DeLoRA: Decomposition Based Low Rank Adaption *

WashU CSE 527A Final Project Report

Ravi Dantuluri
Washington University in St. Louis
d.ravi@domain

Nicholas Sullivan
Washington University in St. Louis
d.sullivan@wustl.edu

Pascal Yang
Washington University in St. Louis
pascal@wustl.edu

Abstract

In this paper, we explore novel parameter-efficient tuning methods built upon the state-of-the-art method low-rank adaption (LoRA) proposed by (Hu et al., 2021). Inspired by singular value decomposition, we freeze the two low-rank adaption matrices in the LoRA method and insert a smaller trainable matrix in the middle. We name our method Decomposition Based Low Rank Adaption (DeLoRA). This method significantly reduces the number of trainable parameters while maintaining decent performance when fine-tuning RoBERTa (Liu et al., 2019) on General Language Understanding Evaluation (GLUE) (Wang et al., 2019). For example, on SST-2 and CoLA (Wang et al., 2019), DeLoRA yields comparable results to LoRA with 88% to 67% less trainable parameters. Additionally, in the DeLoRA method, the traditional LoRA weights are frozen, allowing DeLoRA to be used as light-weight subsequent adaption of LoRA module pre-trained in one task for similar tasks.

1 Introduction

Fine-tuning, in the context of large language models, is the act of adapting a pre-trained language model to specific tasks (Ding et al., 2022), which entails training pre-trained models with task-specific data to produce new instances. Such a practice is crucial to the current development of language models because it allows a single pre-trained model to adapt to multiple downstream tasks with relatively small-scale tuning data compared to the pre-train corpus. However, with the exponential growth in the parameter size of large language models (LLMs), fine-tuning has become impractical for those with fewer computing resources. For example, the widely used trained model LLaMA2-70B (Touvron et al., 2023) requires 37.6 GB of VRAM (Ding et al., 2022) during a full parameter tuning. Besides VRAM consumption, the size of a fine-tuned model also makes sharing and storage inconvenient.

Parameter-efficient fine-tuning methods attempt to solve the above problems by trying to reduce the number of trainable parameters of large language models. This mitigates the resource constraints while maintaining performance. The current state-of-the-art method, LoRA(Hu et al., 2021), decomposes the weight matrix updates into pairs of low-rank matrices. This method significantly reduces the number of trainable parameters during fine-tuning and has been shown to perform at equal to or better than full fine-tuning.

Our method, decomposition-based LoRA (DeLoRA), takes the LoRA method one step further by freezing the two low-rank matrices (initialized randomly) and introducing a trainable Σ matrix in the middle. The approach allows us to further scale down the number of trainable parameters because the dimensions of Σ are completely independent of the hidden state dimension of the model, as opposed to LoRA's low-rank matrices. At the same time, DeLoRA inherits many advantages of LoRA, including an unchanged model structure and no additional inference latency. By doing so, our contribution to the field is a technique that allows for minimal loss in accuracy in exchange for a reduction of parameters up to almost 66% from LoRA.

2 Related Work

2.1 Parameter-efficient Fine-tuning

Fine-tuning of a large language model takes a pre-trained model on general corpus and retrains the model onto specific downstream tasks(Devlin et al., 2019). Parameter-efficient fine-tuning optimizes re-training by reducing the number of trainable parameters, thus reducing memory usage and speeding up the process. This is a crucial topic as training larger models generally results in better per-

*The LaTeX template is adopted from ACL style and formatting.

formance and remains an active research direction in the natural language processing field (Hu et al., 2021). Many works have introduced various efficient tuning methods, including training only the bias terms (BitFit by Zaken et al. (2022)), inserting additional trainable layers (Adapter by Houlsby et al. (2019)), and training low-rank approximations (LoRA by Hu et al. (2021)).

2.2 Low-Rank Structure of Deep Learning

A lot of parameter-efficient fine-tuning methods rely on the hypothesis that the fine-tuning of pre-trained large language models (PLMs) towards downstream tasks has a low intrinsic dimension. The term was originally used to describe the number of parameters needed in a minimal representation of the data.

