batch Size
Without BitFit	89.28%	3e-4	64
With BitFit	63.32%	3e-4	16

Table 2: Hyperparameter Search on Validation Set results for $\text{BERT}_{\text{tiny}}$ with and without BitFit.

The results in Table 2 indicate that **standard fine-tuning outperforms BitFit** in terms of validation accuracy. This suggests that for this dataset, training all parameters provides a significant performance boost over tuning only bias terms. Additionally, while fine-tuning with BitFit allows us to decrease the batch size, the optimum learning rate is found to be 3e-4 across both approaches.

Problem 3b: Test Models and Report Results

	# Trainable Parameters	Test Accuracy
Without BitFit	4,386,178	87.304%
With BitFit	3,074	63.62%

Table 3: Test results for $\text{BERT}_{\text{tiny}}$ with and without BitFit.

- The results in Table 3 indicate that full fine-tuning i.e, updating every parameter in the model, yielded a test accuracy of 87.304%. In contrast, when using BitFit (where only the bias parameters, 3,074 in total, are updated), test accuracy dropped to 63.62%.
- This suggests that for this dataset, **standard fine-tuning by training all parameters outperforms BitFit and provides a significant performance boost over tuning only bias terms**.
- In Zaken et al. (2020), BitFit performed competitively with full fine-tuning across several tasks, with only a small performance gap in most cases while updating a very small fraction of the parameters. Their findings, particularly on larger Transformer models, indicate that **large models possess inherent redundancy such that updating only the bias parameters is often sufficient** to capture task-specific adaptations.
  - However, **their results were predominantly observed on larger pre-trained Transformer models ($\textbf{BERT}_{\textbf{base}}$ and $\textbf{BERT}_{\textbf{large}}$), which tend to have substantial redundancy in their parameters**. For these larger models with millions of parameters, freezing the majority of parameters (apart from biases) does not drastically impair the model's ability to adapt to new tasks and still allows for effective transfer learning.
  - **Our experiments used $\textbf{BERT}_{\textbf{tiny}}$, a much smaller model** than the standard $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ architectures. In a small model, there is less redundancy, meaning that nearly every parameter is vital for representing task-specific information.
  - Freezing almost all parameters and updating only the biases leaves very little room for the model to adjust its internal representations to a new task. Consequently, **limiting updates to just the bias parameters significantly reduces the model's flexibility**, resulting in lower test accuracy.
- Zaken et al. tested on GLUE and other NLP benchmarks, while we tested on IMDb, a sentiment classification task.
  - **Our evaluation dataset could comprise a distribution that is predominantly different from the GLUE benchmarks** and could potentially hint at a much more challenging dataset for BitFit to generalize well on.
  - BitFit was shown to work well on some NLP tasks, but IMDb sentiment classification may **require deeper representations** for richer contextual understanding and effectively capturing nuances in language like sarcasm, thus **benefiting more from training all parameters**.
- Thus, **while BitFit offers a highly parameter-efficient approach, its benefits appear to be more pronounced for larger, overparameterized models** and hints that it may not always be suitable for all pre-trained Transformers or all tasks.
  - Our findings with $\text{BERT}_{\text{tiny}}$ suggest that such a parameter-efficient approach **may not translate well to smaller models**, where the lack of redundancy causes a noticeable drop in performance. While computationally efficient, its effectiveness may vary based on **task complexity** and **model size** and **smaller models might benefit more from full fine-tuning**.