SARA: Single Head Attention-based Random Matrix Adaptation

Anonymous ACL submission

Abstract

Fully fine-tuning large language models by updating all parameters is both computationally expensive and storage-intensive, particularly when deploying multiple task-specific models. Existing parameter-efficient fine-tuning (PEFT) methods, such as LoRA, reduce the number of trainable parameters via low-rank adaptations, yet they still face scalability challenges as model sizes increase. In this work, we introduce SARA (Single-Head Attention-based Random Matrix Adaptation), a novel PEFT approach that leverages random matrices combined with a single-head attention mechanism to further minimize trainable parameters while preserving competitive performance. By integrating with frozen pretrained weights and fine-tuning only a minimal set of additional parameters, SARA offers significant memory savings without compromising accuracy. We validate our method through experiments on two standard benchmarks: the GLUE benchmark for natural language understanding and the E2E challenge for natural language generation. Our results demonstrate that SARA achieves competitive performance with a substantially reduced parameter footprint, making it a promising solution for resource-constrained model adaptation.

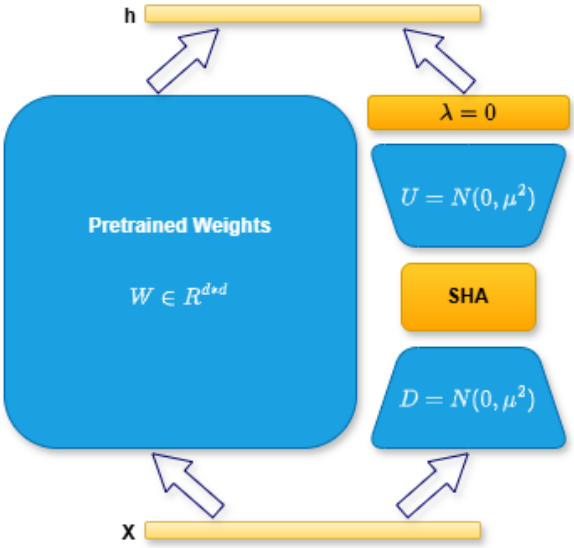

Figure 1: Architecture of the proposed SARA method. SARA trains only the Single-Head Attention (SHA) and the scale vector λ, while freezing and sharing low-rank matrices across layers. The scale vector λ is learnable, and SARA reduces trainable parameters significantly. Like LoRA, the low-rank matrices, attention mechanism, and λ can be merged into the weight matrix W without added latency. The figure shows input x, hidden states h, and frozen matrices U (up-projection) and D (down-projection).

1 Introduction

Despite their impressive capabilities, modern pretrained language models remain challenging to adapt to new tasks due to the heavy computational and storage demands of full fine-tuning. Early work by Aghajanyan et al. (2021) revealed that these models have low intrinsic dimensionality, meaning that task-specific knowledge can be encoded within a small subspace of the full parameter space. This insight inspired methods like prefix tuning (Li and Liang, 2021), which optimize soft prompts in a compressed subspace before projecting them to the original dimension. Building on this idea, low-rank adaptation (LoRA) (Hu et al., 2022) introduces trainable low-rank matrices into transformer layers to significantly reduce the number of parameters.

Despite LoRA's impressive parameter reduction, the intrinsic dimensionality insight suggests that even fewer parameters may suffice. Recent advances like VeRA (Kopiczko et al., 2024) further address scalability by replacing layer-specific matrices with shared random frozen projections and trainable scaling vectors.

We propose SARA, Single-head Attention-based Random matrix Adaptation, which combines the efficiency of random projections with dynamic interaction modeling. As illustrated in Figure 1,

SARA first projects inputs into a low-dimensional space using frozen random matrices, then employs a lightweight single-head attention (SHA) mechanism (Vaswani et al., 2023) to capture task-relevant relationships. Finally, the adapted representations are scaled and projected back to the original space. This approach reduces parameters significantly compared to LoRA.

We evaluate SARA on GLUE and E2E benchmarks using RoBERTa and GPT-2 models. Our experimental results show its effectiveness; despite having fewer parameters, it achieves comparable performance to LoRA. Our contributions are summarized as follows: (i) We introduce a novel PEFT method with no additional inference cost, further reducing trainable parameters compared to LoRA and VeRA while achieving comparable performance. (ii) We validate SARA on two standard benchmarks (GLUE and E2E), applying to RoBERTa and GPT-2 models, demonstrating its effectiveness and scalability.

