Language Models are Super Mario: Absorbing Abilities from Homologous Models as a Free Lunch

Le Yu, Bowen Yu,* Haiyang Yu, Fei Huang, Yongbin Li*
Alibaba Group
{chuanyi.yl, yubowen.ybw, yifei.yhy, f.huang, shuide.lyb}@alibaba-inc.com

Abstract

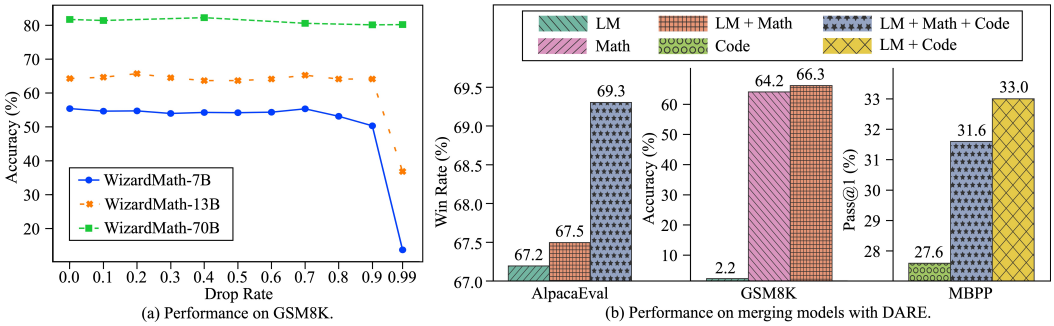
Humans have harbored a longstanding desire to acquire additional abilities through absorption. Super Mario serves as an embodiment of this human dream, which can collect items to gain extra skills such as throwing fireballs and being temporarily invincible. In this paper, we uncover that Language Models (LMs), either encoder- or decoder-based, can obtain new capabilities by assimilating the parameters of homologous models without the need for retraining or GPUs. Typically, new abilities of LMs can be imparted by Supervised Fine-Tuning (SFT), reflected in the disparity between fine-tuned and pre-trained parameters (i.e., delta parameters). We initially observe that by introducing a novel operation called DARE (**D**rop **A**nd **RE**scale), most of the delta parameters can be directly set to zeros without affecting the capabilities of SFT LMs and larger models can tolerate a higher proportion of discarded parameters. Based on this observation, we further sparsify delta parameters of multiple SFT homologous models with DARE and subsequently merge them into a single model by parameter averaging. We conduct experiments on eight datasets from the GLUE benchmark with BERT and RoBERTa. We also merge WizardLM, WizardMath, and Code Alpaca based on Llama 2. Experimental results show that: (1) The delta parameter value ranges for SFT models are typically small, often within 0.005, and DARE can eliminate 99% of them effortlessly. However, once the models are continuously pre-trained, the value ranges can grow to around 0.03, making DARE impractical. We have also tried to remove fine-tuned instead of delta parameters and find that a 10% reduction can lead to drastically decreased performance (even to 0.0). This highlights that SFT merely stimulates the abilities via delta parameters rather than injecting new abilities into LMs; (2) DARE can merge multiple task-specific LMs into one LM with diverse abilities. For instance, the merger of WizardLM and WizardMath increases the GSM8K zero-shot accuracy of WizardLM from 2.2 to 66.3, retaining its instruction-following ability while surpassing WizardMath's original 64.2 performance. All resources are available at `https://github.com/yule-BUAA/MergeLM`.

1 Introduction

Human beings have always expressed their ambition to acquire additional abilities through various ways such as movies and games. For example, in X-Men's Apocalypse, the character can absorb the powers of other mutants to strengthen himself. Likewise, the protagonist in the Super Mario games can gain superpowers like throwing fireballs by absorbing in-game items. Large Language Models (LLMs), such as GPT-4 [45], can reasonably be considered as early iterations of artificial general intelligence systems, given their performance is remarkably close to human-level capabilities. In this

*Corresponding author.

Preprint. Under review.



(a) Performance on GSM8K.

(b) Performance on merging models with DARE.

Figure 1: In (a), DARE can effectively eliminate 90% or even 99% delta parameters of WizardMath without sacrificing much performance on GSM8K. In (b), LM, MATH, and Code are abbreviations of WizardLM-13B, WizardMath-13B, and llama-2-13b-code-alpaca. DARE can merge several task-specific SFT language models into a single model, which supports the utilities of all SFT models.

paper, we astonishingly find that LMs, similar to Apocalypse and Super Mario, can enhance their capabilities by absorbing other models without the need for training or GPUs.

Formally, Supervised Fine-Tuning (SFT) is the most widely adopted strategy for assigning task-specific capabilities to LMs by optimizing their parameters [13, 67]. The effectiveness of SFT is fully evident in the alteration of the model parameters before and after SFT, referred to as *delta parameters* [12]. We initially demonstrate that SFT LM (either encoder- or decoder-based) always tends to acquire *excessively redundant delta parameters*. To be specific, we present DARE, which randomly resets some delta parameters to zeros based on a drop rate p and subsequently scales the remaining parameters by a factor of $1/(1-p)$. Despite its simplicity, with the assistance of DARE, when the LM model parameters reach 70 billion, we can eliminate up to 99% delta parameters with minimal impact on model performance (see Figure 1(a)). The more parameters the LM has, the larger p it can tolerate. This discovery suggests that SFT LM indeed learns a multitude of low-rank structures akin to LoRA [25]. Thus, even when most of these structures are removed, resulting in a low-rank and extremely sparse delta parameter set, the LM can still retain its capabilities.

Based on this observation, we can confidently merge multiple homologous SFT LMs (pre-trained from the same backbone) without significant concerns about the decrease in their capabilities. As long as a small portion of the delta parameters remains unaffected in the merging process, the abilities of LMs unlocked by SFT can still be preserved. We first employ DARE to eliminate redundant delta parameters in each model before merging, which can potentially mitigate the interference of parameters among multiple models [62]. Then, we apply established model merging techniques [59, 26, 44, 27, 62] to the parameters with reduced redundancy to create a single model with diverse capabilities. We conduct extensive experiments on encoder-based LMs on eight datasets from the GLUE benchmark, and decoder-based Llama 2 with three distinct abilities: instruction-following, mathematical reasoning, and code-generating. We observe that:

(1) SFT LMs exhibit a substantial number of redundant delta parameters whether they are based on BERT, RoBERTa, or Llama 2. DARE allows the removal of approximately 90% or even 99% delta parameters without significantly affecting the performance of downstream tasks. The rescale operation in DARE is a crucial component to guarantee effective ablations of delta parameters. Without rescaling, removing only 10% delta parameters would noticeably affect performance. We attribute this phenomenon to the fact that rescaling helps preserve the connectivity of model parameters [46].

(2) DARE is able to enhance the performance of most existing model merging methods when merging encoder-based LMs on the eight datasets from GLUE. When it comes to larger LMs based on Llama 2, the simple parameter averaging method can already produce surprisingly good results. As shown in Figure 1(b), we merge WizardLM and WizardMath by combining DARE and parameter averaging, leading to a significant improvement of WizardLM's mathematical reasoning ability from 2.2 to 64.2 accuracy on GSM8K, while also modestly enhancing its instruction-following ability with win rate from 67.2 to 67.5 on AlpacaEval. It is worth noticing that all these benefits are achieved by solely using CPUs without further training. Similar improvements can also be observed when merging code-generating models.

(3) DARE is applicable to SFT delta parameters whose value ranges are relatively small. Different from the observations of delta parameters, dropping only 10% fine-tuned parameters would lead to a catastrophic decrease in performance, even approaching zero. We also find that the delta parameters of SFT LMs usually stay within a range of 0.005 or less, indicating minimal modifications to the pre-trained LM. However, once we continue pre-training, the delta parameters can rapidly reach around 0.03, making DARE infeasible. This further confirms that SFT primarily unlocks the abilities of the pre-trained LM, rather than introducing additional abilities.

Last but not least, we have implemented an open-sourced codebase at `https://github.com/yule-BUAA/MergeLM`, which integrates existing popular model merging methods and supports both encoder- and decoder-based language models. We hope this work can advance the understanding of how alignment works from the perspective of parameters.

2 Related Work

Supervised Fine-tuning of Language Models. Supervised fine-tuning of language models aims to impart pre-trained LMs with particular abilities by optimizing them on task-specific labeled data, which has become the de facto standard paradigm in natural language processing [13, 67]. Full Fine-Tuning (FFT) is a conventional category of SFT [47, 11], which fully adjusts all the parameters of LMs but will become prohibitively expensive with the increasing size of LMs [43, 21]. To tackle this issue, Parameter-Efficient Fine-Tuning (PEFT) has been proposed, which only updates a small number of parameters (either inherently in LMs or extra introduced) while freezing the remaining major parameters [24, 38, 35, 32, 25]. Indeed, the effects of SFT (whether FFT or PEFT) can be ultimately captured by the difference between parameters of LMs before and after SFT, i.e., delta parameters. In this paper, we conduct a thorough analysis of the delta parameters of various SFT LMs and find they are *extremely redundant*. We propose an innovative approach to achieve competitive performance with standard SFT LMs by removing most (90% or even 99%) delta parameters.

Network Pruning Technique. With the rapidly increasing size of neural networks, network pruning technique has been widely applied to reduce the computational costs [7, 37]. The objective of network pruning is to eliminate unnecessary model parameters whiling maintaining the performance as much as possible [68, 40, 16, 17, 60]. Magnitude-based pruning is one classical pruning method, which selects parameters according to their magnitudes (i.e., absolute parameter values) [20, 34, 31]. To be specific, parameters with magnitudes lower than a certain threshold are removed, and the remaining parameters are preserved. In fact, the proposed DARE is somewhat related to the concept of network pruning since DARE can also drop parameters. But it differs from existing pruning methods in two main aspects: 1) DARE focuses on delta parameters while most pruning methods operate on the fine-tuned parameters; 2) DARE is data-free and training-free without the requirements of any extra data or (iterative) retraining, which are often needed by previous pruning methods.

Model Merging. Model merging has become a trending research direction in recent years, which aims to merge multiple task-specific models into a single model with diverse abilities [59, 44, 26, 27, 62, 65]. The superiority of model merging over multi-task learning [9, 66] (which also intends to obtain one model with several abilities) is that model merging usually pays attention to the combination of model parameters without accessing the original training data [44, 27]. Average Merging [59] is one common model merging approach, which utilizes averaged parameters to construct the merged model. Task Arithmetic [26] employs a pre-defined scaling term to distinguish the importance of various models. Fisher Merging [44] performs automatic weighted merging of parameters, where the weights are calculated by the Fisher information matrix [15]. RegMean [27] masterly solves model merging by optimizing a linear regression problem with closed-form solutions. TIES-Merging [62] tackles the task conflicts in [26] by trimming low-magnitude parameters, resolving sign disagreements, and disjointly merging parameters with consistent signs. In this paper, we employ DARE as a general preprocessing technique for existing model merging methods by first sparsifying delta parameters of several SFT homologous models and then merging them into a single model, which is equipped with the abilities of all SFT models.

