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

Le Yu [1] **Bowen Yu** [1] **Haiyang Yu** [1] **Fei Huang** [1] **Yongbin Li** [1]

Abstract

In this paper, we unveil that Language Models (LMs) can acquire new capabilities by assimilating parameters from homologous models without retraining or GPUs. We first introduce DARE to set most delta parameters (i.e., the disparity between fine-tuned and pre-trained parameters) to zeros without affecting the abilities of Supervised Fine-Tuning (SFT) LMs, which randomly **D**rops delta parameters with a ratio p **A**nd **RE**scales the remaining ones by $1/(1-p)$ to approximate the original embeddings. Then, we use DARE as a versatile plug-in to sparsify delta parameters of multiple SFT homologous models for mitigating parameter interference and merge them into a single model by parameter fusing. We experiment with encoder- and decoder-based LMs, showing that: (1) SFT delta parameter value ranges are typically small (within 0.002) with extreme redundancy, and DARE can effortlessly eliminate 90% or even 99% of them; (2) DARE can merge multiple task-specific LMs into one LM with diverse capabilities. Notably, this phenomenon is more pronounced in large-scale LMs, where the merged LM reveals the potential to surpass the performance of any source LM, providing a new discovery. We also utilize DARE to create a merged LM that ranks first among models with 7 billion parameters on the Open LLM Leaderboard.

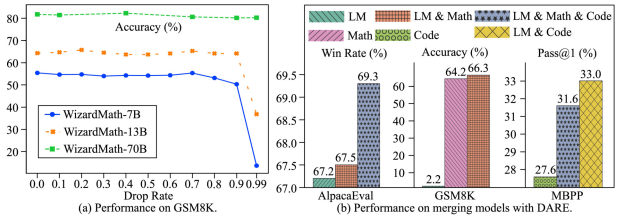

Figure 1: (**Left**) DARE can effectively eliminate 90% or even 99% delta parameters of WizardMath on GSM8K. (**Right**) DARE can merge multiple task-specific SFT language models into a single model with all the abilities. LM, MATH, and Code are abbreviations of WizardLM-13B, WizardMath-13B, and llama-2-13b-code-alpaca.

1. Introduction

Human beings have harbored a longstanding desire to acquire additional abilities through various ways, as expressed in mediums like 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. In this paper, we astonishingly find that Language Models (LMs), similar to Apocalypse and Super Mario, can enhance their capabilities by absorbing other models without the need for retraining or even GPUs.

Formally, Supervised Fine-Tuning (SFT) is the most widely adopted strategy for unlocking task-specific abilities to LMs by optimizing their parameters (Dodge et al., 2020; Zhao et al., 2023). The effectiveness of SFT is fully evident in the alteration of the model parameters before and after SFT, referred to as *delta parameters* (Ding et al., 2023). We first show that SFT LM (either encoder- or decoder-based) always tends to acquire excessively redundant delta parameters. To be specific, we present DARE (**D**rop **A**nd **RE**scale), which randomly sets certain delta parameters to zeros with a drop rate p and subsequently rescales the remaining ones by a factor of $1/(1-p)$. Although conceptually simple, DARE can eliminate up to 99% delta parameters with minimal impact on the performance when the LM's parameters reach 70 billion (see Figure 1(a)). Moreover, the more parameters the LM has, the larger p it can tolerate. We attribute the effectiveness of DARE to its ability to approximate the original embeddings, which is verified theoretically and empirically.

Furthermore, we can merge multiple homologous SFT LMs (fine-tuned from the same backbone) based on DARE with-

[1]Alibaba Group. Correspondence to: Bowen Yu <yubowen.ybw@alibaba-inc.com>, Yongbin Li <shuide.lyb@alibaba-inc.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

out compromising their capabilities. As long as a small portion of the delta parameters remain unaffected during merging, 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 (Yadav et al., 2023). Then, we apply established model merging techniques (Wortsman et al., 2022; Ilharco et al., 2023; Matena & Raffel, 2022; Jin et al., 2023; Yadav et al., 2023) to fuse the parameters with reduced redundancy for creating one model with diverse capabilities.