2 Related Work

We introduce a new PEFT method, broadly categorized into four types:

Adapter-based Methods add small trainable modules to frozen layers, as in Houlsby et al. (2019) and He et al. (2022). While effective, they often increase inference latency.

Prompt-based Approaches prepend learnable tokens to inputs, as in Lester et al. (2021), Li and Liang (2021), and Razdaibiedina et al. (2023). These methods avoid architectural changes but can be sensitive to initialization.

Representation Editing Techniques modify hidden representations for task adaptation. For example, Wu et al. (2024b) introduces ReFT and LoReFT, while Wu et al. (2024a) proposes RED, focusing on scaling and bias vectors for efficiency.

LoRA-based Methods Inject trainable low-rank matrices into frozen layers. LoRA (Hu et al., 2022) reduces parameters while maintaining performance. Variants like DoRA (Liu et al., 2024), VeRA (Kopiczko et al., 2024), and NoLA (Koohpayegani et al., 2024) further optimize this approach. Our work builds on these advancements.

3 Method

We introduce **SARA** (Single-Head Attention-based Random Matrix Adaptation), a parameter-efficient fine-tuning method that uses frozen pre-trained weights and low-rank adaptations, similar to LoRA (Hu et al., 2022). SARA integrates a single-head attention (SHA) mechanism (Vaswani et al., 2023) in a low-dimensional space to capture task-specific interactions dynamically. As shown in Figure 1, SARA adapts a pre-trained linear layer with weight matrix $W_0 \in \mathbb{R}^{d\times k}$ by adding a lightweight adaptation ΔW, so that $W' = W_0 + \Delta W$. The input $x \in \mathbb{R}^d$ is projected into a lower-dimensional space using a frozen matrix $D \in \mathbb{R}^{d\times r}$. SHA captures interactions in this space, and the result is mapped back to the original dimension with matrix $U \in \mathbb{R}^{r\times d}$ and scaled by a trainable vector $\lambda \in \mathbb{R}^d$. The adaptation is:

$$\Delta W = \lambda \odot U \cdot \text{SHA}(D^\top x),$$

where $\odot$ is element-wise multiplication.

3.1 Parameter Efficiency

SARA minimizes trainable parameters by freezing pretrained weights and random matrices, training only the SHA parameters and scaling vector. For each layer, the trainable parameters include SHA weights (W_Q, W_K, W_V, W_O) and λ. Assuming $h = r/2$, the total trainable parameters per layer are:

$$\text{Params}_{\text{SARA}} = 2r^2 + d.$$

In contrast, LoRA uses trainable low-rank matrices for Query and Value projections, resulting in:

$$\text{Params}_{\text{LoRA}} = 4rd.$$

SARA's design reduces memory usage while maintaining effective adaptation.

3.2 Initialization Strategies

Initialization is crucial for SARA's performance and stability. The down projection matrix D and up projection matrix U are initialized using Kaiming initialization (He et al., 2015) and remain frozen during training, ensuring a consistent adaptation basis. The attention mechanism parameters (W_Q, W_K, W_V, W_O) are also initialized with Kaiming initialization to maintain balanced variance and stable training dynamics. The scaling vector λ is set to a small constant (0.001), allowing pre-trained

Model	Rank (r)	SARA Params	LoRA Params	SARA Memory (MB)	LoRA Memory (MB)
RoBERTa-base	4	19,200	147,456	0.07	0.56
	8	25,504	294,912	0.08	1.13
	16	30,720	589,824	0.12	2.25
RoBERTa-large	4	50,688	393,216	0.19	1.50
	8	55,296	786,432	0.21	3.00
	16	73,728	1,572,864	0.28	6.00

Table 1: Comparison of Trainable Parameters and Memory Usage for SARA and LoRA across Different Ranks for RoBERTa-base, RoBERTa-large, PEFT modules are added to query and value layers.

weights to dominate initially while gradually increasing task-specific influence. Ablation studies confirm that Kaiming initialization outperforms alternatives like uniform or standard normal distributions.

4 Experiments

We evaluate SARA's performance and efficiency on both the GLUE benchmark (Wang et al., 2019) for language understanding and the E2E benchmark (Novikova et al., 2017) for generation. For language understanding, we fine-tune RoBERTa base and large models (Liu et al., 2019), and for generation tasks, we fine-tune GPT-2 Medium and Large models (Radford et al., 2019).