Federated Learning. In federated learning, multiple data owners collaboratively train a model in an iterative process with the guarantee of data privacy [63, 64]. The model merging technique naturally satisfies data privacy requirements and can be treated as a single iteration in federated learning, which wishes to effectively integrate the messages of data owners (usually denoted as model updates

in practice). Since our DARE can facilitate the model merging task, we believe it can serve as a promising solution for the federated learning field.

3 Methodology

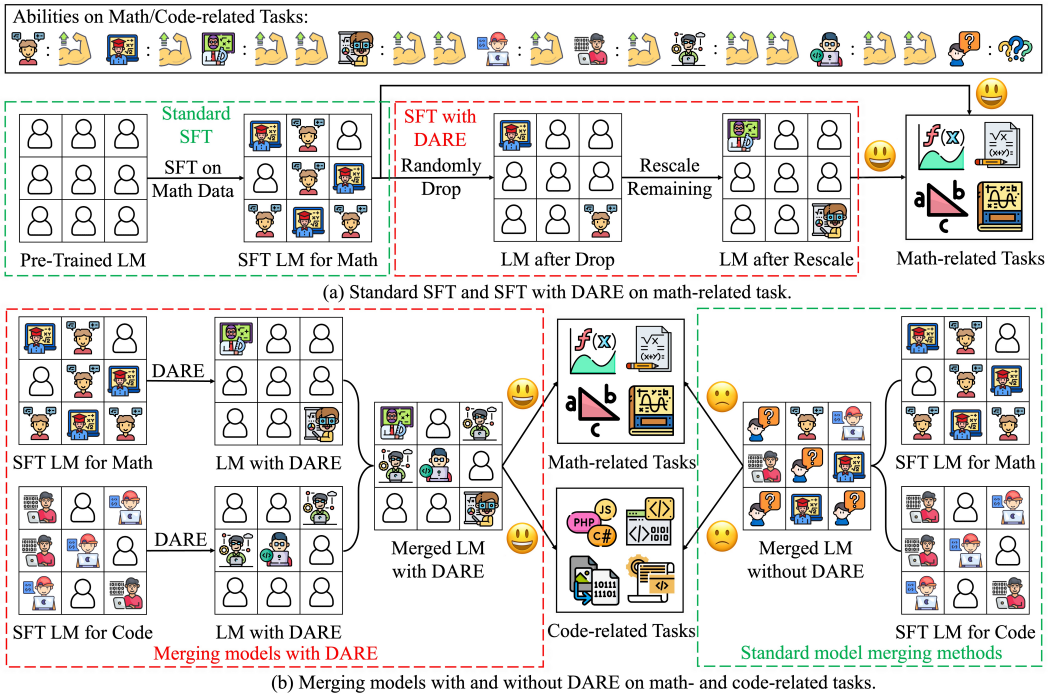
3.1 Preliminaries

SFT Delta Parameters. Let $\boldsymbol{\theta}_{\text{PRE}} \in \mathbb{R}^d$ denote the parameters of a pre-trained LM (d is the dimension), such as LLaMA [53] or Llama 2 [54]. For task t, SFT can provide a fine-tuned LM with parameters $\boldsymbol{\theta}^t_{\text{SFT}} \in \mathbb{R}^d$ by optimizing the pre-trained model on task-specific data. Give the parameters of both pre-trained LM ($\boldsymbol{\theta}_{\text{PRE}}$) and SFT LM ($\boldsymbol{\theta}^t_{\text{SFT}}$), we define delta parameters as the difference between parameters of LMs before and after SFT, which is computed by $\boldsymbol{\delta}^t = \boldsymbol{\theta}^t_{\text{SFT}} - \boldsymbol{\theta}_{\text{PRE}} \in \mathbb{R}^d$. Delta parameters reflect the changes in parameters during the SFT process and analyzing the properties of delta parameters can offer a better understanding of SFT.

Model Merging Problem. Given a set of tasks $\{t_1, t_2, \cdots, t_K\}$ and K fine-tuned models on these tasks with parameters $\{\boldsymbol{\theta}^{t_1}_{\text{SFT}}, \boldsymbol{\theta}^{t_2}_{\text{SFT}}, \cdots, \boldsymbol{\theta}^{t_K}_{\text{SFT}}\}$, model merging aims to integrate the parameters of K models into a single model that can well handle K tasks simultaneously. Following [44, 27, 62], we focus on merging fine-tuned models that are optimized from the same pre-trained backbone.

3.2 DARE: A Simple Approach for Reducing Delta Parameter Redundancy

In this work, we investigate the extremely redundant properties of the delta parameters of SFT LMs and propose DARE to effectively reduce delta parameter redundancy. DARE is the abbreviation of **D**rop **A**nd **RE**scale, whose workflow is shown in Figure 2(a). Given the delta parameters of an SFT LM, DARE first randomly drops the parameters based on a specific drop rate and then rescales the remaining ones to obtain parameters with reduced redundancy. We find DARE can approximately eliminate 90% or even 99% delta parameters without sacrificing much performance on both encoder- and decoder-based LMs.



(a) Standard SFT and SFT with DARE on math-related task.

(b) Merging models with and without DARE on math- and code-related tasks.

Figure 2: Illustrations of DARE and merging models with DARE. DARE can achieve comparable performance with standard SFT when it removes 90% or even 99% delta parameters. Moreover, DARE is able to tackle the parameter interference issue when merging models and yield consistent improvements. We denote the abilities of different elements for math/code-related tasks at the top.

DARE is conceptually simple and only consists of two steps: drop and rescale. Given delta parameters $\boldsymbol{\delta}^t = \boldsymbol{\theta}^t_{\text{SFT}} - \boldsymbol{\theta}_{\text{PRE}}$, DARE first performs random drop on $\boldsymbol{\delta}^t$ according to a drop rate p (resetting their values to zeros) and then rescales the remaining ones by a factor of $1/(1-p)$ as follows:

$$\begin{aligned} \boldsymbol{m}^t &\sim \text{Bernoulli}(p), \\ \widetilde{\boldsymbol{\delta}}^t &= \boldsymbol{m}^t \odot \boldsymbol{\delta}^t, \\ \hat{\boldsymbol{\delta}}^t &= \widetilde{\boldsymbol{\delta}}^t / (1-p). \end{aligned} \tag{1}$$

Finally, we combine $\hat{\boldsymbol{\delta}}^t$ and $\boldsymbol{\theta}_{\text{PRE}}$ via addition to obtain the parameters for inference, i.e., $\boldsymbol{\theta}^t_{\text{DARE}} = \hat{\boldsymbol{\delta}}^t + \boldsymbol{\theta}_{\text{PRE}}$. Note that the rescale operation is essential in DARE, aiming to keep the expectations of model outputs approximately unchanged. We will demonstrate its effectiveness in Section 4.4. Last but not least, we can find that DARE is somewhat similar to the operations in Dropout [52] but DARE is also distinct from Dropout in two major aspects: 1) DARE is designed to handle delta parameters while Dropout mainly focuses on fine-tuned parameters or model outputs; 2) DARE is used to reduce the delta parameter redundancy *without training*, which *permanently* eliminates delta parameters and only retrains the remaining delta parameters for inference. Dropout aims to prevent models from overfitting, which *temporarily* removes a part of the parameters during the training process. For inference, Dropout is deactivated and all the parameters will be used.

3.3 Model Merging with DARE

As DARE can effectively reduce the redundancy of delta parameters in LMs by resetting most of them to zeros, we hypothesize that DARE is able to help address the interference of parameters when merging multiple models [62]. Take Figure 2(b) as an example, when merging two task-specific models (for math- and code-related tasks, respectively), DARE can help existing model merging methods better absorb the abilities of both models with less or no parameter interference.

Formally, given K models that are fine-tuned on K tasks with parameters $\left\{\boldsymbol{\theta}^{t_1}_{\text{SFT}}, \boldsymbol{\theta}^{t_2}_{\text{SFT}}, \cdots, \boldsymbol{\theta}^{t_K}_{\text{SFT}}\right\}$, we first apply DARE on each parameters $\boldsymbol{\theta}^{t_k}_{\text{SFT}}, 1 \le k \le K$, deriving $\left\{\boldsymbol{\theta}^{t_1}_{\text{DARE}}, \boldsymbol{\theta}^{t_2}_{\text{DARE}}, \cdots, \boldsymbol{\theta}^{t_K}_{\text{DARE}}\right\}$. Then, we adopt existing model merging methods to merge the derived parameters and obtain the merged single model. Let us take Task Arithmetic [26] as an example, whose original computation process is denoted by

$$\boldsymbol{\theta}_{\text{M}} = \boldsymbol{\theta}_{\text{PRE}} + \lambda \cdot \sum_{k=1}^{K} \boldsymbol{\delta}^{t_k} = \boldsymbol{\theta}_{\text{PRE}} + \lambda \cdot \sum_{k=1}^{K} (\boldsymbol{\theta}^{t_k}_{\text{SFT}} - \boldsymbol{\theta}_{\text{PRE}}), \tag{2}$$

where λ is the scaling term to determine the importance of the models to be merged. When equipped with DARE, the calculation process of Task Arithmetic is rewritten as

$$\begin{aligned} \boldsymbol{\theta}^{t_k}_{\text{DARE}} &= \text{ DARE}\left(\boldsymbol{\theta}^{t_k}_{\text{SFT}}\right), \text{ for } 1 \le k \le K, \\ \boldsymbol{\theta}_{\text{M}} &= \boldsymbol{\theta}_{\text{PRE}} + \lambda \cdot \sum_{k=1}^{K} \hat{\boldsymbol{\delta}}^{t_k} = \boldsymbol{\theta}_{\text{PRE}} + \lambda \cdot \sum_{k=1}^{K} (\boldsymbol{\theta}^{t_k}_{\text{DARE}} - \boldsymbol{\theta}_{\text{PRE}}). \end{aligned} \tag{3}$$

In Section 4.3, we find that DARE can significantly improve the performance of Task Arithmetic when merging multiple LLMs. It is also worth noticing that DARE is employed as a preprocessing technique and can be generally applied to any existing model merging methods, such as Average Merging [59], Fisher Merging [44], RegMean [27], and TIES-Merging [62]. We will validate its generalizability in Section 4.3.

4 Experiments

In this section, we conduct extensive experiments on both encoder- and decoder-based LMs to demonstrate the effectiveness of DARE in reducing the redundancy of SFT delta parameters. We also show the ability of DARE to provide superior model merging performance.

4.1 Experimental Setup

Datasets and Pre-Trained Backbones for Decoder-based LMs. We choose AlpacaEval [36] for evaluating instruction-following models (WizardLM-7B, WizardLM-13B, and WizardLM-70B [61]),

and use GSM8K [8] and MATH [22] for testing mathematical reasoning models (WizardMath-7B, WizardMath-13B, and WizardMath-70B [41]). HumanEval [6] and MBPP [1] are adopted for estimating code-generating models (WizardCoder-Python-7B, WizardCoder-Python-13B, WizardCoder-Python-34B [42], and llama-2-13b-code-alpaca [5]). The above models are fine-tuned based on pre-trained backbones including LLaMA [53], Llama 2 [54], and Code Llama [49] with the corresponding model sizes. Please see Table 2 in Section A.1 for their versions and correspondences with pre-trained backbones.

