TOU: Truncated-factorized reduction for an efficient-parameter model fine-tuning

Phuong Thi-Mai Nguyen,*, Minh-Son Dao, Koji Zettsu
Big Data Integration Center
National Institute of Information and Communications Technology
Tokyo, Japan
maiphuong,dao,zettsu@nict.go.jp

Abstract

The fine-tuning of large-scale pre-trained models represents an effective approach to the transfer of knowledge to new tasks. Nevertheless, this approach typically necessitates the updating of all model parameters, which can result in considerable computational and memory costs. We put forth a methodology, designated as TOU, which employs truncated SVD to decompose weight matrices for comprehensive model fine-tuning. The objective of this method is to retain the benefits of full fine-tuning while reducing the computational and memory costs. Rather than updating the full weight matrices directly, weight matrices are factorized into low-rank components using truncated SVD, freezing one of two factored matrices, thereby enabling the efficient adaptation of the entire model. This significantly reduces the number of trainable parameters, leading to faster training and reduced memory usage. After fine-tuning, TOU accepts reconstructing to recover the structure of original model without any loss of performance. TOU utilises low-rank factorization of a reshaped and reorganised weight matrix to create space-efficient and expressive linear layers. Experiments on Vision Transformer models show that our method achieves a 70% reduction in trainable parameters while maintaining (accuracy drops $< 1\%$), reduces 65% in term of training time, and 27% in term of memory usage comparable performance to full weight fine-tuning. Furthermore, TOU yields better performance than LoRA in terms of accuracy, training speed, and memory usage when setting the same target fine-tuned layers.

1 Introduction

Model fine-tuning is an essential technique for transferring knowledge from large-scale pre-trained models to new tasks, particularly in natural language processing (NLP) and computer vision . The traditional approach is full model fine-tuning, whereby all the parameters of the pre-trained model are updated for the target task. While effective, this approach is not without its drawbacks. It is, after all, computationally expensive, particularly for large models such as BERT or GPT (1; 2; 3). It also requires significant memory and training time. While full fine-tuning is expensive, it is essential in cases where task performance is paramount, when adapting to a domain that is significantly different from the pre-training data, or when tasks require deep, task-specific knowledge (4). In recent years, several approaches have been proposed to make fine-tuning more efficient, scalable, and effective. These address challenges such as computational costs, memory efficiency, and generalization to new domains (5). Efficient fine-tuning techniques have gained significant attention, especially in the context of large-scale pre-trained models (6). Early approaches like full fine-tuning, where all model parameters are updated, were computationally expensive and required extensive resources (7).

*Corresponding author.

38th Workshop on Fine-Tuning in Machine Learning (NeurIPS 2024).

Adapter tuning is one of the first methods to fine-tune pre-trained models by updating a small number of parameters(6). It introduces adapter layers with a bottleneck structure between the layers of pre-trained language models (PLMs), and only these newly added layers are adjusted during fine-tuning. Prefix-tuning fine-tunes PLMs by modifying parameters that are prepended to each transformer layer (8). Inspired by GPT-3, prompt tuning (3) focuses on adding and updating task-specific parameters in the input embeddings. BitFit ((9)) updates only the bias terms in PLMs, leaving the rest of the model unchanged.

More recent, Low-rank factorization is one of powerful approach to parameter-efficient fine-tuning of large-scale models (10). By decomposing weight matrices into smaller, low-rank representations, these techniques reduce memory and computational costs while maintaining or even improving task performance (11; 12). Shapeshifter, a Parameter-Efficient Transformer using factorized reshaped matrices presents a groundbreaking approach to reducing the number of parameters in transformers by factorizing weight matrices into smaller, reshaped components (13). Shapeshifter increases training time due to several factors related to the method's use of factorized reshaped matrices (13).