4.1 Baselines

To ensure a comprehensive evaluation of SARA, we compare it with a range of established parameter-efficient fine-tuning (PEFT) baselines:

- **Fine-Tuning (FT)**: Fine-tuning (FT) involves training models by updating all of their parameters. A variation of FT, introduced by (Lee et al., 2019), selectively updates specific layers while freezing others. This method specifically adapts only the final two layers, referred to as FTtop2.
- **BitFit**: Adjusts only bias parameters (Zaken et al., 2022).
- **Adapter**: Inserts small trainable layers between transformer layers while keeping the main model frozen (Houlsby et al., 2019).
- **Adapter-FFN**: Modifies only the feed-forward components within adapters for enhanced efficiency (Pfeiffer et al., 2021).
- **Prompt Tuning (PT)**: Tunes a set of prompt tokens appended to the input without updating core model weights (Lester et al., 2021).
- **Prefix Tuning**: Adjusts prepended prefix tokens to guide model behavior.
- **LoRA**: Uses low-rank updates for each transformer layer to reduce the number of trainable parameters (Hu et al., 2022).
- **VeRA**: Shares low-rank matrices across layers with additional layer-specific scaling vectors (Kopiczko et al., 2024).
- **RED**: Employs scaling and bias vectors to efficiently control network outputs.(Wu et al., 2024a)
- **ReFT**: Fine-tunes hidden representations instead of full model weights.(Wu et al., 2024b)

4.2 GLUE Benchmark

Evaluation Strategy We evaluate SARA on the GLUE benchmark (Wang et al., 2019) using RoBERTa-base (125M) and RoBERTa-large (350M) for sequence classification tasks. Following Wu et al. (2024a), we split the validation set in half with a fixed seed, select the model with the highest validation accuracy on one half, and report test accuracy on the other.

Hyperparameter Tuning Hyperparameters are tuned per task using a constant seed, with results averaged over that seed and four additional unseen seeds. Details are provided in the Appendix.

Results Analysis Table 2 shows SARA's performance on GLUE. For RoBERTa-base, SARA achieves 83.7 (vs. FT's 84.4) using only 0.015% of trainable parameters, comparable to RED and LoReFT (both 83.9). For RoBERTa-large, SARA scores 87.7 (vs. FT's 88.0), aligning with other PEFT methods. As a LoRA-based approach, SARA uses significantly fewer parameters than representation editing methods while delivering competitive results.

4.3 E2E Benchmark

Evaluation Strategy We evaluate *SARA* on the E2E benchmark, which tests generation task by converting structured data into coherent text. We

	Method	Params(%)	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg.
BASE	FT*	100%	94.4	87.9	62.4	92.5	78.3	90.6	84.4
	BitFit*	0.080%	94.0	88.0	54.0	91.0	69.8	89.5	81.1
	Adapter*	0.318%	93.3	88.4	60.9	92.5	76.5	90.5	83.7
	Adapter-FFN*	0.239%	93.0	88.8	58.5	92.0	77.7	90.4	83.4
	LoRA*	0.239%	93.9	88.7	59.7	92.6	75.3	90.3	83.4
	RED*	0.016%	93.9	89.2	61.0	90.7	78.0	90.4	83.9
	DiReFT†	0.015%	92.6	88.3	58.6	91.3	76.4	89.3	82.8
	LoReFT†	0.015%	93.4	89.2	60.4	91.2	79.0	90.0	83.9
	SARA	0.015%	93.7	88.7	60.8	90.7	78.3	90.0	83.7
LARGE	FT*	100%	96.0	91.7	68.2	93.8	85.8	92.6	88.0
	Adapter*	0.254%	95.2	90.5	65.4	94.6	85.3	91.5	87.1
	Adapter-FFN*	0.225%	96.1	90.5	64.4	94.3	84.8	90.2	86.7
	LoRA*	0.225%	96.0	89.8	65.5	94.7	86.3	91.7	87.3
	RED*	0.014%	96.0	90.3	68.1	93.5	86.2	91.3	87.6
	DiReFT†	0.014%	95.4	88.5	66.7	93.9	86.9	91.2	87.1
	LoReFT†	0.014%	96.0	90.1	68.0	94.1	87.5	91.6	87.9
	SARA	0.015%	95.6	90.3	67.1	93.4	87.8	91.9	87.7

Table 2: Accuracy comparison of RoBERTa-base and RoBERTa-large against existing PEFT methods on the GLUE benchmark.*Performance results of all baseline methods are taken from (Wu et al., 2024a) and † performance results of baselines taken from (Wu et al., 2024b). We report averaged performance of five runs with distinct random seeds for our method.