Datasets and Pre-Trained Backbones for Encoder-based LMs. We conduct experiments on the GLUE benchmark [55] for encoder-based LMs, containing one sentence acceptability dataset CoLA [56], one sentiment detection dataset SST-2 [51], two paraphrase datasets MRPC [14] and QQP [50], one sentence similarity dataset STS-B [4], and three natural language inference datasets MNLI [3, 58], QNLI [48], and RTE [10, 19, 18, 2]. Note that WNLI [33] is not used since it requires tricks to obtain good performance [29]. All the datasets are designed for classification tasks except for STS-B, which is used for the regression task with similarity scores from 0 to 5. As the test labels of GLUE are not publicly available, we split the original training data into training and validation sets with the ratio of 90% and 10%. The original validation data is used as the test set. We choose bert-base-uncased[2] [11] and roberta-base[3] [39] as pre-trained backbones, and further fine-tune them to get SFT models on eight datasets.

Evaluation Metrics. AlpacaEval[4] uses win rate as the evaluation metric, which is computed by the fraction of times a powerful LLM (we use ChatGPT in this work) prefers the outputs from the target model over outputs from Text-Davinci-003. GSM8K and MATH are evaluated by zero-shot accuracy in solving mathematical problems. HumanEval[5] and MBPP[6] adopt pass@1 for evaluation, which denotes the fraction of generated individual code samples that pass the unit tests. For the GLUE benchmark[7], Matthews correlation coefficient is used for CoLA. Accuracy is utilized for estimating SST-2, QNLI, and RTE. Matched accuracy is adopted by MNLI. Accuracy and F1 score are computed for MRPC and QQP. STS-B leverages Pearson and Spearman correlation coefficients for evaluation.

Implementation Details. Since the decoder-based LMs we use have already been fine-tuned, they can be directly utilized for inference. Following [61, 41, 42], the inference is implemented by vLLM [30], where the batch size is set to 1 on AlpacaEval, HumanEval and MBPP, 60 on GSM8K, and 50 on MATH. Temperature is set to 0.0 for greedy decoding. The maximal number of generated tokens is set to 1,024 on GSM8K, and 2,048 on the other four datasets. For encoder-based LMs, We fine-tune bert-base-uncased and roberta-base for 10 epochs with a warmup strategy and choose the checkpoint with the best validation performance for testing. The weight decay is set to 0.01. We try using 1e-5 and 5e-5 as learning rates and present the optimal learning rate of each fine-tuned model in Table 3 in Section A.2. We set the batch size to 16 in both fine-tuning and inference processes. Experiments are conducted on NVIDIA Tesla V100 and A100 GPUs.

4.2 Extreme Redundancy in Delta Parameters of SFT LMs

We demonstrate the extremely redundant property of delta parameters of both decoder- and encoder-based LMs after SFT. To achieve such a goal, we vary the value of drop rate d in [0.0, 0.1, 0.2, $\cdots$, 0.9, 0.99] and apply DARE with each drop rate to get models after dropping the corresponding delta parameters. Note that when d is set to 0.0, we actually obtain the standard SFT models. We report the performance of various decoder-based LMs on AlpacaEval, GSM8K, and HumanEval as well as encoder-based LMs on eight GLUE datasets in Figure 3 and Figure 4. Please refer to the additional results of decoder-based LMs on MATH and MBPP in Figure 12 in Section B.1.

From the results, we can make the following conclusions. Firstly, *the delta parameters of both encoder- and decoder-based LMs are highly redundant*. In most cases, we can drop 90% delta parameters without significantly decreasing the performance (the models would even obtain better results when removing certain delta parameters). The drop rate can even be set to 99% under some

[2] https://huggingface.co/bert-base-uncased

[3] https://huggingface.co/roberta-base

[4] https://github.com/tatsu-lab/alpaca_eval

[5] https://github.com/openai/human-eval

[6] https://github.com/bigcode-project/bigcode-evaluation-harness

[7] https://gluebenchmark.com/

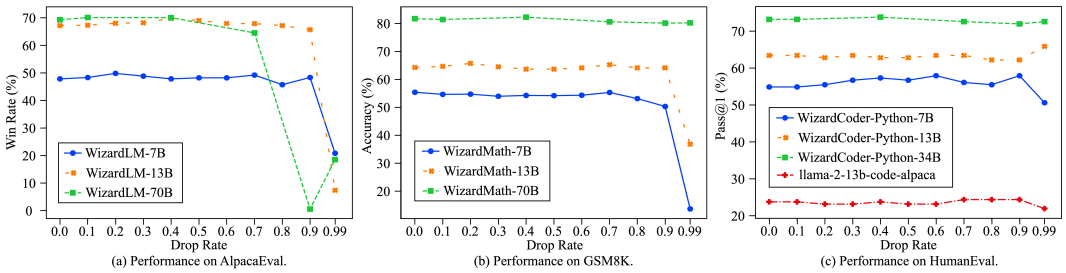


Figure 3: Performance of various decoder-based LMs on AlpacaEval, GSM8K, and HumanEval.

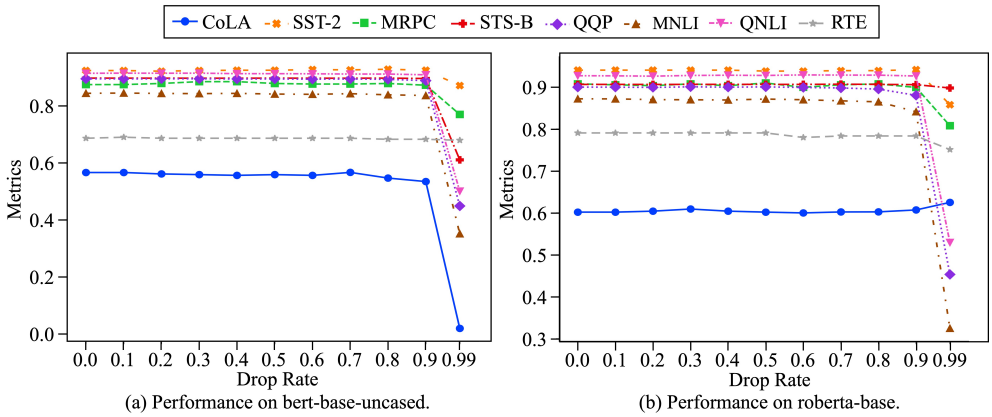


Figure 4: Performance under different drop rates on GLUE on encoder-based LMs.

conditions, such as WizardMath-70B, WizardCoder-Python series models, llama-2-13b-code-alpaca, bert-base-uncased on RTE, and roberta-base on CoLA, STS-B, and RTE. These observations reveal the extreme redundancy of delta parameters of SFT LMs and motivate the design of more effective and efficient SFT strategies.

Secondly, *the tolerance of drop rates increases with the sizes of LMs, i.e., LMs with larger sizes are able to work well with higher drop rates*. For example, WizardMath-70B performs well with a 0.99 drop rate while WizardMath-7B and WizardMath-13B fail. Similar trends also happen with WizardCoder-Python series models. This may be because LMs with larger sizes can show stronger abilities [57], and thus we hypothesize they learn a multitude of low-rank structures similar to LoRA [25] by just fine-tuning considerably fewer parameters during SFT. Such a property depicts some connections with the scaling laws of LMs [28, 23], indicating that there may exist quantifiable correlations between model sizes and drop rates they can afford.

Finally, we notice that the performance of WizardLM-70B drastically declines on AlpacaEval when the drop rate is 0.9 (different from the behaviors of WizardMath-70B and WizardCoder-Python-34B). One possible reason is that the instruction-following task on AlpacaEval is more difficult and requires general abilities with more delta parameters via SFT, causing more obvious dependencies among parameters (especially on LMs with larger sizes). Therefore, when the ratio of dropped delta parameters reaches a relatively small value (e.g., 0.9 in this case), the dependent relationships among parameters are destroyed, leading to unsatisfactory performance.

4.3 Merging Models with DARE on SFT LMs

We employ DARE as a preprocessing technique for existing model merging methods to verify its effectiveness in facilitating the merging performance. In practice, we first utilize DARE to obtain models after dropping some delta parameters and then merge these models by existing methods. We search the drop rate of DARE in $[0.1, 0.2, \cdots, 0.9]$ and select the optimal drop rate with the best performance. We conduct experiments on five model merging methods, including Average Merging

[59], Task Arithmetic [26], Fisher Merging [44], RegMean [27], and TIES-Merging [62]. Please refer to Section A.3 for the descriptions of these methods.

Note that for feasible computations, we only evaluate decoder-based LMs with Task Arithmetic by setting the scaling term to 0.5 and 1.0 and report the best results (Average Merging can be treated as a special case of Task Arithmetic when the scaling term is 0.5). Since model merging methods assume all the models are fine-tuned from the same pre-trained backbone, we choose to merge WizardLM-13B, WizardMath-13B, and llama-2-13b-code-alpaca, which all adopt Llama-2-13b as the pre-trained backbone. WizardCoder-Python-13B is not selected since it is fine-tuned based on CodeLlama-13b-Python. For encoder-based LMs, we experiment with all five methods for merging pairwise model and perform the grid search on several essential hyperparameters to find their optimal settings (see Table 4 in Section A.4 for more details). Following [27, 62], we also fine-tune the models under the multi-task learning setting and report the oracle results. We show the merging results of decoder-based LMs and partial results of encoder-based LMs in Table 1 and Figure 5. Please see Figure 13 in Section B.2 for additional results on merging encoder-based LMs on GLUE.

Table 1: Results of merging decoder-based LMs by Task Arithmetic, where LM, Math, and Code are the abbreviations of WizardLM-13B, WizardMath-13B, and llama-2-13b-code-alpaca. We use blue, green, and red colors to distinguish each single model and utilize mixed colors to denote the merged models. The best and second-best results among the single model, the merged models with and without DARE are marked in **bold** and underlined fonts.

Merging Methods	Models	Preprocess	Instruction-Following	Mathematical Reasoning		Code-Generating	
			AlpacaEval	GSM8K	MATH	HumanEval	MBPP
Single Model	LM	/	67.20	2.20	0.04	36.59	34.00
	Math	/	/	64.22	14.02	/	/
	Code	/	/	/	/	23.78	27.60
Task Arithmetic	LM & Math	No	67.04	**66.34**	13.40	28.66	30.60
		w/ DARE	**67.45**	66.26	12.86	26.83	32.40
	LM & Code	No	**68.07**	/	/	31.70	32.40
		w/ DARE	67.83	/	/	**35.98**	**33.00**
	Math & Code	No	/	64.67	13.98	8.54	8.60
		w/ DARE	/	**65.05**	13.96	10.37	9.80
	LM & Math & Code	No	69.03	58.45	9.88	18.29	29.80
		w/ DARE	**69.28**	56.48	10.16	23.17	**31.60**

From Table 1, we find that DARE usually facilitates the performance of Task Arithmetic on merging decoder-based LMs, and *it even achieves better results than the single model in some cases*, especially for instruction-following and code-generating tasks. This is quite different from previous merging methods that conclude the merged model often performs worse than the single model [44, 27, 62], demonstrating the effectiveness of the proposed DARE and revealing the superiority of decoder-based LMs compared with encoder-based LMs used in previous works. Moreover, since llama-2-13b-code-alpaca is not well fine-tuned for the code-generating task (it cannot perform better than WizardLM-13B), we hypothesize this will affect the model merging performance. Hence, we additionally evaluate the code-generating ability of the merged model of WizardLM-13B and WizardMath-13B. We observe that the merged model can obtain better results than llama-2-13b-code-alpaca though neither WizardLM-13B or WizardMath-13B is particularly fine-tuned on code-generating tasks, explaining the unsatisfactory results of the merged model of WizardMath-13B and llama-2-13b-code-alpaca. This observation suggests that a prerequisite for effective model merging is that each model to be merged should be well fine-tuned.