The most prominent methods include adapter modules, Low-Rank Adaptation (LoRA), and parameter-efficient fine-tuning (PEFT) (14; 10; 15; 16; 17). Adapters are lightweight modules inserted into transformer layers, which allow fine-tuning by updating only a small portion of the model, while the rest of the model's parameters remain frozen (14). They may struggle to fully capture complex patterns or task-specific features, and they may not generalize well to different tasks without significant fine-tuning (6; 14; 18). Adapter techniques still add extra parameters to each layer, thereby increasing inference time. This could become inefficient in scenarios where model size is a critical constraint, such as in memory-limited edge devices (5). These have emerged as the most effective techniques for updating only a small subset of model parameters while keeping most weights frozen (19). These techniques have shown they can perform just as well as other methods while using a lot less computing power (13). Prompt-based learning is another area of research that has explored conditioning the model on task-specific prompts without modifying internal parameters (18). Furthermore, methods like prefix tuning and layer-wise fine-tuning are highly effective in reducing the number of trainable parameters. They focus on task-relevant layers or add small learnable vectors to the model input (5). Comparative studies prove that these approaches are not only efficient but also maintain or even enhance performance across a variety of NLP tasks, making them ideal for real-world applications where computational efficiency is critical (6).

LoRA is a popular method for fine-tuning large pre-trained models efficiently by introducing low-rank updates to the weight matrices. LoRA offers a compelling approach for fine-tuning large models efficiently, but it is not without trade-offs. The primary disadvantages include limited capacity for complex tasks, challenges in selecting the appropriate rank, potential performance degradation in certain layers, and added memory overhead from low-rank matrices (19). Moreover, its task-specific nature can limit transferability, making it less suitable for multi-task or highly dynamic applications (2). These drawbacks must be carefully considered, especially when applying LoRA in high-stakes, complex, or resource-constrained environments.

Adapters and LoRA assume that only a small portion of the model needs to be fine-tuned. This assumption may not hold for tasks that are highly complex, domain-specific, or significantly different from the pre-training data. In such cases, full fine-tuning of the entire model may still be necessary to achieve the best results

To solve these problems, we proposed a new Efficient Fine-Tuning method called TOU. Inspired by Shapeshifter using Low-rank Decomposition to reduce the size of trainable parameters. We proposed a method that factorized pretrained weight into the two smaller matrices, finetuning with new tasks, finally reconstruct to recover the structure of original model. The main contributions of this paper are summarized as follows:

(i) The proposed model for fine-tuning.

(ii) To illustrate the performance of our proposed method, a comprehensive evaluation is conducted on real dataset and various evaluation metrics.

The paper is organized as follows: Section 1 introduces the research motivation and related works. Section 2 explains the methodology. Section 3 discusses the experimental results. Section 4 concludes the work and points out future works.

2 TOU Fine-tuning method

2.1 Low-Rank Factorization with Truncated SVD

To enhance the efficiency and compactness of our model, we employ truncated singular value decomposition (truncated SVD) to factorize a 2D tensor (matrix) into two smaller matrices (20). Truncated SVD is a powerful technique for dimensionality reduction that retains only the most significant singular values and vectors, providing a high-quality approximation of the original matrix while significantly reducing its size (21). Truncated SVD is a powerful technique for dimensionality reduction that retains only the most significant singular values and vectors, providing a high-quality approximation of the original matrix while significantly reducing its size(20):

$$\mathcal{X} \approx U_k S_k V_k^T$$

Where:

- $U_k \in \mathbb{R}^{m \times k}$ contains only the top k left singular vectors.
- $S_k \in \mathbb{R}^{k \times k}$ contains only the top k left singular value.
- $V_k \in \mathbb{R}^{n \times k}$ contains only the top k right singular vectors.