In the context of parameter-efficient fine-tuning, it refers to the minimum dimensions of the subspace needed to train a neural network on specific tasks (Li et al., 2018). Using a randomly generated matrix for projection, Li et al. (2018) found that the intrinsic dimension of many neural networks is very small compared to the original parameter space. For example, 90% of performance can be achieved with 750 parameters from FC network on MNIST task. Inspired by the findings, Aghajanyan et al. (2020) measured and confirmed the low intrinsic dimensions of fine-tuning PLMs on common NLP tasks.

2.3 Low-Rank Adaption (LoRA)

Inspired by Aghajanyan et al. (2021), Hu et al. (2021) proposed LoRA, the use of low-rank decomposition as proxy parameters to update attention weights in transformer models. LoRA has been shown to perform closely or even better than full-parameter tuning on various transformer-based models, including RoBERTa, GPT-2, and GPT-3. For example, LoRA can outperform full-parameter tuning with rank 4 ($r = 4$) when tuning GPT-3 with 175B parameters on MultiNLI-matched. And rank 1 is sufficient to adapt GPT-2 to the same task. At the same time, since the updated weights are merged into the model weights during inference time, LoRA does not change the model architecture and introduces no latency to the model.

Our method is directly developed and built upon LoRA. We freeze the two low-rank matrices in LoRA and insert a trainable matrix in the middle. In this way, we can further scale down the number of trainable parameters while inheriting LoRA's advantage in unchanged model architecture and unaffected latency.

2.4 VeRA

VeRA introduced by, Kopiczko et al. (2023), presents a method similar to LoRA and to DeLoRA. The central idea of VeRA is the training of two scaling (and therefore diagonal matrices) which proceed each low-rank matrix in LoRA. These matrices scale each of the low-rank randomized matrices.

VeRA has been shown to perform closely or even better than LoRA and for other fine-tuning methods with a fraction of the parameters. Notably, VeRA outperforms LoRA on the E2E data set using GPT-2 as the base model model. Importantly, like LoRA, it does not change the model architecture and introduces no latency to the model.

While our focus is on introducing DeLoRA as a novel approach built upon LoRA, this VeRA discussion highlights the broader landscape of parameter-efficient fine-tuning methods. VeRA's success reinforces the efficacy of strategies involving low-rank approximations with transforming matrices. This further validates the concept behind DeLoRA. In future work, more detailed comparisons between DeLoRA, VeRA and LoRA could provide an understanding of low-rank matrices approximation of the change in weight matrices. This would allow for a broader view of parameter-efficient fine-tuning techniques and would assist researchers in selecting the most suitable method based on their constraints and requirements. In summary, VeRA adds a tool for parameter-efficient fine-tuning, contributing to a more comprehensive approach to model adaptation.

3 Approach

Our approach uses the concept of Singular Value Decomposition (SVD) in combination with LoRA (Hu et al., 2021). In the LoRA paper, the authors represent the fine-tuning process as $W' = W + \Delta W$. Instead of calculating ΔW directly, they approximate it. They do this with the previously mentioned low-rank matrices A and B. Given that W is a $d \times d$ matrix, they set A to be size $d \times r$ and B to be $r \times d$. With a small enough r, this approximation requires few total parameters which need to be trained.

The inspiration behind the novel aspect of DeLoRA comes from SVD. SVD is a method of de-

composing a matrix into three other matrices $M = U\Sigma V$. Instead of only decomposing $\Delta W = A\ B$, we freeze matrix A and matrix B and insert a matrix between the two. The decomposition becomes $\Delta W = A\Sigma B$.