From Figure 5, we conclude that DARE can often yield modestly better results of various merging methods, achieving an average improvement of 0.58%, 0.36%, 0.37%, -0.03%, and 0.84% on Average Merging, Task Arithmetic, Fisher Merging, RegMean, and TIES Merging, respectively. However, the merged model still struggles to surpass the single model in most cases, which is in line with the observations in [44, 27, 62].

Last but not least, according to both Table 1 and Figure 5, we can further conclude that the improvements that DARE bring are more obvious in decoder-based LMs than encoder-based LMs. One most likely reason is that decoder-based LMs are able to accommodate more abilities than encoder-based LMs due to their much larger model sizes.

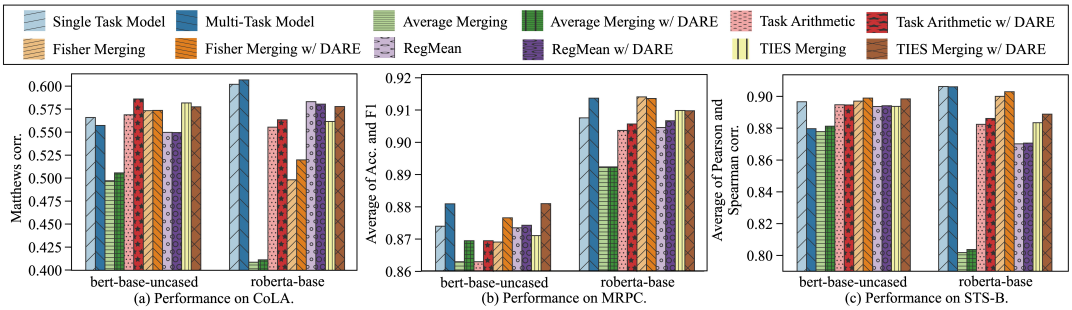


Figure 5: Results of merging encoder-based LMs on CoLA, MRPC, and STS-B. In each subfigure, the first two bars represent the results of models fine-tuned under single-task and multi-task settings. Other pairwise bars denote one specific model merging method and its combination with DARE.

4.4 Importance of the Rescale Operation

The rescale operation in DARE is quite essential, which aims to keep the expectations of model outputs holistically unchanged. To validate its importance, we consider the DropOnly variant that solely drops delta parameters based on the drop rate without rescaling the remaining delta parameters and compare it with DARE. Figure 6 and Figure 7 show partial results on decoder- and encoder-based LMs and please see Figure 14 and Figure 15 in Section B.3 for additional results.

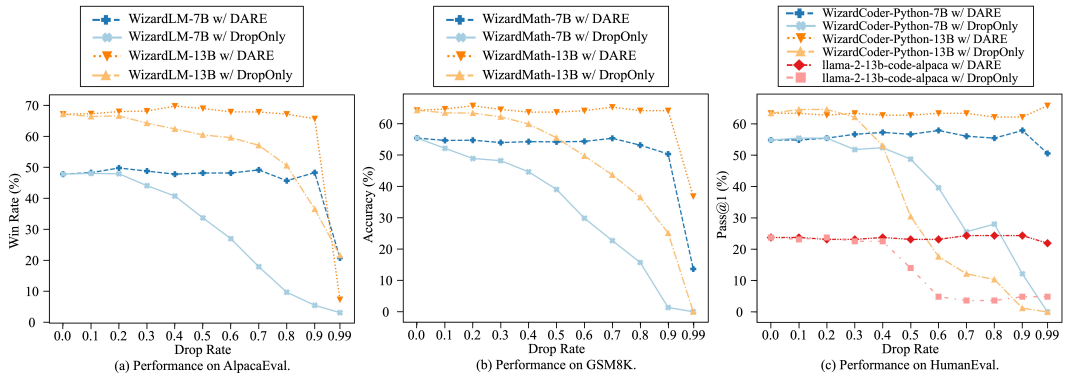


Figure 6: Comparisons between DARE and DropOnly on AlpacaEval, GSM8K, and HumanEval on decoder-based LMs.

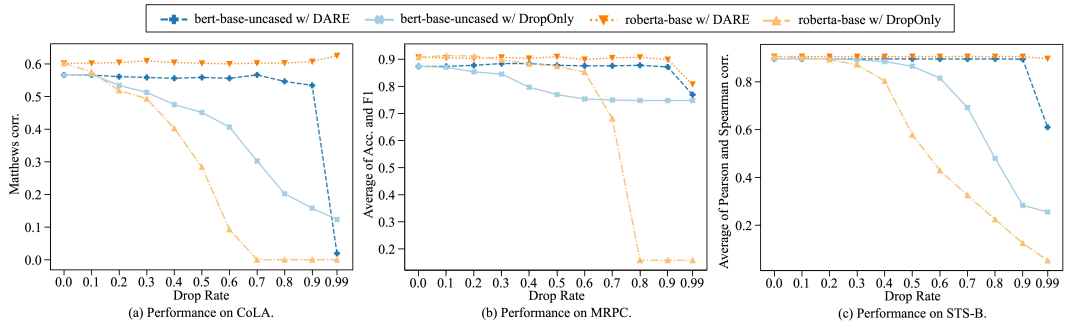


Figure 7: Comparing DARE and DropOnly on CoLA, MRPC, and STS-B on encoder-based LMs.

From the results, we can see that removing the rescale operation usually leads to much worse results, and the performance gaps between DARE and DropOnly become more significant with the increase in the drop rate. This observation demonstrates the effectiveness of the rescale operation in DARE and shows its power in eliminating more delta parameters.

4.5 Comparison with Magnitude-based Pruning

To demonstrate the advantage of DARE in removing delta parameters, we compare it with the commonly used Magnitude-based Pruning (MP) [20, 34, 31], which chooses parameters based on their magnitudes. For more fair and credible comparisons, we adapt classical MP to our settings by enabling it to operate on delta parameters and discard the (iterative) retraining process. The partial comparison results on decoder- and encoder-based LMs are illustrated in Figure 8 and Figure 9, and please refer to Figure 16 and Figure 17 in Section B.4 for extra results.

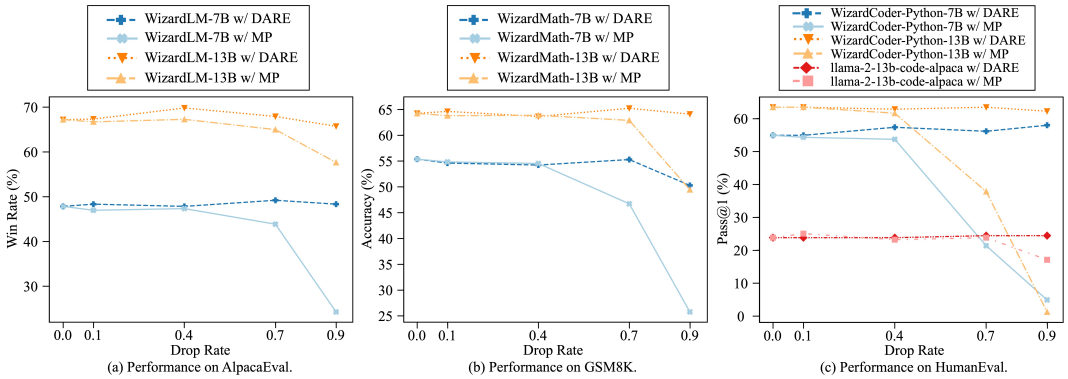


Figure 8: Comparisons between DARE and MP on AlpacaEval, GSM8K, and HumanEval on decoder-based LMs.

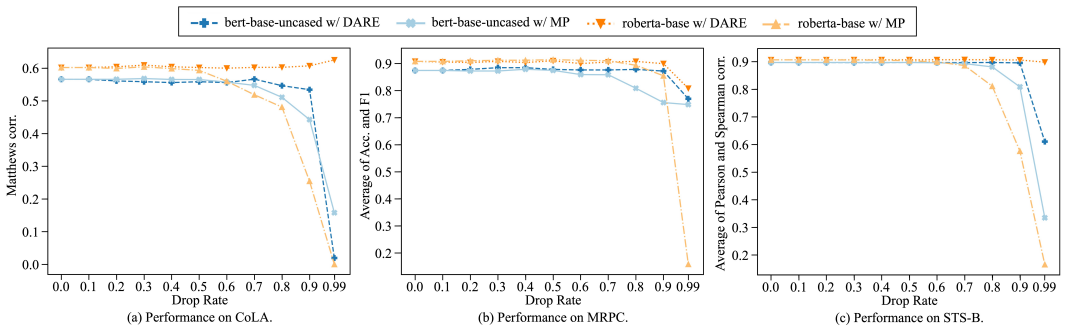


Figure 9: Comparing DARE and MP on CoLA, MRPC, and STS-B on encoder-based LMs.

From the results, we find that DARE outperforms MP in most cases and the superiority of DARE is more obvious when the drop rate becomes higher, verifying the effectiveness of DARE in abandoning delta parameters. We have also tried to combine MP with the rescale operation in DARE but got worse results than using MP separately. This is because MP removes parameters with smaller magnitudes and retains certain parameters with the largest magnitudes. Simply rescaling the remaining ones would amplify the expectations and result in unpredictable performance.

4.6 When Can DARE Be Used?

We also investigate the prerequisites that DARE can work. We choose Llama-2-13b instead of CodeLlama-13b-Python as the pre-trained backbone for WizardCoder-Python-13B and apply DARE to derive the model after dropping certain delta parameters for evaluation. We find that the pass@1 metrics on HumanEval/MBPP drastically decrease from 63.41/55.4 to 0.0/0.0 when only 10% delta parameters are removed. We deduce this is because Code Llama models are additionally trained with 500B tokens of code and code-related data [49], resulting in more obvious changes in parameter values with respect to Llama 2 models. Since WizardCoder-Python-13B is fine-tuned based on CodeLlama-13b-Python, when it uses Llama-2-13b as the pre-trained backbone, the ranges of SFT delta parameters would become much larger, making DARE lose effectiveness. To verify this, we depict the *absolute values* of SFT delta parameters of 13B decoder-based LMs vs. various pre-trained

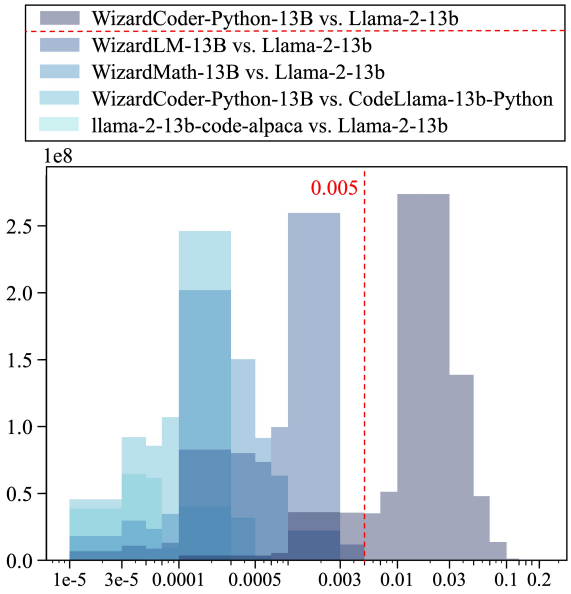


Figure 10: Delta parameter absolute values of 13B decoder-based LMs vs. the pre-trained backbones.

backbones in Figure 10. Please see Figure 18, Figure 19 and Figure 20 in Section B.5 for the SFT delta parameter ranges on decoder- and encoder-based LMs. Note that for decoder-based LMs, the results are plotted by randomly choosing 10% delta parameters due to their huge parameter numbers for feasibility. We also present the statistics about the deciles of delta parameter ranges of both decoder- and encoder-based LMs in Table 5 in Section B.5.