2.2 Low-Rank Factorization and Re-factorization of Weight Matrices

Define a weight matrix as $W \in \mathbb{R}^{m \times n}$. The matrix W can be factored into low-rank decomposition $W = AB$, where $A \in \mathbb{R}^{m \times r}$, $B \in \mathbb{R}^{r \times n}$, and the rank $r \leq \min(m, n)$. The low-rank factorization is optioned by truncated SVD, where $A = U_k S_k$ and $B = V_k^T$. The root layer then is replaced by sequence with two smaller size layer whose weights are A and B. To minimize the error after factorization, we add small noise to two smaller weights as $A = A + \mathcal{N}\left(0, \alpha^2\right)$ and $B = B + \mathcal{N}\left(0, \alpha^2\right)$. This helps to prevent overfitting and improve generalization. During training, all layers, including low-rank factors, are updated. After fine-tuning, we restore the original structure of the model by replacing the sequence of updated A and B by a single layer with weight $W_r = AB$.

Algorithm 1 outlines the general process of TOU fine-tuning. It injects trainable rank-decomposition matrices into each layer then reconstructs. This workflow works without altering the original architecture's fundamental structure. This allows for easy implementation and compatibility with

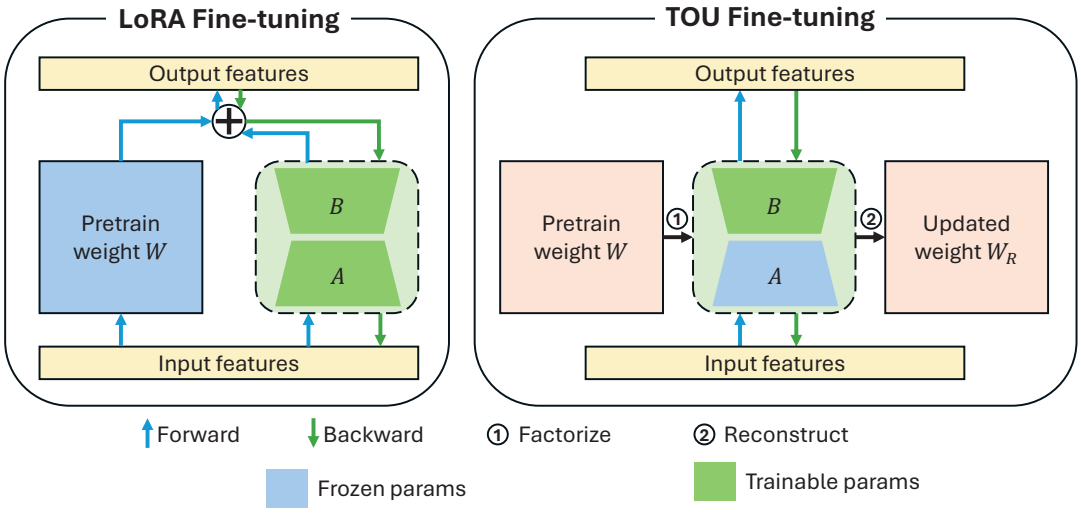

Figure 1: Compared to prior adaptation methods, LoRA, our approach is computationally efficient in the fine-tuning stage. More importantly, it does not suffer an accuracy loss because training after reconstruction is not required.

Algorithm 1 TOU fine-tuning algorithm

for $module$ in θ **do**
  $W \leftarrow module.weight$
  $A, B \leftarrow$ **truncatedSVD**(W, r)
  $layer_A \leftarrow Linearlayer(A + noise)$
  $layer_B \leftarrow Linearlayer(B + noise)$
  frozen$(layer_A)$
  $module \leftarrow sequence(layer_A, layer_B)$
end for
Fine-tune θ with D in 12 epochs
for $module$ in θ **do**
  $A \leftarrow module(layer_A).weight$
  $B \leftarrow module(layer_B).weight$
  $layer \leftarrow Linearlayer(AB)$
  $module \leftarrow layer$
end for
Return θ

a wide range of pre-trained models. TOU is on par with LoRA in maintaining the original model structure, inferior in the number of trainable parameters, but superior in inference speed. We illustrate our fine-tuning pipeline in Figure 1.

