Zeroth-order optimization for LLM Fine-Tuning

Grigoriy Evseev
MIPT, Moscow
evseev.gv@phystech.edu

Egor Petrov
MIPT, Moscow
petrov.egor.d@phystech.edu

Veprikov Andrey
MIPT, Moscow
ISP RAS, Moscow
veprikov.as@phystech.edu

Aleksandr Beznosikov
MIPT, Moscow
Innopolis University, Innopolis
anbeznosikov@gmail.com

Abstract

In the field of natural language processing, the standard approach is to pre-train large language models (LLMs) using first-order optimization techniques such as SGD and Adam. However, as the size of LLMs increases, the significant memory overhead associated with back-propagation to compute gradients becomes a serious problem due to insufficient memory for training. For this reason, more and more zeroth-order optimization (ZO) methods are being developed, which only require forward pass of the model to compute gradients. In this paper, we present a new ZO approach for LLM pre-training, and compare it with existing methods such as ZO-SGD and ZO-Adam.

Keywords: Zeroth-Order Optimization, Large Language Models (LLMs), Fine-Tuning, Machine Learning.
Highlights:

1. Novel Zeroth-Order Optimization Method

   We propose a novel zeroth-order optimization method developed for fine-tuning large language models (LLMs) is proposed.

2. Comparative Analysis

   A detailed comparison with existing zeroth-order methods such as ZO-SGD and ZO-Adam is made.

3. Memory and Computational Efficiency

   Combine SignSGD and ZO Jaguar method to achieve efficiency in both memory and performance comparing to existing methods.

1 Introduction

Fine-tuned pre-trained large language models (LLMs) are the current standard in solving modern language problems, such as natural language processing (NLP) [1, 2]. First-order (FO) optimizers, e.g., SGD [3] and Adam [4], have been the predominant choices for LLM fine-tuning. However, as the size of the models used increased, the backpropagation (BP) algorithm began to consume a significant amount of memory, making the use of FO algorithms resource-intensive. This problem has led to the problem of obtaining resource-consuming algorithms for the pre-training of LLMs.

To solve the model optimization problem without applying BP algorithm zeroth-order algorithms have been proposed. Despite its new application to LLM fine-tuning, the underlying optimization principle is the function value-based gradient estimate is referred to as the ZO gradient estimation [5, 6, 7, 8, 9]. However, to date, there are still many optimization methods that have not been considered from a ZO optimization perspective. In the work [10] some ZO algorithms, such as ZO SGD, ZO Adam, have been considered and experimentally demonstrated their effectiveness in pre-training LLMs in terms of memory utilization as well as the quality of their results compared to their FO counterparts.

In this work, we propose a zeroth-order optimization method, ZO Jaguar, which adapts FO Jaguar algorithm to a zeroth-order setting. We apply ZO Jaguar to fine-tune large LMs with billions of parameters and show that, both empirically and theoretically, ZO Jaguar can successfully optimize LLMs. Specifically, our contributions are:

1. We construct a ZO analog of the Jaguar algorithm in the context of fine-tuning LLMs and compare it to already researched methods.
2. We experimentally verify X% reduction in memory utilization compared to STA .

2 Problem statement

The paper studies the proble of fine-tuning pre-trained LLMs of the form:

$$\min_{\Delta W \in \mathbb{R}^{n \times d}} \{f(W_0 + \Delta W)\}, \tag{1}$$

where W_0 – pre-trained weights, ΔW – model retraining, which is usually not a full mesh but some targeting modules.

Algorithm 1 `Zeroth-Order Jaguar (ZO-Jaguar)`

Parameters: smoothing parameter τ, step size γ, number of iterations T.
Initialization: generate $X^0 \sim \mathcal{N}(0, \mathbb{I}_{n \times d})$
for t in $1, \ldots, T$: **do**
  Generate $i \sim U[1, n]$
  $z_+ = X^t$
  $(z_+)_i = X_i^t + \tau \cdot 1^d$
  $z_- = X^t$
  $(z_-)_i = X_i^t - \tau \cdot 1^d$
  $\widehat{\nabla} f^{t+1} = \widehat{\nabla} f^t$
  $(\widehat{\nabla} f^{t+1})_i = \text{sign}(f(z_+) - f(z_-)) \cdot \tau \cdot 1^d$
  $X^{t+1} = X^t - \gamma \widehat{\nabla} f^{t+1}$
end for

3 Main Results

4 Computational experiment

In this section, we consider the empirical results of the Algorithm 1. The benchmark consists of evaluating accuracy and memory utilization efficiency of LLM training. We compare the Algorithm 1 with various well-known BP-based algorithms (FO SGD, FO Adam) and BP-free algorithms (ZO SGD, ZO Adam).

LLM fine-tuning tasks, schemes, and models. We consider three tasks of increasing complexity: (1) binary classification on the Stanford Sentiment Treebank v2 (SST2) dataset [11], (2) question answering with the Choice Of Plausible Alternatives (COPA) dataset [12], (3) commonsense reasoning with WinoGrande [13], as well as (4) multi-sentence reading comprehension (MultiRC) [14], which is used exclusively for efficiency evaluation.