From the results, we find the absolute values of delta parameters of WizardCoder-Python-13B vs. CodeLlama-13b-Python (often greater than 0.01) are several orders of magnitude bigger than those of WizardCoder-Python-13B vs. Llama-2-13b (usually within 0.0002), causing the failure of DARE. For other 13B decoder-based LMs fine-tuned from Llama-2-13b, most of their absolute values of delta parameters are less than 0.005, making DARE a proper choice. To this end, we conclude that DARE can work well when the absolute values of SFT delta parameters are relatively small (e.g., less than 0.005). Otherwise, DARE may fail.

4.7 Can We Drop Fine-tuned Instead of Delta Parameters?

As previous network pruning methods mainly operate on the fine-tuned instead of delta parameters, we also conduct experiments under this setting. For decoder-based LMs, we find they perform badly when removing fine-tuned parameters even with 10% as the drop rate. Quantitatively, the performance sharply drops from 67.20 to 8.56 on AlpacaEval for WizardLM-13B, from 64.22/14.02 to 0.38/0.16 on GSM8K/MATH for WizardMath-13B, from 63.41/55.40 to 0.0/0.20 on HumanEval/MBPP for WizardCoder-Python-13B, from 23.78/27.60 to 0.0/0.0 on HumanEval/MBPP for llama-2-13b-code-alpaca. Similar observations can also be found on MP or decoder-based LMs with 7B and 70B sizes. Partial results on encoder-based LMs are shown in Figure 11 and please refer to Figure 21 in Section B.6 for additional results. We observe that directly eliminating the fine-tuned parameters by either DARE or MP would also lead to bad performance on encoder-based LMs. Moreover, decoder-based LMs tend to be affected more than encoder-based LMs and this may be because they are more capable and strongly correlated with the fine-tuned parameters. Removing a relatively small ratio of fine-tuned parameters would greatly harm their performance.

5 Conclusion

In this work, we discussed the extremely redundant properties of SFT delta parameters in LMs and proposed a simple approach DARE to effectively reduce the number of delta parameters needed for SFT without needing any extra data, training or even GPUs. DARE can impressively drop 90%

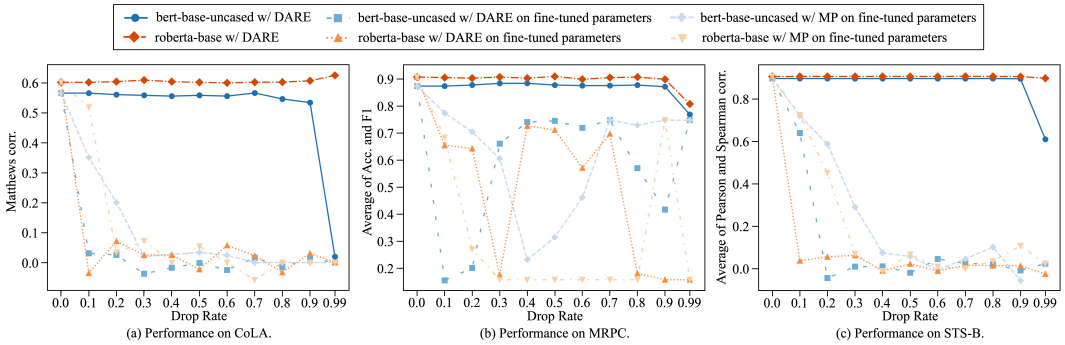


Figure 11: Performance of DARE and MP when dropping fine-tuned parameters on CoLA, MRPC, and STS-B on encoder-based LMs.

or even 99% SFT delta parameters without sacrificing much performance compared with using all SFT delta parameters. We further employed DARE as a general preprocessing technique for existing model merging methods to merge multiple task-specific fine-tuned models into a single model with diverse abilities. Extensive experimental results on both decoder- and encoder-based LMs demonstrated the effectiveness of DARE in reducing SFT delta parameter redundancy and facilitating the model merging performance. We also provided a deeper analysis of why DARE works as well as the prerequisites for using DARE. We hope our findings can inspire researchers to design more effective and efficient SFT strategies and believe DARE has the potential to serve as a promising technique for the federated learning field.

References

[1] Jacob Austin, Augustus Odena, Maxwell I. Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie J. Cai, Michael Terry, Quoc V. Le, and Charles Sutton. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021.

[2] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. *TAC*, 7:8, 2009.

[3] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642. The Association for Computational Linguistics, 2015.

[4] Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity - multilingual and cross-lingual focused evaluation. *CoRR*, abs/1708.00055, 2017.

[5] Sahil Chaudhary. Code alpaca: An instruction-following llama model for code generation. `https://github.com/sahil280114/codealpaca`, 2023.

[6] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé de Oliveira Pinto, Jared Kaplan, Harrison Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.

[7] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *CoRR*, abs/1710.09282, 2017.

[8] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.

[9] Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *CoRR*, abs/2009.09796, 2020.

[10] Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In Joaquin Quiñonero Candela, Ido Dagan, Bernardo Magnini, and Florence d'Alché-Buc, editors, *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer, 2005.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186. Association for Computational Linguistics, 2019.

[12] 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. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nat. Mac. Intell.*, 5(3):220–235, 2023.

[13] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah A. Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *CoRR*, abs/2002.06305, 2020.

[14] William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing, IWP@IJCNLP 2005*. Asian Federation of Natural Language Processing, 2005.

[15] Ronald A Fisher. On the mathematical foundations of theoretical statistics. *Philosophical transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, 222(594-604):309–368, 1922.

[16] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations*. OpenReview.net, 2019.

[17] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *CoRR*, abs/1902.09574, 2019.

[18] Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9, 2007.

[19] R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, volume 7, pages 785–794, 2006.

[20] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[21] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *The Tenth International Conference on Learning Representations*. OpenReview.net, 2022.

[22] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1*, 2021.

[23] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *CoRR*, abs/2203.15556, 2022.

[24] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 2019.

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

[26] Gabriel Ilharco, Marco Túlio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*. OpenReview.net, 2023.

[27] Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion by merging weights of language models. In *The Eleventh International Conference on Learning Representations*. OpenReview.net, 2023.

[28] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

[29] Vid Kocijan, Ana-Maria Cretu, Oana-Maria Camburu, Yordan Yordanov, and Thomas Lukasiewicz. A surprisingly robust trick for the winograd schema challenge. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 4837–4842. Association for Computational Linguistics, 2019.

[30] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626. ACM, 2023.

[31] Jaeho Lee, Sejun Park, Sangwoo Mo, Sungsoo Ahn, and Jinwoo Shin. Layer-adaptive sparsity for the magnitude-based pruning. In *9th International Conference on Learning Representations*. OpenReview.net, 2021.

[32] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059. Association for Computational Linguistics, 2021.

[33] Hector J. Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference*. AAAI Press, 2012.

[34] Guiying Li, Chao Qian, Chunhui Jiang, Xiaofen Lu, and Ke Tang. Optimization based layer-wise magnitude-based pruning for DNN compression. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 2383–2389. ijcai.org, 2018.

[35] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 4582–4597. Association for Computational Linguistics, 2021.

[36] Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`, 2023.

[37] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

[38] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT understands, too. *CoRR*, abs/2103.10385, 2021.

[39] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[40] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *7th International Conference on Learning Representations*. OpenReview.net, 2019.

[41] Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *CoRR*, abs/2308.09583, 2023.

[42] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. *CoRR*, abs/2306.08568, 2023.

[43] Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. Unipelt: A unified framework for parameter-efficient language model tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6253–6264. Association for Computational Linguistics, 2022.

[44] Michael Matena and Colin Raffel. Merging models with fisher-weighted averaging. In *NeurIPS*, 2022.

[45] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

[46] Yujia Qin, Cheng Qian, Jing Yi, Weize Chen, Yankai Lin, Xu Han, Zhiyuan Liu, Maosong Sun, and Jie Zhou. Exploring mode connectivity for pre-trained language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6726–6746, 2022.

[47] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[48] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392. The Association for Computational Linguistics, 2016.

[49] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton-Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. Code llama: Open foundation models for code. *CoRR*, abs/2308.12950, 2023.

[50] Iyer Shankar, Dandekar Nikhil, and Csernai Kornel. First quora dataset release: question pairs (2017). *URL https://www. quora. com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs*, 2017.

[51] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642. ACL, 2013.

[52] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

[53] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023.

[54] 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, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023.

[55] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations*. OpenReview.net, 2019.

[56] Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. Neural network acceptability judgments. *Trans. Assoc. Comput. Linguistics*, 7:625–641, 2019.

[57] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Trans. Mach. Learn. Res.*, 2022, 2022.

[58] Adina Williams, Nikita Nangia, and Samuel R. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1112–1122. Association for Computational Linguistics, 2018.

[59] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR, 2022.

[60] Mengzhou Xia, Zexuan Zhong, and Danqi Chen. Structured pruning learns compact and accurate models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1513–1528. Association for Computational Linguistics, 2022.

[61] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. *CoRR*, abs/2304.12244, 2023.

[62] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. Resolving interference when merging models. In *NeurIPS*, 2023.

[63] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2):12:1–12:19, 2019.

[64] Chen Zhang, Yu Xie, Hang Bai, Bin Yu, Weihong Li, and Yuan Gao. A survey on federated learning. *Knowl. Based Syst.*, 216:106775, 2021.

[65] Jinghan Zhang, Shiqi Chen, Junteng Liu, and Junxian He. Composing parameter-efficient modules with arithmetic operations. *CoRR*, abs/2306.14870, 2023.

[66] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Trans. Knowl. Data Eng.*, 34(12):5586–5609, 2022.

[67] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *CoRR*, abs/2303.18223, 2023.

[68] Michael Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. In *6th International Conference on Learning Representations*. OpenReview.net, 2018.

A Detailed Experimental Settings

A.1 Details of SFT and Pre-Trained Backbones of Decoder-based LMs

Table 2 gives versions and correspondences with pre-trained backbones of SFT decoder-based LMs.

Table 2: Versions and correspondences with pre-trained backbones of SFT decoder-based LMs.