2.3 Objective to apply TOU for transformer model

A Transformer (22) is a stack of layers composed of two sub-layers: four weight matrices in multi-head self-attention (W_q, W_k, W_v, W_o) followed by a FFN sub-layer with two weight matrices $(\mathbf{fc}_a, \mathbf{fc}_b)$. As LoRA technique, we trick all W_q, W_k, W_v, W_o as single matrix of dimension $d_{model} \times d_{model}$. In LoRA, authors focused on only adapting the attention weights for downstream tasks and freezing the MLP modules (10). However, unlike LoRA, TOU does not restrict its modifications to the attention sub-layer. Instead, TOU can be applied to any layer in the Transformer model, including the FFN sub-layer. This flexibility allows for more comprehensive adaptation to downstream tasks.

By applying TOU to the entire Transformer model, we can potentially achieve better performance compared to LoRA, which only focuses on adapting the attention weights. This is because TOU can capture more complex relationships and interactions between different parts of the model.

3 Experiments

3.1 Experimental Setup

Foundation model and datasets . The experiments utilized fine-tuning a Vision Transformer (ViT) architecture on CIFAR100 dataset. We start from the ViT-B-16 pretrained on ImageNet dataset.

- Model: Vision Transformer (ViT) architecture, specifically ViT-B-16, pre-trained on ImageNet.
- Dataset: CIFAR-100.

Baseline . For the baseline methods, we compare with full weights fine-tuning and LoRA. Fine-tune is a common approach for adaptation. During fine-tuning, the model is initialized to the pre-trained weights and biases, and all model parameters undergo gradient updates. In the experiments, we use entire model fine-tuning. LoRA adds trainable pairs of rank decomposition matrices in parallel to existing weight matrices. For fair comparison, we set up two trajectories to apply LoRA just same as our proposed method. The number of trainable parameters is determined by the rank r and the shape of the original weights. The only different parameter setting is LoRA can use the smaller rank r than TOU.

Comparison matrices . We use number of trainable parameters, total of parameters, training time, accuracy and peak memory for evaluating the proposed work. Total parameters are the full

Table 1: Metrics of different efficient fine-tuning methods training on pretrained ViT for 10k time steps (approximately 12 epochs).

Methods	Trainable params (M)	Total params (M)	Accuracy (%)	Time	Memory usage (MiB)
FT	85.55	85.55	92.06	2h33m	11982
LoRA	0.59	86.47	89.46	1h13m	10976
Full TOU	24.83	48.72	91.09	53m	9752
TOU frozen FFN	11.51	48.72	90.40	52m	9408

training-time size of the model. Training time is measured as wall-clock time. Peak memory is measured using CUDA calls in PyTorch, which report the maximum amount of memorial located during fine-tuning.

Training details . All weights matrices are adapted for factorization tasks and frozen the first factorized matrix. We designed to adapt TOU in two trajectories to make a fair comparison with LoRA.

Trajectory 1. Adapt the full fine-tuning techniques with all layers in multi-head self-attention and MLP for downstream tasks.

Trajectory 2. Adapt the fine-tuning techniques with only layers in multi-head self-attention and freeze the FFN sub-layer.

We named these techniques correspond with trajectories as full TOU, and TOU frozen FFN from now. Following LoRA (10), we use a paged AdamW optimizer, and a batch size of 64 in the tuning period. We choose the constant learning rate schedule and set the learning rate to be 5×10^{-5} all LoRA and TOU fine-tuning experiments. The number of fine-tuning steps is 12 epochs. We use the same training configuration for all the experiments in this paper and avoid any hyperparameter screening to ensure a fair comparison. All experiments are conducted on NVIDIA GeForce RTX 4070Ti GPU.

3.2 Main results and efficiency

The comparison of the results for TOU and baseline methods is presented in Table 1. Across full fine-tuning or frozen FFN trajectory, the TOU consistently demonstrates outperformance to other methods in term of fine-tuning time.