For fine-tuning large language models (LLMs) on these tasks, we explore four parameter-efficient fine-tuning (PEFT) schemes:

- Full fine-tuning (FT) — updating all parameters of the pre-trained model.
- LoRA — applying low-rank weight perturbations [15].
- Prefix-tuning (Prefix) — adding learnable parameters to token embeddings [16].
- Prompt-tuning (Prompt) — introducing a set of learnable tokens as an adaptive input for a fixed model [17].

Additionally, we evaluate several representative language models, including Roberta-Large [18], OPT [19], LLaMA2 [20], Vicuna [21], and Mistral [22].

Evaluation metrics. We assess ZO LLM fine-tuning based on two categories of metrics: accuracy and efficiency. Accuracy reflects the model's performance on test data for specific tasks, such as test accuracy in classification. Efficiency encompasses multiple factors, including memory usage (peak memory consumption and GPU cost), query efficiency (i.e., the number of function queries required for ZO optimization), and run-time efficiency. Together, these metrics offer a comprehensive view of the computational resources required for ZO LLM fine-tuning, aiding in the evaluation of its practicality and cost-effectiveness.

ZO fine-tuning on downstream tasks COPA under OPT-13B. Fig. 1 presentes our expected results on the fine-tuning performance on COPA dataset under OPT-13B.

Basic code results. Fig 2 shows the result of applying ZO SGD to the OPT-13B model in the SST2 dataset.

5 Conclusion

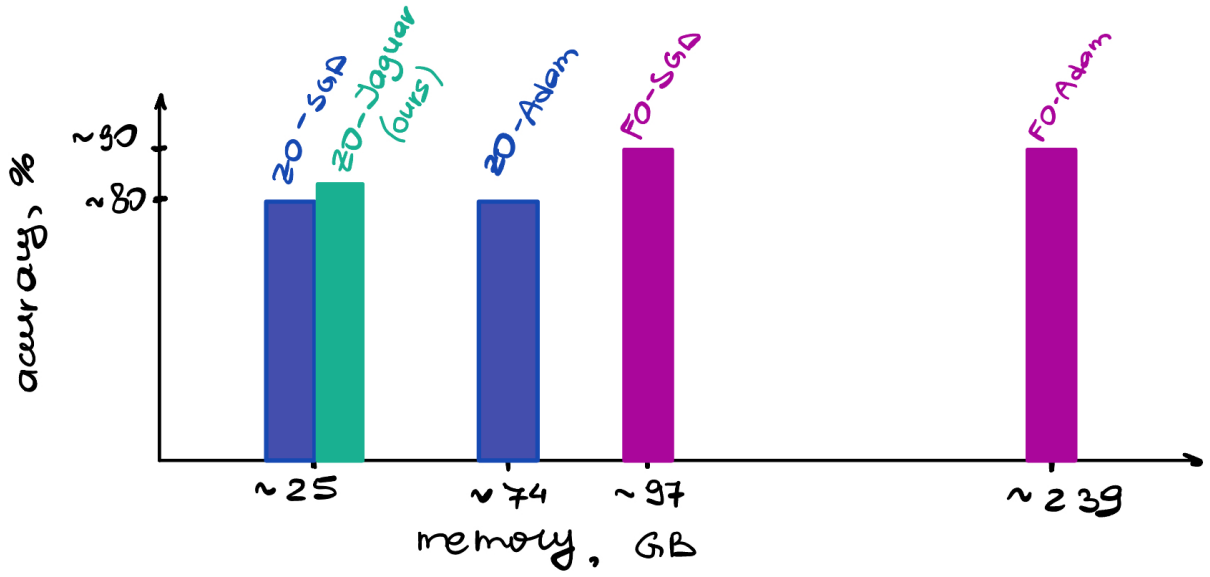

Figure 1: Expected results of OPT-13B on the tasks COPA and WinoGrande fine-tuned using ZO/FO optimizers in different PEFT settings.

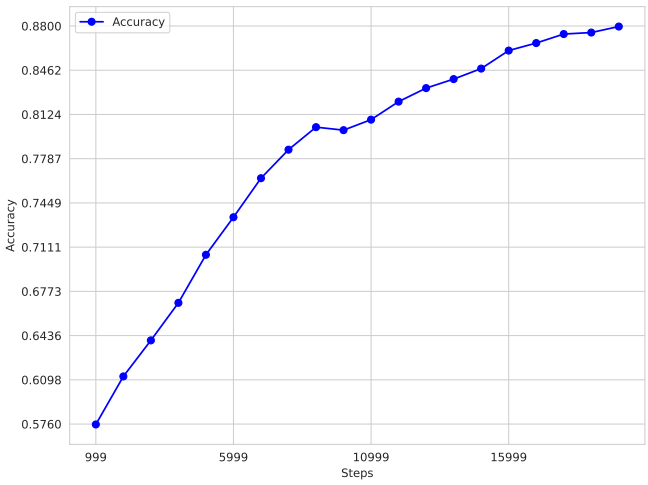

Figure 2: ZO SGD to the OPT-13B model in the SST2 dataset