Tasks	SFT Decoder-based LMs	Pre-Trained Backbones
Instruction-Following	WizardLM-7B[8]	llama-7b[9]
	WizardLM-13B[10]	Llama-2-13b[11]
	WizardLM-70B[12]	Llama-2-70b[13]
Mathematical Reasoning	WizardMath-7B[14]	Llama-2-7b[15]
	WizardMath-13B[16]	Llama-2-13b[11]
	WizardMath-70B[17]	Llama-2-70b[13]
Code-Generating	WizardCoder-Python-7B[18]	CodeLlama-7b-Python[19]
	WizardCoder-Python-13B[20]	CodeLlama-13b-Python[21]
	WizardCoder-Python-34B[22]	CodeLlama-34b-Python[23]
	llama-2-13b-code-alpaca[24]	Llama-2-13b[11]

A.2 Learning Rate Configurations of Encoder-based LMs on GLUE

The optimal settings of the learning rate of each fine-tuned model are presented in Table 3.

[8] https://huggingface.co/WizardLM/WizardLM-7B-V1.0
[9] https://huggingface.co/decapoda-research/llama-7b-hf
[10] https://huggingface.co/WizardLM/WizardLM-13B-V1.2
[11] https://huggingface.co/meta-llama/Llama-2-13b-hf
[12] https://huggingface.co/WizardLM/WizardLM-70B-V1.0
[13] https://huggingface.co/meta-llama/Llama-2-70b-hf
[14] https://huggingface.co/WizardLM/WizardMath-7B-V1.0
[15] https://huggingface.co/meta-llama/Llama-2-7b-hf
[16] https://huggingface.co/WizardLM/WizardMath-13B-V1.0
[17] https://huggingface.co/WizardLM/WizardMath-70B-V1.0
[18] https://huggingface.co/WizardLM/WizardCoder-Python-7B-V1.0
[19] https://huggingface.co/codellama/CodeLlama-7b-Python-hf
[20] https://huggingface.co/WizardLM/WizardCoder-Python-13B-V1.0
[21] https://huggingface.co/codellama/CodeLlama-13b-Python-hf
[22] https://huggingface.co/WizardLM/WizardCoder-Python-34B-V1.0
[23] https://huggingface.co/codellama/CodeLlama-34b-Python-hf
[24] https://huggingface.co/layoric/llama-2-13b-code-alpaca

Table 3: Configurations of learning rates of bert-base-uncased and roberta-base on GLUE.

Models	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	QNLI	RTE
bert-base-uncased	5e-5	1e-5	5e-5	5e-5	1e-5	1e-5	1e-5	1e-5
roberta-base	1e-5	1e-5	5e-5	1e-5	1e-5	1e-5	1e-5	1e-5

A.3 Descriptions of Existing Model Merging Methods

We use five model merging methods in the experiments:

- **Average Merging** simply averages the parameters of multiple models for building the merged model [59].
- **Task Arithmetic** uses a scaling term to control the contributions between the pre-trained backbone and the models to be merged [26].
- **Fisher Merging** first estimates the importance of parameters by calculating the Fisher information matrix, and then merges parameters based on their importance [44].
- **RegMean** recasts model merging as a linear regression problem and derives closed-form solutions to solve it [27].
- **TIES-Merging** aims to address the issue of task conflicts in model merging. It firstly trims parameters with lower magnitudes, and then resolves sign disagreements. Parameters with consistent signs are finally merged [62].

A.4 Details of Grid Search on Hyperparameters of Model Merging Methods for Encoder-based LMs

Table 4 shows searched ranges of model merging methods' hyperparameters for encoder-based LMs.

Table 4: Hyperparameter searched ranges of model merging methods for encoder-based LMs.

Model Merging Methods	Search Ranges of Hyperparameters
Task Arithmetic	scaling term to merge model parameters: [0.1, 0.3, 0.5, 0.7, 0.9, 1.0]
Fisher Merging	scaling term to merge model parameters: [0.1, 0.3, 0.5, 0.7, 0.9, 1.0], number of examples to compute Fisher information matrix: [256, 512, 1024, 2048]
RegMean	scaling term to reduce non-diagonal items: [0.1, 0.3, 0.5, 0.7, 0.9, 1.0], number of examples to compute inner product matrices: [256, 512, 1024, 2048]
TIES-Merging	scaling term to merge model parameters: [0.1, 0.3, 0.5, 0.7, 0.9, 1.0], ratio to retain parameters with largest-magnitude values: [0.1, 0.2, 0.3]

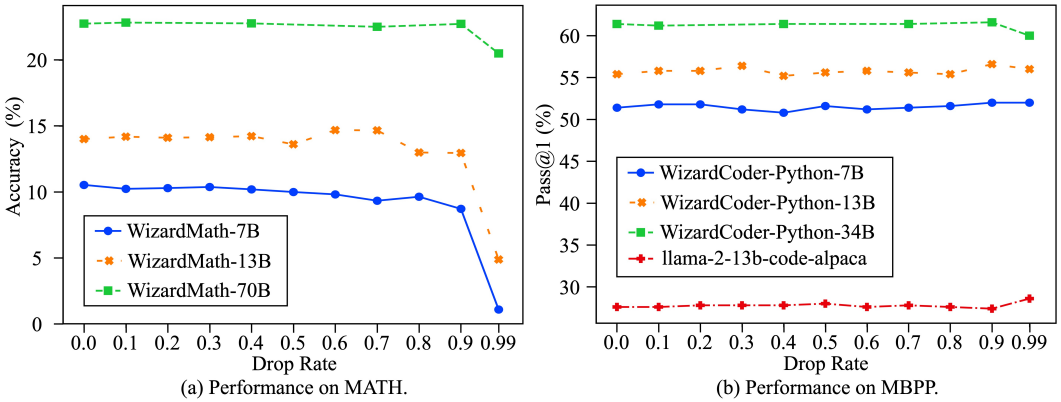
B Additional Experimental Results

B.1 Additional Results of Delta Parameter Redundancy of Decoder-based LMs

Figure 12 shows results of delta parameter redundancy of decoder-based LMs on MATH and MBPP.

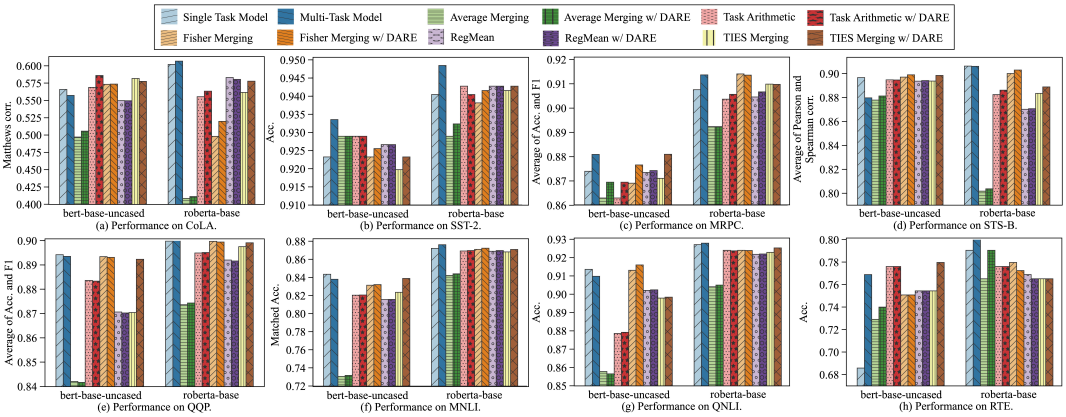
B.2 Additional Results of Merging Models with DARE on Encoder-based LMs

Figure 13 shows the performance of merging encoder-based LMs on GLUE.



(a) Performance on MATH. (b) Performance on MBPP.

Figure 12: Performance of various decoder-based LMs on MATH and MBPP.



(a) Performance on CoLA. (b) Performance on SST-2. (c) Performance on MRPC. (d) Performance on STS-B.
(e) Performance on QQP. (f) Performance on MNLI. (g) Performance on QNLI. (h) Performance on RTE.

Figure 13: Performance of merging encoder-based LMs on GLUE.

B.3 Additional Results of Comparisons between DARE and DropOnly

The comparison results between DARE and DropOnly on Math and MBPP on decoder-based LMs and all results on GLUE on encoder-based LMs are shown in Figure 14 and Figure 15, respectively.

B.4 Additional Results of Comparisons between DARE and MP on MATH and MBPP

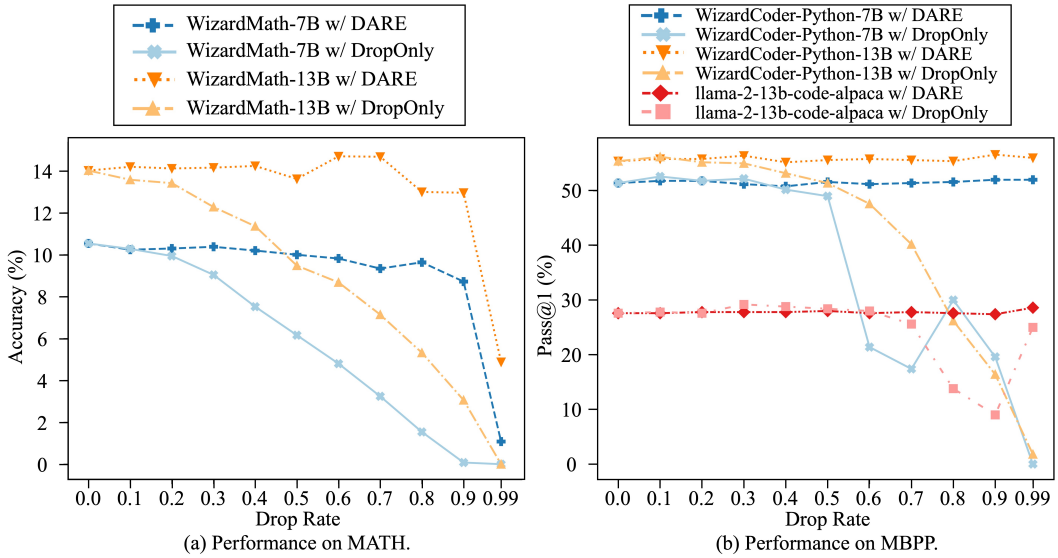
Comparisons between DARE and magnitude-based pruning on Math and MBPP on decoder-based LMs and all results on GLUE on encoder-based LMs are shown in Figure 16 and Figure 17.

B.5 Ranges of SFT Delta Parameters of Decoder-based LMs and Encoder-based LMs

We show the SFT delta parameter ranges of decoder- and encoder-based LMs in Figure 18, Figure 19 and Figure 20. We also give the statistics about the deciles of delta parameter ranges in Table 5, which are computed on 0.1%/10% randomly chosen parameters of decoder-based/encoder-based LMs for feasibility.