Compared to full weight fine-tuning, TOU is memory efficient by significantly reducing the amount of trainable parameters (reduce 70%). Although TOU cannot achieve the same reduction ratio as LoRA. However, while LoRA can reduces 99.6% trainale parameters, TOU still dominates in terms of total number of parameters. On full weights fine-tuning, our full TOU achieves better training speed (66% speedup) and drops 0.97% accuracy.

On frozen FFN trajectory, our TOU achieves better accuracy (2.13% improvement) with comparable training time to LoRA. In terms of trainable parameters, our TOU could not reach the small number as LoRA, TOU frozen FFN reduced 87% trainble parameter compared to full model.

In terms of memory overhead, our TOU can reduce the overall memory cost by up to 27% and 16%, compared to full fine-tuning and LoRA, respectively. It means we can use a lower resource budget (e.g., cheaper GPU services with smaller memory size) to achieve the better fine-tuning performance compare with LoRA. In term of full fine-tuning, TOU is an effective method when full fine-tuning is necessary but there are constraints on available resources.

LoRA only works well when adapting LoRA with attention layers and frozen FFN. LoRA's effectiveness is limited to scenarios where only the attention layers are adapted and the FFN sub-layer is frozen. When adapted to the entire model, TOU outperforms LoRA regarding training speed and accuracy. This highlights the effectiveness of TOU, which solves the LoRA problem when LoRA assumes that only a small portion of the model needs to be fine-tuned. This assumption may not hold for highly complex, domain-specific, or significantly different tasks from the pre-training data. In such cases, full fine-tuning of the entire model may still be necessary to achieve the best results. When adapting LoRA and TOU for only multi-head and frozen FFN, TOU continues to yield better accuracy with similar training time. TOU is only slightly faster than LoRA by 6 minutes in this case.

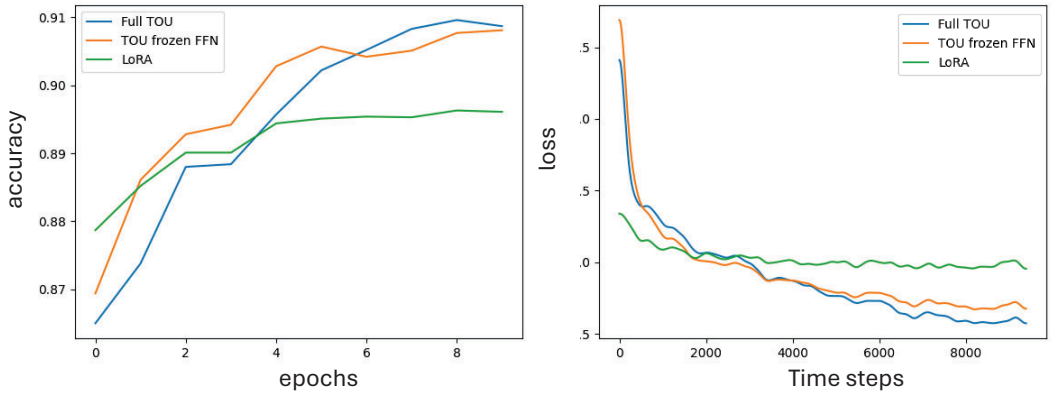

Figure 2: Train LoRA and full TOU, TOU frozen FFN for 12 epochs. The accuracy of vavidation were calculated every epochs.

Figure 2 shows that TOU consistently outperforms the ordinary LoRA method in the early stages of training. As depicted in Figure 2, TOU frozen FFN achieves considerably lower loss than LoRA at time step 2k, and higher accuracy from epoch 2 performance. Additionally, the figure shows a loss function converge better in TOU's performance. When increasing the number of epochs in training, we observed that full TOU tends to have lower training losses and higher accuracy compared to TOU frozen FFN towards the end approximately 5 epoch's completion time frame.

