Research Group Social Computing
School for Computation, Information and Technology
Technical University of Munich

Selective Parameter Updating

SiVA, k-LST, Prompts

Baykam Say, Coşku Barış Coşlu, and Mato Gudelj

Motivation

- Training state-of-the-art language models is a **resource-intensive task**, where even fine-tuning necessitates substantial computational power, **memory**, and **time**
- Costs make it impractical for smaller institutions and individuals to fine tune large language models
- Main goal: practical fine-tuning on **consumer GPUs**
- Excessive computational resources required for training and fine-tuning raise environmental concerns due to the associated energy consumption
- Selectively updating parameters during training can significantly reduce the computational resources required
- Fine-tuning with PEFT can accelerate research iteration time, as experiments are faster and more cost-effective
- There is still **ample room for improvement** in current parameter-efficient fine-tuning (PEFT) methods, as it's a relatively new field
- The importance of PEFT methods will only grow as model size keeps outpacing consumer GPU memory capacity trends

Results

We have evaluated the performance of our methods in terms of accuracy and F1-score on the SST-2 dataset, as well as peak memory usage and training samples per second.

Our SiVA method is closely related to LoRA, while k-LST is a generalization of LST. We show that our methods improve upon their respective baselines. As a result, we present a set of new PEFT methods that cover both the low memory and full fine tuning quality regimes.

Table 1 Comparison of Different Methods (batch size $= 16$)

Method	Accuracy	F1	Memory Usage (MB)	Samples per Second
Full Fine-Tuning	96.44	96.52	8188	51.8
LoRA	96.22	96.30	4526	64.6
LST	93.23	93.33	1779	73.8
Last 3 Layers	94.04	94.18	2117	179.6
SiVA	96.56	96.63	4529	73.3
SiVA - key value	95.76	95.86	2920	106.4
5-LST	94.04	94.04	2209	60.3
9-LST	95.07	95.18	2381	52.4
MeZO	91.74	91.96	1805	54.5
LST + Prompt	94.84	94.98	1803	73.8
9-LST + Prompt	95.41	95.55	2428	52.4
Last 3 Layers + Prompt	94.84	94.87	2122	179.6

- SiVA can be used to achieve a balanced reduction in memory usage and computation time when no compromise can be made on model performance.
- k-LST can be used to achieve the highest reduction in memory usage time with minimal loss of model performance.
- Prompting can be used to further increase k-LST fine-tuning performance at no additional cost.

Singular Value Adaptation (SiVA)

Novel way of updating weight matrices in a lower rank

1. Decompose the original weight matrix into U, S, V matrices using SVD
2. Split U, S, V into smaller training (using the highest r singular values) and reconstruction (rank d) matrices
3. Reconstruct the original model without using the trainable part
4. Train the lower rank U and V matrices
5. During forward pass, reconstruct trainable U, S, V and add it to the original matrix

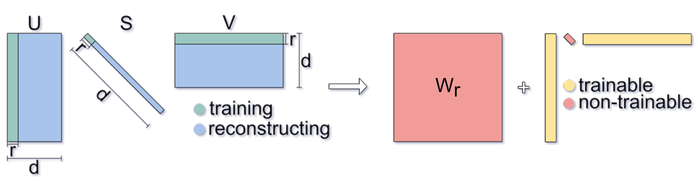

Figure 1 SiVA diagram

- Unlike LoRA, it has no random or zero initialization
- Approximates full fine-tuning as close as possible
- Represents the whole weight matrix and applies updates over it in a lower rank

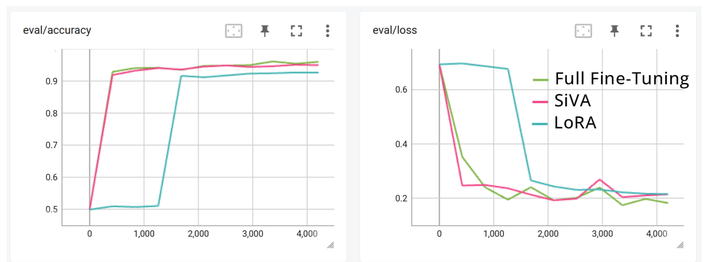

Figure 2 Full fine-tuning, SiVA, and LoRA comparison with the same hyperparameters & initialization

k-Ladder Side Tuning (k-LST)

Memory efficient PEFT method based on LST

- **Trainable side network** as in LST
- Ladder features from k backbone features
- Stacked downsampled backbone features queried with **cross attention**
- $W^q = \theta_{i-1}, W^k = W^v = C_{i-1:i+1}$
- $\theta_i = \text{Attention}(QW^q, KW^k, VW^v)$

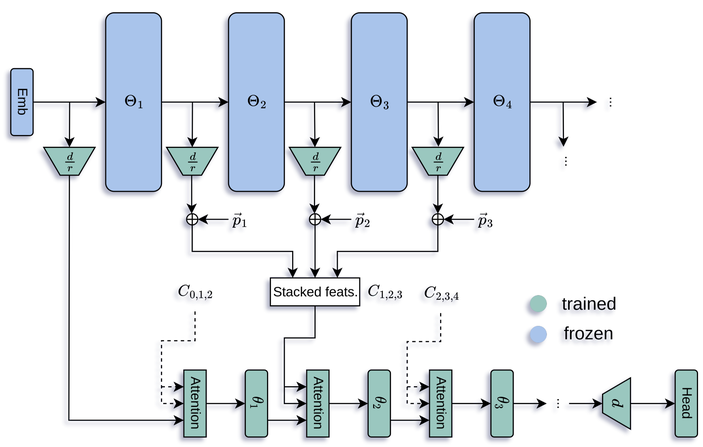

Figure 3 k-LST diagram (k = 3)

Advantages

- No backpropagation through the base model
- Completely **unmodified base model** forward pass
- Comparable results to other PEFT methods at a **lower memory footprint**
- Tunable memory-performance tradeoff with r and k

MeZO and Training with Prompts

Memory-efficient Zeroth-order Optimizer (MeZO):

- At each training step, does two forward passes with different amounts of perturbation on parameters
- Estimates gradients using only those forward passes (no backpropagation)
- Very memory-efficient but high computation time
- Works under a prompt-based fine-tuning setting

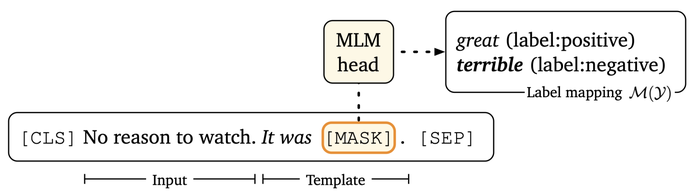

Figure 4 Prompt-based Fine-tuning

Some MeZO Ideas:

- Before training with LST, fine-tune backbone model using MeZO
- Fine-tune entire model using MeZO, then fine-tune last few layers with backpropagation.

Experiment Results:

- Adding a prompt to input sentences improves performance for both LST and layer freezing
- Only the added prompt seems to be relevant for this improvement and not the prior training with MeZO.
- The choice of prompt has a significant impact on the performance gains.