Because A and B are random matrices, we can train Σ to appropriately transform the randomness from the two matrices to achieve an adequate approximation of ΔW. Simply transforming A and B takes fewer parameters than training all of A and all of B. To back up this claim, we tried to approximate a randomized low-rank matrix using our method. That is, we randomized three matrices, calling them A, B and C. We then tried to approximate C with $A\Sigma B$, only updating Σ. We found that with high enough dimensions of C (around 100×100,) we could almost completely approximate C

We initialize the elements of A and B to be iid normally distributed, and then we freeze them, leaving only the Σ matrix to be trained. Our first approach makes Σ sized $r \times r$ of trainable parameters, leaving us with r^2 parameters compared to LoRA's $2dr$. (r is much smaller than d). To do this, we modified the code from LoRA (https://github.com/microsoft/LoRA). We are comparing our methods to LoRA and other fine-tuning methods. The other approach is to make the Σ matrix a diagonal. This is due to the underlying theory of SVD. The Σ matrix scales the other two matrices, U and V, which do the "stretching" and "rotating". Because we believe that ΔW has low rank (Hu et al., 2021), by just learning to scale the "stretches" and "rotations," we hope to learn the ΔW matrix; however, this requires future work.

4 Experiments

We tested our method by adapting RoBERTa base (Liu et al., 2019) on the general language understanding (GLUE) benchmark (Wang et al., 2019). We chose this combination to yield a clear comparison with the results from Bitfit (Zaken et al., 2022), Adapter (Houlsby et al., 2019), and LoRA (Hu et al., 2021), which included a similar setup.

4.1 Data

The GLUE benchmark (Wang et al., 2019), short for the General Language Understanding Evaluation benchmark, is a collection of diverse natural language understanding tasks designed to evaluate and analyze the performance of machine learning models on various aspects of language understand-

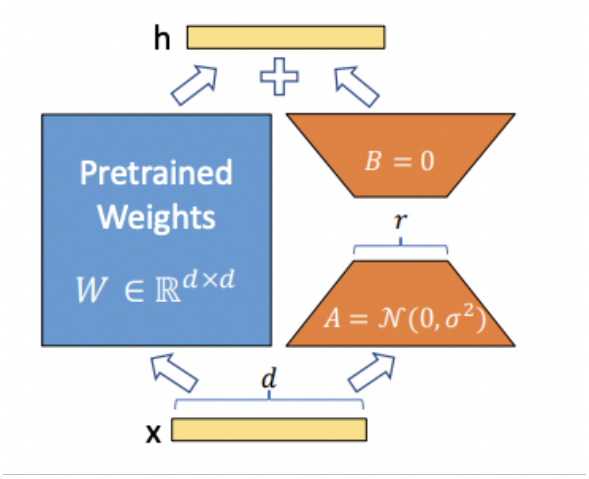

Figure 1: LoRA visualized (Hu et al., 2021). The method freeze the original weights and updates only the low-rank approximation of the original matrix. B is initialized to 0 and A to a normal distribution.

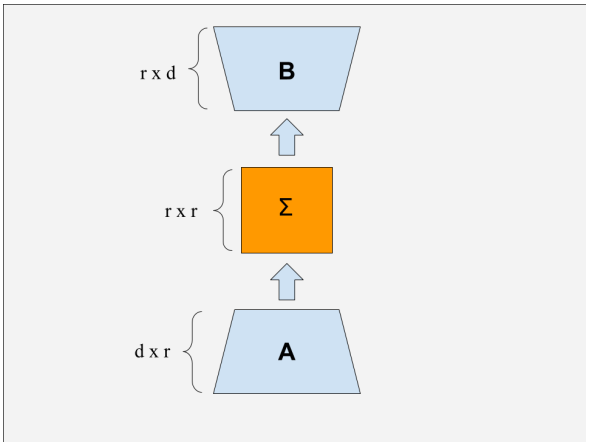

Figure 2: DeLoRA visualized. We freeze A and B and insert a trainable Σ matrix in the middle. A and B are initilized similar to how LoRA initialize A.

ing. GLUE comprises a diverse set of tasks ranging from textual entailment (MNLI) and sentiment analysis (SST-2) to question similarity (QQP).

4.2 Model

In our study, we have selected RoBERTa-base (Liu et al., 2019) as the primary model for evaluating our method. The model is an optimized version of the groundbreaking BERT model (Devlin et al., 2019), which has revolutionized the field of natural language processing. While RoBERTa-base has been outperformed by many other larger models in benchmarks such as GLUE, it is still a competitive model in comparable size. We utilize the pre-trained checkpoint of RoBERTa-base from HuggingFace Transformers library (Wolf et al., 2020).