	Method	#Params	BLEU	MET	R-L
Medium	FT*	355M	65.9	45.9	69.1
	FT^{top2}*	25.2M	65.9	44.3	68.8
	Adapter^{D}*	0.9M	64.3	44.9	67.7
	LoRA^{D}*	0.8M	67.4	46.0	69.6
	Adap-FFN*	0.8M	64.4	44.7	67.5
	Prefix-Tune*	0.8M	63.9	41.8	66.9
	IA3*	0.17M	63.6	40.5	66.4
	RED*	0.05M	64.9	45.0	67.6
	SARA	0.08M	64.4	43.9	67.5
Large	FT*	774M	65.6	45.4	68.4
	Adapter*	1.8M	65.9	45.8	68.6
	LoRA*	1.5M	68.2	46.2	69.9
	Adap-FFN*	1.5M	65.5	45.6	68.5
	Prefix-Tune*	1.5M	65.5	44.0	67.3
	IA3*	0.32M	65.1	42.8	66.8
	RED*	0.09M	65.8	46.1	69.0
	SARA	0.15M	65.5	44.4	68.3

Table 3: Performance comparison of GPT-2 medium and large models fine-tuned by SARA and other PEFT baselines on the E2E NLG Challenge. *Baseline results are from (Wu et al., 2024a). Adap-FFN: Adapter-FFN, P-Tuning: Prefix Tuning

finetune GPT-2 Medium and GPT-2 Large on the E2E dataset, selecting the best checkpoint based on validation performance and reporting its test accuracy. Each experiment is repeated over three random seeds, with average performance reported.

Hyperparameter Tuning SARA's hyperparameters are tuned using a fixed seed and then evaluated on two additional seeds to ensure robustness. Final results are averaged over all three seeds.

Results Table 3 summarizes SARA's performance on the E2E benchmark. We report standard generation metrics along with the number of trainable parameters for each method. Our experiments show that SARA attains competitive scores relative to established baselines, demonstrating its ability to generate coherent and fluent text while significantly reducing parameter counts. Notably, when applied to GPT-2 Medium and GPT-2 Large models, SARA maintains a favorable balance between performance and efficiency, validating its effectiveness in parameter-efficient settings for natural language generation.

5 Conclusion

We introduce **SARA**, a novel memory-efficient fine-tuning method that reduces parameters compared to LoRA and VeRA while achieving competitive performance on benchmarks like **GLUE**, and **E2E**. SARA combines random matrices with single-head attention to capture task-specific information in a low-dimensional space, enabling scalable adaptation for multi-task and personalized AI. Future work will extend SARA to other architectures and larger models, and further optimize parameter allocation and initialization.

Limitations

This work has several limitations:

- While our experiments are comprehensive, they are limited to smaller LMs compared to the current SOTA LLMs. Evaluating our approach using larger models could strengthen our work.
- Similarly, our datasets are also limited in terms of input and output formats and types. Evaluating on instruction following tasks like different types of reasoning is something that could help us understand how our SARA method compares with other PEFT methods in these settings.
- The language of our experiments is limited to English. This also limits our experimental findings.

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Towards a unified view of parameter-efficient transfer learning.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Preprint*, arXiv:1502.01852.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Soroush Abbasi Koohpayegani, KL Navaneet, Parsa Nooralinejad, Soheil Kolouri, and Hamed Pirsiavash. 2024. Nola: Compressing lora using linear combination of random basis. *Preprint*, arXiv:2310.02556.

Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. 2024. Vera: Vector-based random matrix adaptation.