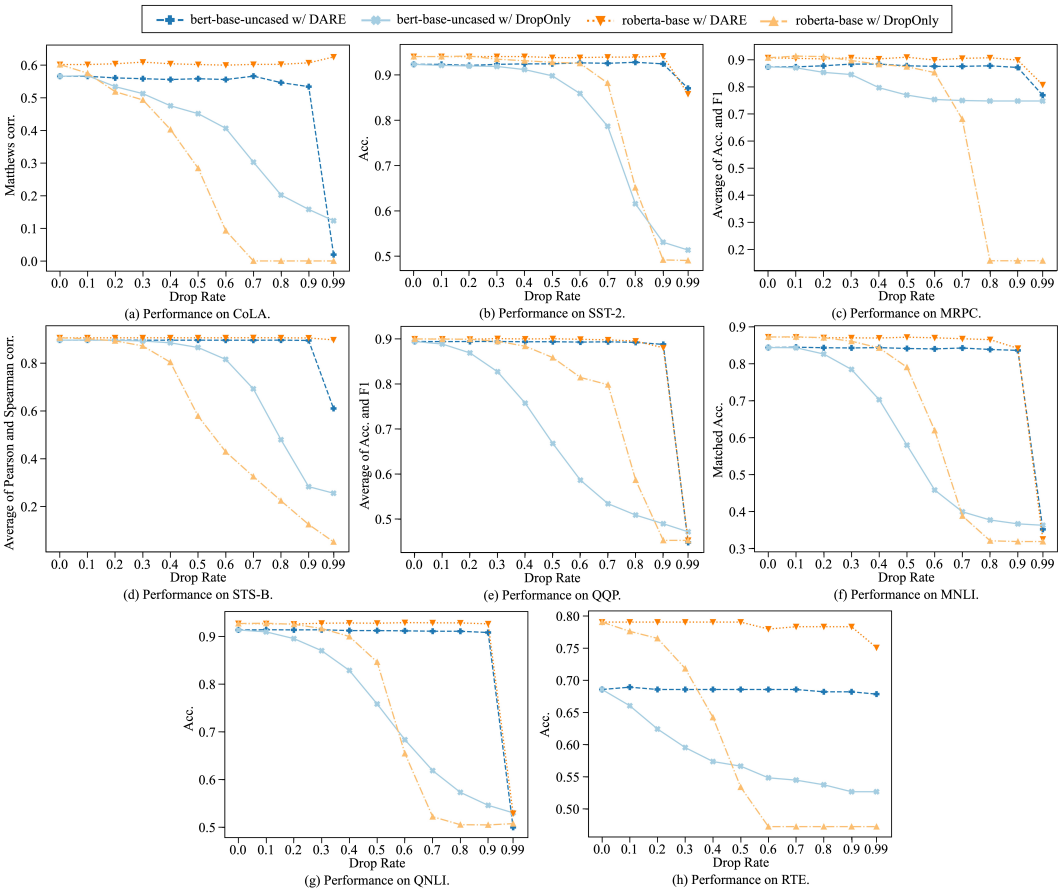
B.6 Additional Results of Dropping Fine-tuned Parameters on Encoder-based LMs

Figure 21 shows the results of removing fine-tuned parameters on GLUE on encoder-based LMs.



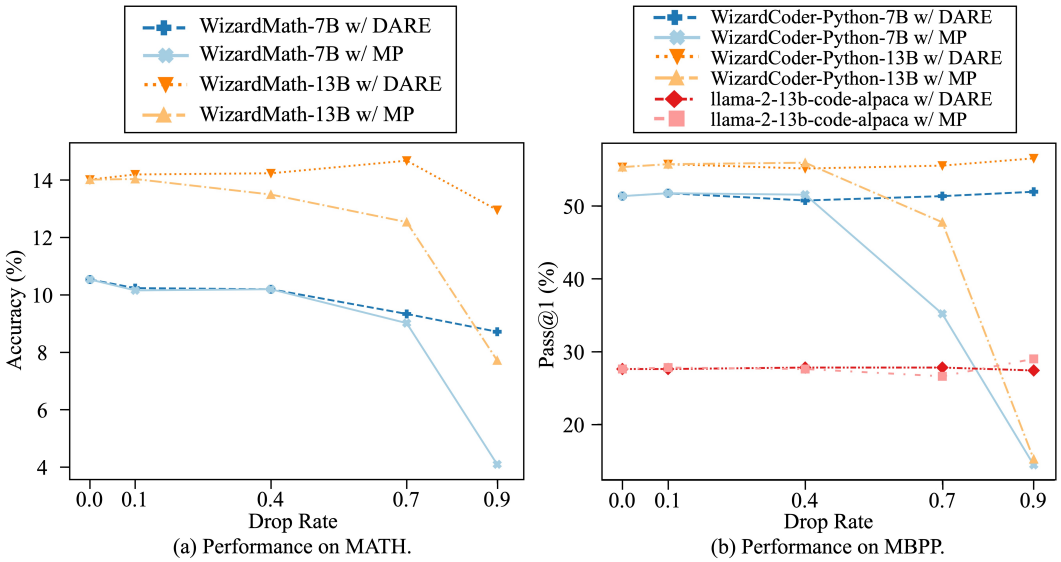
(a) Performance on MATH. (b) Performance on MBPP.

Figure 14: Comparisons between DARE and DropOnly on Math and MBPP on decoder-based LMs.



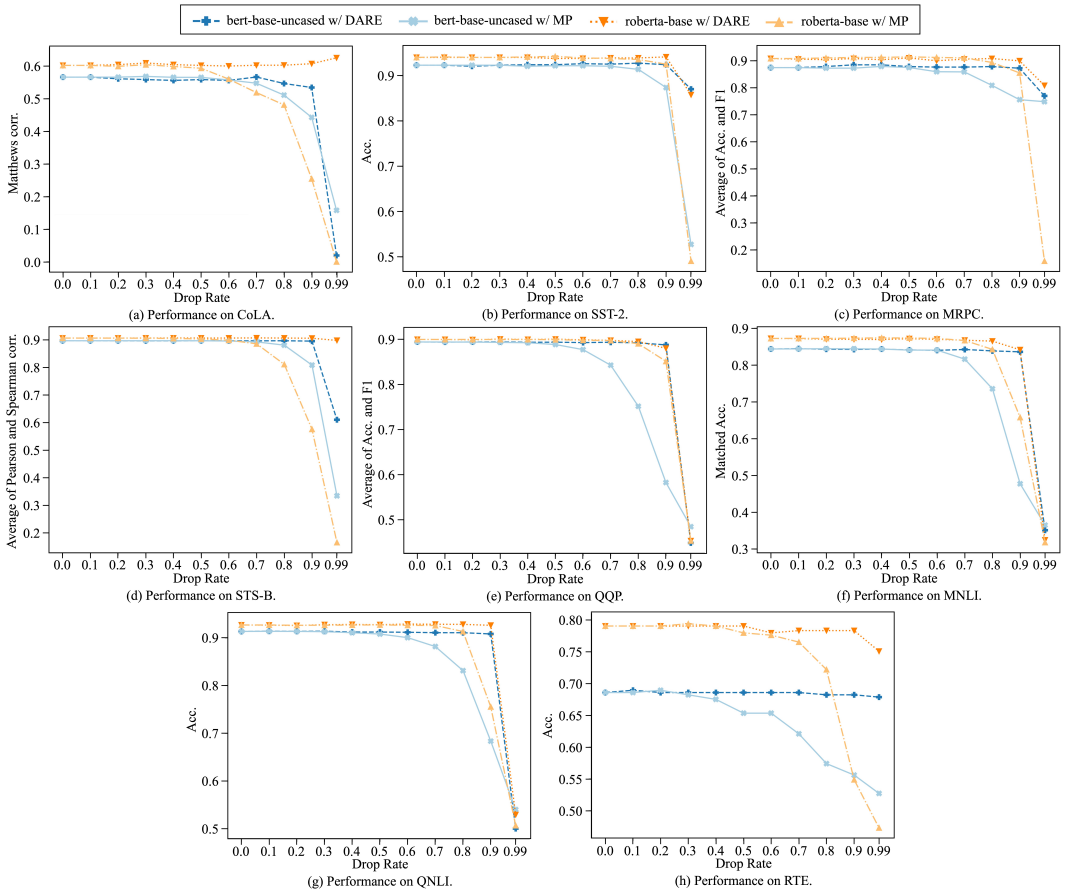
(a) Performance on CoLA. (b) Performance on SST-2. (c) Performance on MRPC.
(d) Performance on STS-B. (e) Performance on QQP. (f) Performance on MNLI.
(g) Performance on QNLI. (h) Performance on RTE.

Figure 15: Comparisons between DARE and DropOnly on GLUE on encoder-based LMs.



(a) Performance on MATH. (b) Performance on MBPP.

Figure 16: Comparisons between DARE and MP on Math and MBPP on decoder-based LMs.



(a) Performance on CoLA. (b) Performance on SST-2. (c) Performance on MRPC.
(d) Performance on STS-B. (e) Performance on QQP. (f) Performance on MNLI.
(g) Performance on QNLI. (h) Performance on RTE.

Figure 17: Comparisons between DARE and MP on GLUE on encoder-based LMs.

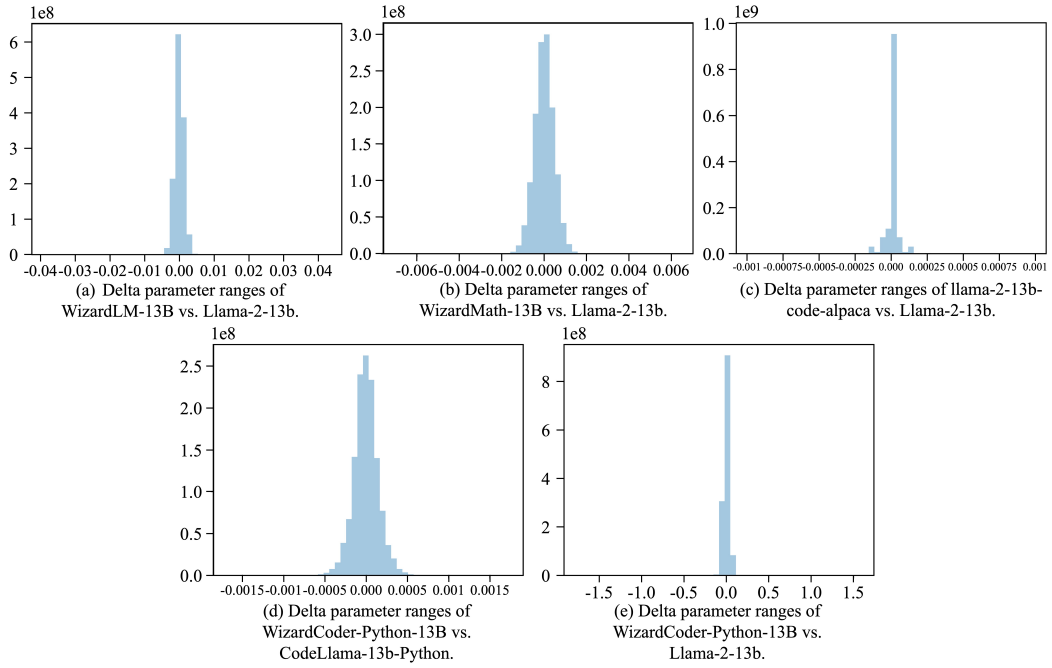


Figure 18: Delta parameter ranges of 13B decoder-based LMs vs. the pre-trained backbones.