To implement DeLoRA, we build upon the open source code from Hu et al. (2021), which provides code that re-models RoBERTa-base's attention matrices with LoRA blocks. To implement DeLoRA, we added an additional trainable layer Σ between the original pair of low-rank matrices. During training time, we freeze the original model and the low-rank matrices(randomly initialized), setting only Σ blocks to be trainable.

4.3 Evaluation Method

We report the evaluation performance of DeLoRA tuned RoBERTa-base on the GLUE benchmark. Note that for MRPC, STS-B, and RTE tasks, Liu et al. (2019) mentioned that RoBERTa benefits from first training on MNLI task. Therefore, in our experiment, we first train the model on MNLI task and further train the checkpoint onto MRPC, STS-B, and RTE. This is also the practice adopted by other methods we are comparing to.

We also train the MNLI-LoRA checkpoint provided by Hu et al. (2021) onto the MRPC, STS-B, and RTE datasets. We do so by freezing the provided LoRA weight matrices and train an additional Σ layer in the middle. In this way, we test DeLoRA's ability to adapt pre-trained LoRA weights to new tasks. This method could potentially be used by very resource-limited researchers.

4.4 Experimental Details

We followed the same hyper-parameter setting as the original LoRA paper, except for the followings. **1) lora_r** defines the dimension of LoRA and DeLoRA modules. We experiment with different ranks from 8 to 64 because LoRA did the same. **2) lora_alpha** defines the weight of the update to the original model following the formula $weight = \frac{lora\_alpha}{lora\_r}$. We set this to be two times the current lora_r, resulting in a weight of 2 that is similar to the experiment setup of Hu et al. (2021).

Method	# Trainable Parameters	CoLA	SST-2	QQP	QNLI	MNLI	MRPC	RTE	STS-B
Full-FT*	125M	63.6	94.8	91.9	92.8	87.6	90.2	78.7	91.2
BitFit*	0.1M	62.0	93.7	84.0	91.8	84.7	92.7	81.5	90.8
Adpt*	0.3M	60.8	94.2	90.2	93.1	87.1	88.5	71.5	89.7
Adpt*	0.9M	62.6	94.7	90.6	93.0	87.3	88.4	75.9	90.3
LoRA*	0.3M	63.4	95.1	90.8	93.3	87.5	89.7	86.6	91.5
DeLoRA	< 0.1M	62.1	93.2	88.1	90.2	84.6	88.2	77.6	90.0
DeLoRA Rank (r)		64	32	64	64	64	64	64	64
DeLoRA # Trainable Parameters		100k	25k	100k	100k	100k	100k	100k	100k

Table 1: Comparing RoBERTA_base adapted with different methods. We report the overall accuracy for MNLI, Matthew's correlation for CoLA, Pearson correlation for STS-B, and accuracy for other tasks. * indicates numbers reported from the original papers.

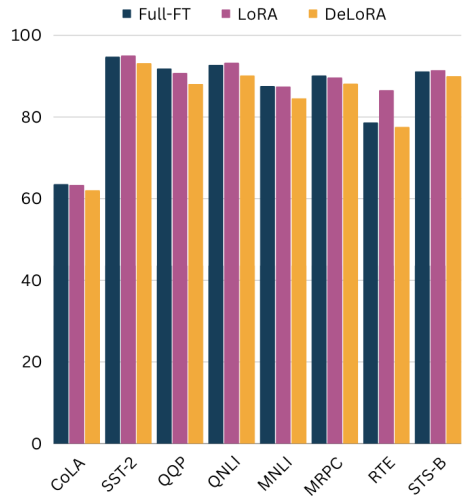

Figure 3: Comparison Visualized

5 Results

We tested the DeLoRA's performances with Σ being a normal matrix and a diagonal matrix. For the latter, we failed to yield comparable results even using the same dimension as the original model. Further work is needed to discover the reason behind this since VeRA performed so well. Therefore, we only report results that set Σ to be a normal matrix and omit the results where Σ is a diagonal matrix.