4 Conclusion and future works

We proposed TOU, a novel approach for efficient full-weight fine-tuning that leverages low-rank factorization using truncated SVD. TOU is straightforward to implement, applied across different foundation models, and highly efficient in both full-weight fine-tuning and small set-weight updates. Our experimental results demonstrate that TOU outperforms LoRA and traditional fine-tuning in terms of speed, memory efficiency, and overall performance.

To further enhance TOU, we propose two promising areas for future exploration: (i) Adaptive Rank Selection for factorized matrices - TOU directly replaces the pre-trained matrix with smaller matrices, which means that the information from the original weight is lost. To address this, we suggest exploring element-wise factorization and adaptive rank selection. By factorizing each element of the weight matrix individually, we can more precisely determine the optimal rank for each layer. This approach allows for a more flexible and efficient representation of the model; (ii) Combining with Other Efficient Methods - TOU can be combined with other efficient fine-tuning methods to achieve even greater performance gains. For example, we could combine quantization techniques or optimize the training process to further reduce the number of trainable parameters.

References

[1] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, 2019.

[2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 2020-December, 2020.

[3] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," in *EMNLP 2021 - 2021 Conference on Empirical Methods in Natural Language Processing, Proceedings*, 2021.

[4] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, 2020.

[5] M. Mosbach, M. Andriushchenko, and D. Klakow, "On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines," in *ICLR 2021 - 9th International Conference on Learning Representations*, 2021.

[6] N. Houlsby, A. Giurgiu, S. Jastrzçbski, B. Morrone, Q. de Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter-efficient transfer learning for nlp," in *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, 2019.

[7] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, vol. 1, 2018.

[8] Z. Li, E. Wallace, S. Shen, K. Lin, K. Keutzer, D. Klein, and J. E. Gonzalez, "Train large, then compress: Rethinking model size for efficient training and inference of transformers," in *37th International Conference on Machine Learning, ICML 2020*, vol. PartF168147-8, 2020.

[9] E. Ben-Zaken, S. Ravfogel, and Y. Goldberg, "Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, vol. 2, 2022.

[10] E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," in *ICLR 2022 - 10th International Conference on Learning Representations*, 2022.

[11] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, 2018.

[12] Y. C. Hsu, T. Hua, S. E. Chang, Q. Lou, Y. Shen, and H. Jin, "Language model compression with weighted low-rank factorization," in *ICLR 2022 - 10th International Conference on Learning Representations*, 2022.

[13] A. Panahi, S. Saeedi, and T. Arodz, "Shapeshifter: A parameter-efficient transformer using factorized reshaped matrices," in *Advances in Neural Information Processing Systems*, vol. 2, 2021.

[14] J. Pfeiffer, A. Kamath, A. Rücklé, K. Cho, and I. Gurevych, "Adapterfusion: Non-destructive task composition for transfer learning," in *EACL 2021 - 16th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference*, 2021.

[15] J. He, C. Zhou, X. Ma, T. Berg-Kirkpatrick, and G. Neubig, "Towards a unified view of parameter-efficient transfer learning," in *ICLR 2022 - 10th International Conference on Learning Representations*, 2022.

[16] M. Valipour, M. Rezagholizadeh, I. Kobyzev, and A. Ghodsi, "Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation," *ArXiv*, vol. abs/2210.07558, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252907428

[17] Y. Xu, L. Xie, X. Gu, X. Chen, H. Chang, H. Zhang, Z. Chen, X. ZHANG, and Q. Tian, "QA-loRA: Quantization-aware low-rank adaptation of large language models," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=WvFoJccpo8

[18] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *ACL-IJCNLP 2021 - 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 2021.

[19] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[20] P. C. Hansen, "The truncated svd as a method for regularization," *BIT*, vol. 27, 1987.

[21] A. Falini, "A review on the selection criteria for the truncated svd in data science applications," *Journal of Computational Mathematics and Data Science*, vol. 5, 2022.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 2017-December, 2017.