Jaejun Lee, Raphael Tang, and Jimmy Lin. 2019. What would elsa do? freezing layers during transformer fine-tuning. *Preprint*, arXiv:1911.03090.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *Preprint*, arXiv:2104.08691.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *Preprint*, arXiv:2101.00190.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. Dora: Weight-decomposed low-rank adaptation. *Preprint*, arXiv:2402.09353.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. *Preprint*, arXiv:1706.09254.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. *Preprint*, arXiv:2005.00247.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, Jimmy Ba, and Amjad Almahairi. 2023. Residual prompt tuning: Improving prompt tuning with residual reparameterization. *Preprint*, arXiv:2305.03937.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need. *Preprint*, arXiv:1706.03762.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. *Preprint*, arXiv:1804.07461.

Muling Wu, Wenhao Liu, Xiaohua Wang, Tianlong Li, Changze Lv, Zixuan Ling, Jianhao Zhu, Cenyuan Zhang, Xiaoqing Zheng, and Xuanjing Huang. 2024a. Advancing parameter efficiency in fine-tuning via representation editing. *Preprint*, arXiv:2402.15179.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. 2024b. Reft: Representation finetuning for language models. *Preprint*, arXiv:2404.03592.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *Preprint*, arXiv:2106.10199.

Appendix

A Datasets

A.1 GLUE Benchmark

We follow (Wu et al., 2024a) for evaluating models on the GLUE validation set. The validation set is split into two subsets: one for in-training evaluation and the other for testing. After each training epoch, the model is evaluated on the in-training subset, and the best-performing model across all epochs is selected for testing. For datasets with large validation sets like QNLI, we use 1,000 samples for in-training evaluation. For smaller datasets, we use half of the validation samples. The evaluation metrics include the Matthews correlation coefficient for CoLA, the Pearson correlation coefficient for STS-B, and accuracy for the remaining datasets. Due to time constraints, we remove the time-intensive MNLI and QQP tasks.

A.2 E2E Challenge

The E2E NLG Challenge, introduced by (Novikova et al., 2017), was designed to train and evaluate end-to-end, data-driven natural language generation models. For our experiments, we utilized datasets from Hugging Face Datasets. This benchmark consists of 42.1K training instances, 4.67K validation instances, and 4.69K test instances. Following prior work, we used the official evaluation script to make sure BLEU, NIST, METEOR, ROUGE-L, and CIDEr scores.

B Hyperparameters

B.1 GLUE Benchmark

We conduct hyperparameter tuning with RoBERTa-base and RoBERTa-large for each task individually, selecting hyperparameters based on performance on the held-out validation set with a fixed random seed of 42. To ensure robustness, we further evaluate our model using four additional unseen seeds {43, 44, 45, 46}. We follow the evaluation setup of (Wu et al., 2024a). As noted in (Wu et al., 2024a), we also observe instability in evaluation results on RTE and MRPC due to their small dataset sizes. To ensure a fair comparison, we replace certain random seeds following their methodology. The hyperparameters used for RoBERTa-base are detailed in Table 4, while those for RoBERTa-large are provided in Table 5. Additionally, we conduct further experiments on the GLUE benchmark following the methodology of (Hu et al., 2022). This experiment is designed to compare our results with VeRA. Specifically, we evaluate models using only the validation set and record the best epoch's performance. To provide a reliable comparison, we report the median performance after five runs with distinct random seeds. Table 7 presents our results alongside VeRA and the baseline numbers reported in VeRA's paper.

B.2 E2E Challenge

We conduct hyperparameter tuning with GPT-2 medium and GPT-2 large for e2e challenge, selecting hyperparameters based on performance on the held-out validation set with a fixed random seed of 42. To ensure robustness, we further evaluate our model using two additional unseen seeds {43, 44}. We follow the evaluation setup of (Wu et al., 2024a). The hyperparameters used in our experiments for GPT-2 medium and GPT-2 large are detailed in Table 6.

Table 4: Hyperparameters for RoBERTa-base on the GLUE benchmark. Different learning rates were used for the classifier head and SARA's learnable parameters.

Hyperparameter	**STS-B**	**RTE**	**MRPC**	**CoLA**	**SST2**	**QNLI**
Learning Rate (SARA)	1.00E-02	6.00E-03	3.00E-03	3.00E-03	4.00E-03	8.00E-03
Learning Rate (Classifier)	1.00E-02	6.00E-03	9.00E-03	3.00E-03	4.00E-03	8.00E-03
Max Seq. Len.	256	256	256	256	256	256
Warmup Ratio	0.06	0.06	0.06	0.06	0.06	0.06
LR-Scheduler	linear	linear	linear	linear	linear	linear
Batch Size	32	32	32	16	64	64
Epochs	80	80	20	80	60	25
Scale Vector (λ)	0.001	0.001	0.001	0	0.001	0.001
Rank	8	8	8	8	8	8
α	8	8	16	8	8	8
Dropout	0.1	0.1	0.1	0.1	0.1	0.1

Table 5: Hyperparameters for RoBERTa-large on the GLUE benchmark. Different learning rates were used for the classifier head and SARA's learnable parameters.