5.1 Overall Results

Our results in Table 1 show that DeLoRA can significantly reduce the number of parameters while maintaining decent performance on natural language understanding tasks. For example, the

DeLoRA Rank	# Trainable Parameters	CoLA	SST-2	QQP	QNLI	MNLI
16	1.5k	47.2	92.2	84.3	86.9	79.4
32	25k	54.4	**93.2**	86.5	88.2	82.5
64	100k	**62.1**	92.4	**88.1**	**90.2**	**84.6**

Table 2: Comparing DeLoRA's performance on RoBERTa-base with different ranks.

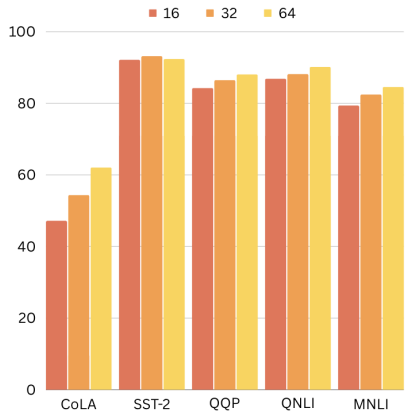

Figure 4: Rank comparison visualized

adapter requires 900 thousand parameters which, compared to DeLoRA, is over an 800% increase. Additionally, in tests such as MNLI, the difference in accuracy is only .5%. Hence, for those with extremely limited computational resources, DeLoRA provides an opportunity to access a very powerful tool. We achieved similar performance to vanilla fine-tuning and outperformed or achieved similar results compared to the BitFit and Adapter with fewer trainable parameters. Though DeLoRA cannot consistently perform better than any of the existing approaches, the results demonstrate the feasibility of the method and promote further investigation into the topic. Considering the significant reduction in trainable parameters required, the small losses in accuracy shown in Table 1 exceeded our expectations because we were able to reduce the number of trainable parameters by as much as almost 67%. Typically, from such a huge loss in the number of parameters, most models would notice huge impacts on their viability. Keeping this in mind, further work should be dedicated to increasing the number of parameters to compare the efficiency of each parameter between the methods.

5.2 Choice of Rank

We compare DeLoRA's performance under different ranks. We report the results in Table 2. In most of the tasks, DeLoRA's performance improves with a higher rank. This can be noticed across the CoLA, SST-2, QQP, QNLI, and MNLI datsets. However, in the case of SST-2, using a rank of 64 proved to be less optimal than having a rank of 32 but was still better than having a rank of 16. This corresponds to the general expectation that more trainable parameters usually yield better results. Further investigation could be done to test the bottleneck of DeLoRA's rank. Additionally, further work investigating higher-rank DeLoRA may be enlightening as it could elucidate the balance between rank selection and model performance.

Method	MRPC	RTE	STS-B
Classifier only	71.1	56.4	62.8
Classifier + DeLoRA	77.5	78.7	86.4

Table 3: Testing DeLoRA's ability to adapt pre-trained LoRA weight to new tasks. The Classifier-only version (baseline) adopt pre-trained LoRA weights on MNLI data set and set only the classifier layers trainable. The Classifier-DeLoRA version inserts an additional trainable delora block to the given LoRA weights on the basis of the Classifier-only version.

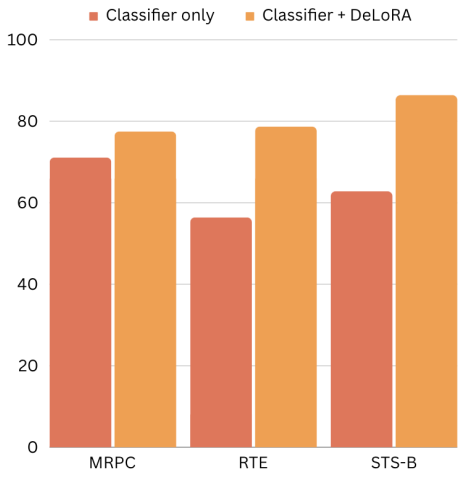

Figure 5: Classifier visualized

5.3 Subsequent Training of LoRA

We tested DeLoRA's potential to be the "LoRA of LoRA," which enables the lightweight adaption of pre-trained LoRA weights to new tasks. Table 3 shows our results. We start from the MNLI-LoRA checkpoint provided by Hu et al. (2021) and train the model's classifier head with and without the DeLoRA block. The performance is significantly better when DeLoRA is added to the training process. And DeLoRA only introduces 1.5k more