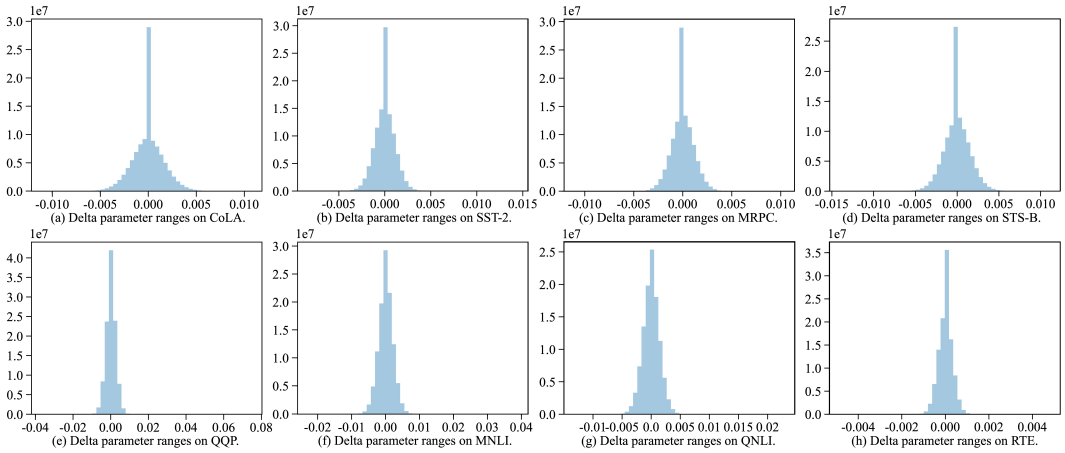


Figure 19: Delta parameter ranges of bert-base-uncased after SFT on GLUE.

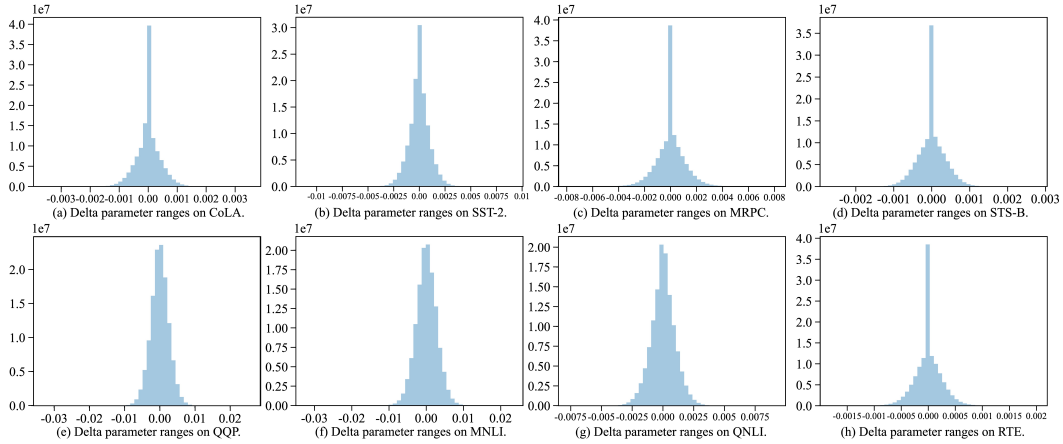


Figure 20: Delta parameter ranges of roberta-base after SFT on GLUE.

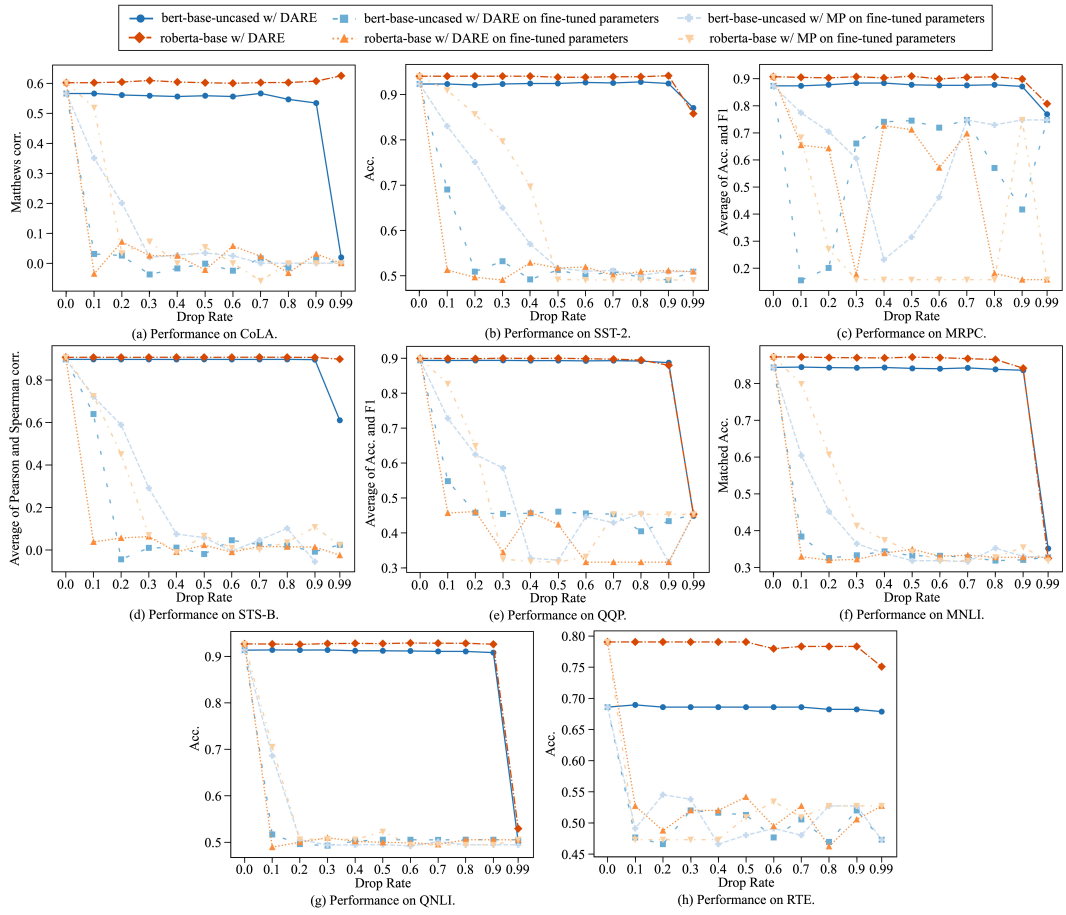


Figure 21: Performance of DARE and MP when dropping fine-tuned parameters on GLUE on encoder-based LMs.

Table 5: Statistics about the deciles of delta parameter ranges of both decoder- and encoder-based LMs.

Models		min	10%	20%	30%	40%	50%	60%	70%	80%	90%	max
WizardLM-13B vs. Llama-2-13b		-0.0285	-0.0016	-0.0010	-0.0006	-0.0003	0.0000	0.0003	0.0006	0.0010	0.0016	0.0341
WizardMath-13B vs. Llama-2-13b		-0.0048	-0.0006	-0.0004	-0.0002	-0.0001	0.0000	0.0001	0.0002	0.0004	0.0006	0.0047
llama-2-13b-code-alpaca vs. Llama-2-13b		-7.3242e-04	-3.0518e-05	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	3.0518e-05	7.3242e-04
WizardCoder-Python-13B vs. CodeLlama-13b-Python		-1.3733e-03	-1.8120e-04	-1.0681e-04	-6.8665e-05	-3.4332e-05	0.0000e+00	3.3379e-05	6.8665e-05	1.0681e-04	1.8120e-04	1.4648e-03
WizardCoder-Python-13B vs. Llama-2-13b		-0.5192	-0.0381	-0.0247	-0.0153	-0.0073	0.0000	0.0073	0.0153	0.0247	0.0381	0.7246
bert-base-uncased	CoLA	-1.0062e-02	-1.9900e-03	-1.1620e-03	-5.6932e-04	-7.3511e-05	3.0810e-05	1.0663e-04	5.7345e-04	1.1661e-03	1.9936e-03	1.0592e-02
	SST-2	-7.3227e-03	-1.3287e-03	-8.0843e-04	-4.2815e-04	-1.1123e-04	4.8140e-05	1.9256e-04	4.4204e-04	8.2212e-04	1.3412e-03	6.6411e-03
	MRPC	-8.0786e-03	-1.4064e-03	-8.4427e-04	-4.4834e-04	-1.0060e-04	8.5477e-06	1.0558e-04	4.5363e-04	8.4961e-04	1.4107e-03	9.7023e-03
	STS-B	-1.3253e-02	-1.8758e-03	-1.1282e-03	-5.9211e-04	-1.2640e-04	2.1029e-05	1.2679e-04	5.9258e-04	1.1304e-03	1.8794e-03	9.5631e-03
	QQP	-0.0216	-0.0032	-0.0019	-0.0011	-0.0004	0.0002	0.0006	0.0013	0.0021	0.0032	0.0306
	MNLI	-1.9360e-02	-2.6150e-03	-1.6359e-03	-9.5239e-04	-3.9054e-04	7.4751e-05	4.7965e-04	1.0297e-03	1.6948e-03	2.6518e-03	2.2719e-02
	QNLI	-1.2412e-02	-1.7661e-03	-1.1141e-03	-6.5727e-04	-2.7595e-04	4.4333e-05	3.3008e-04	7.0713e-04	1.1566e-03	1.7944e-03	1.0278e-02
	RTE	-4.2236e-03	-3.9333e-04	-2.4141e-04	-1.3099e-04	-3.4146e-05	2.1234e-06	3.5670e-05	1.3276e-04	2.4325e-04	3.9541e-04	4.3249e-03
roberta-base	CoLA	-3.2903e-03	-4.9453e-04	-2.6499e-04	-1.0061e-04	-2.5153e-05	1.8091e-06	2.8946e-05	1.0061e-04	2.6651e-04	4.9653e-04	3.0104e-03
	SST-2	-7.5466e-03	-1.1795e-03	-6.6938e-04	-3.4303e-04	-1.4345e-04	1.0615e-05	1.6433e-04	3.8091e-04	6.8196e-04	1.1915e-03	7.6943e-03
	MRPC	-7.0281e-03	-1.2577e-03	-6.8674e-04	-2.9566e-04	-5.8427e-05	2.8922e-06	7.0035e-05	2.9370e-04	6.8584e-04	1.2580e-03	8.0592e-03
	STS-B	-2.6313e-03	-4.4548e-04	-2.5395e-04	-1.1136e-04	-2.1163e-05	1.2186e-06	3.3885e-05	1.1186e-04	2.5456e-04	4.4597e-04	2.3694e-03
	QQP	-2.8996e-02	-3.1613e-03	-2.0129e-03	-1.1793e-03	-5.3198e-04	6.3781e-05	6.5139e-04	1.3161e-03	2.1409e-03	3.2998e-03	2.2995e-02
	MNLI	-2.8375e-02	-3.3906e-03	-2.1666e-03	-1.2824e-03	-5.7767e-04	6.9943e-05	7.0991e-04	1.4290e-03	2.3107e-03	3.5429e-03	2.0889e-02
	QNLI	-7.6891e-03	-1.2210e-03	-7.5125e-04	-4.3479e-04	-1.8336e-04	8.6576e-06	2.0984e-04	4.5467e-04	7.6704e-04	1.2338e-03	6.8666e-03
	RTE	-1.6544e-03	-2.9030e-04	-1.6271e-04	-7.3047e-05	-1.0118e-05	3.8603e-07	1.2353e-05	7.3839e-05	1.6338e-04	2.9109e-04	1.7324e-03