Hyperparameter	**STS-B**	**RTE**	**MRPC**	**CoLA**	**SST2**	**QNLI**
Learning Rate (SARA)	3.00E-03	4.00E-03	3.00E-03	6.00E-03	3.00E-03	3.00E-03
Learning Rate (Classifier)	1.00E-04	4.00E-03	3.00E-03	1.00E-04	1.00E-04	3.00E-03
Max Seq. Len.	256	256	256	256	256	256
Warmup Ratio	0.06	0.06	0.06	0.6	0.06	0.06
LR-Scheduler	linear	linear	linear	linear	linear	linear
Batch Size	32	32	32	32	32	32
Epochs	20	40	20	40	10	10
Scale Vector (λ)	0.001	0.001	0.001	0.001	0.001	0.001
Rank	8	8	8	8	8	8
α	8	8	8	8	8	8
Dropout	0.1	0.1	0.1	0.1	0.1	0.1

Table 6: Hyperparameter configurations for SARA on the E2E benchmark for GPT-2 Medium and Large models.

Hyperparameter	**Medium**	**Large**
Optimizer	AdamW	
Learning Rate Schedule	Linear	
Weight Decay	0.01	
Batch Size	10	
Epochs	5	
Warmup Steps	500	
Gradient Accumulation	16	
Label Smoothing	0.1	
Rank	8	
α	16	
Learning Rate	3E-2	1E-2

Table 7: Accuracy comparison of RoBERTa-base and RoBERTa-large against existing PEFT methods on the GLUE benchmark. *Performance results of all baseline methods are taken from Kopiczko et al. [2024] and † performance resullts of baselines taken from Wu et al.[2024]. To ensure a fair comparison, we report median performance of five runs with distinct random seeds for our method.

	Method	**Params(%)**	**SST-2**	**MRPC**	**CoLA**	**QNLI**	**RTE**	**STS-B**	**Avg.**
BASE	FT*	100%	94.8	90.2	63.6	92.8	78.7	91.2	85.2
	BitFit*	0.080%	93.7	92.7	62.0	91.8	81.5	90.8	85.4
	Adpt^D*	0.239%	94.2	88.5	60.8	93.1	71.5	89.7	83.0
	Adpt^D*	0.717%	94.7	88.4	62.6	93.0	75.9	90.3	84.2
	LoRA*	0.239%	95.1	89.7	63.4	93.3	86.6	91.5	86.6
	VeRA*	0.034%	94.6	89.5	65.6	91.8	78.8	90.7	85.2
	DiReFT†	0.015%	92.2	88.7	59.5	91.3	77.0	89.6	83.0
	LoReFT†	0.015%	93.6	87.8	59.1	91.3	79.9	90.0	83.6
	SARA	0.015%	93.8	89.2	62.0	91.3	79.4	89.6	84.2
LARGE	Adpt^P*	0.845%	96.1	90.2	68.3	94.8	83.8	92.1	87.6
	Adpt^P*	0.225%	96.6	89.7	67.8	94.8	80.1	91.9	86.8
	Adpt^H*	1.690%	96.2	88.7	66.5	94.7	83.4	91.0	86.8
	Adpt^H*	0.225%	96.3	87.7	66.3	94.7	72.9	91.5	84.9
	LoRA*	0.225%	96.2	90.2	68.2	94.8	85.2	92.3	87.8
	LoRA-FA*	1.042%	96.0	90.0	68.0	94.4	86.1	92.0	87.8
	VeRA*	0.017%	96.1	90.9	68.0	94.4	85.9	91.7	87.8
	DiReFT†	0.014%	95.2	88.2	66.7	94.0	86.3	91.0	86.9
	LoReFT†	0.014%	96.1	90.2	68.2	94.1	87.8	91.5	88.0
	SARA	0.015%	96.2	90.5	67.4	93.7	85.9	91.4	87.5