trainable parameters given that the LoRA rank is 8. Compared to the size of the classifier layer (0.5M) the additional parameters is almost negligible. This leads to an obvious potential for further work trying to use DeLoRA (or LoRA, for that matter) to train the classifier layer. Additionally, it begs the question: what is the performance of only a fully fine-tuning classifier?

6 Analysis

Our results have shown that DeLoRA is a feasible parameter-efficient tuning method. Though the method does not consistently outperform any of our baseline, it still achieves adequate performance in most of the tasks. The defect in performance can be explained partly by the number of trainable parameters in our experiment. The highest amount is 100k, which is far from the 300k and 900k in LoRA and Adapter, making DeLoRA an accessible model to those with fewer powerful computational resources. Further work can explore the performance of DeLoRA with a similar number of parameters.

An exciting finding of our experiment is DeLoRA's ability to adapt LoRA. We show that with just 1.5k additional parameters, DeLoRA can improve the performance of LoRA weights trained on MNLI by roughly 20+% on RTE and STS-B. This implies that DeLoRA holds the potential as a lower-rank tuning of a pre-trained LoRA model, which could be useful in use cases when complete LoRA training is not feasible.

This paper encourages further investigation into refine rank selection strategies and leveraging DeLoRA for additional for tasks that have already been fine-tuned using LoRA. In summary, DeLoRA is an effective tool to do parameter-efficient fine-tuning, allowing resource-friendly fine-tuning of large language models.

7 Conclusion

In summary, the introduction of Decomposition Based Low-Rank Adaption (DeLoRA) has shown to be an efficient fine-tuning method. Drawing inspiration from singular value decomposition (SVD) and combining it with the cutting-edge LoRA methodology, DeLoRA harnesses substantially reduce the number of trainable parameters while preserving performance. This is evident by our experiments with RoBERTa on the GLUE benchmark.

DeLoRA's original approach freezes the low-rank matrices in LoRA while inserting a trainable Σ matrix between the two matrices.

DeLoRA exhibits that it is possible to achieve comparable performance while significantly reducing the number of trainable parameters to less than 100,000. For example, on MRPC and STS-B, DeLoRA achieved an accuracy of 88.2 and 90.0, whereas the LoRA version achieved an accuracy of 89.7 and 91.5, respectively.

However, in other areas, it did not perform so well, such as with QQP where DeLoRA's accuracy was 88.1 and LoRA's was 90.8. One thing to note is that during times of greater disparities, such as with SST-2, DeLoRA is trained with a lower rank, and naturally, a lower rank translates to fewer parameters. Hence, there is future work to be done to find a balance between lower parameters and loss in accuracy. For example, a rank of 32 with DeLoRA proved to be most optimal for accuracy with SST-2 and not 64 (Table 2).

Our experiments across the GLUE benchmark affirm the validity of DeLoRA across a variety of tasks. While DeLoRA's performance may not uniformly outshine the other existing methodologies such as BitFit, Adapter, and the original LoRA, it consistently delivers competitive results with a fraction of the parameters.

In conclusion, DeLoRA and subsequent work are promising avenues for parameter-efficient fine-tuning by offering a more accessible and resource-friendly way to fine-tune increasingly large LLMs. DeLoRA adds a tool to the researcher's toolbox, allowing more flexibility in the parameter efficient fine-tuning and giving researchers more control over the total number of parameters fine-tuned.

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2020. Intrinsic dimensionality explains the effectiveness of language model fine-tuning.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*. Association for Computational Linguistics, Minneapolis, Minnesota, pages 4171–4186. https://doi.org/10.18653/v1/N19-1423.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong

Sun. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models.

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. 2021. Lora: Low-rank adaptation of large language models.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki Markus Asano. 2023. Vera: Vector-based random matrix adaptation.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes.

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.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

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