We conduct extensive experiments with encoder-based LMs on GLUE benchmark, and decoder-based LMs 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 regardless of their backbones (e.g., BERT, RoBERTa, LLaMA, Llama 2, or Code Llama). DARE can *remove 90% or even 99% delta parameters* without significantly affecting the model performance. DARE is able to approximate the original embeddings well and provide very similar embeddings for each layer of the LM. The rescale operation is crucial to guarantee the success of DARE, and dropping 30% or 40% delta parameters without rescaling would noticeably lead to worse results.

(2) DARE often retains or enhances the performance of various model merging methods on encoder-based LMs. For decoder-based LMs, simply averaging the parameters can already yield satisfactory results. As shown in Figure 1(b), we merge various decoder-based LMs by DARE and Task Arithmetic (Ilharco et al., 2023), leading to considerable improvements. For example, 3.10% for LM & Math & Code vs. LM on AlpacaEval, 3.18% for LM & Math vs. Math on GSM8K, and 19.57% for LM & Code vs. Code on MBPP. We also use DARE to create a merged LM with 7 billion parameters, *attaining the top-ranking position on the Open LLM Leaderboard*. It is fascinating that all the benefits are achieved by solely using CPUs without retraining.

(3) SFT delta parameters usually stay within 0.002, indicating minimal modifications to the pre-trained LM, and DARE works for delta parameters with relatively small value ranges. However, once models undergo continuous pre-training, the delta parameters can rapidly reach around 0.03, making DARE infeasible. Moreover, dropping only 10% fine-tuned parameters (i.e., the combination of pre-trained and delta parameters) would lead to a catastrophic decrease in performance, even approaching zero. This finding further confirms that SFT primarily unlocks the abilities of pre-trained LMs, rather than introducing new capabilities.

The used resources are publicly available at `https://github.com/yule-BUAA/MergeLM`.

2. Related Work

Supervised Fine-tuning of Language Models. SFT of LMs aims to impart pre-trained LMs with particular abilities by optimizing them on task-specific data, which has become the de facto standard paradigm in natural language processing (Dodge et al., 2020; Zhao et al., 2023). Generally, SFT can be divided into two categories: full fine-tuning (Radford et al., 2018; Devlin et al., 2019) and parameter-efficient fine-tuning (Houlsby et al., 2019; Liu et al., 2021; Li & Liang, 2021; Lester et al., 2021; Hu et al., 2022). Indeed, the effects of SFT are reflected by the difference between parameters of LMs before and after SFT, i.e., delta parameters. In this paper, we reveal the extreme redundancy of various SFT LMs' delta parameters by proposing an innovative approach DARE, achieving competitive performance with standard SFT LMs by removing 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 (Cheng et al., 2017; Liang et al., 2021). The objective of network pruning is to eliminate unnecessary parameters while maintaining the model performance (Zhu & Gupta, 2018; Liu et al., 2019b; Frankle & Carbin, 2019; Gale et al., 2019; Xia et al., 2022). Magnitude-based pruning is one classical pruning method, which selects parameters according to their magnitudes (i.e., absolute parameter values) (Han et al., 2015; Li et al., 2018; Lee et al., 2021). To be specific, parameters with magnitudes lower than a certain threshold are removed, and others are preserved. In fact, DARE is relevant to the concept of network pruning as it can also drop parameters. But DARE differs from existing pruning techniques in: (1) DARE focuses on delta parameters while most pruning methods deal with fine-tuned parameters; (2) DARE can work well without any retraining or extra data, which are often inevitably required by pruning methods.

Model Merging. Model merging has become a trending research direction in recent years, aiming to merge multiple task-specific models into a single model with diverse abilities (Wortsman et al., 2022; Matena & Raffel, 2022; Ilharco et al., 2023; Jin et al., 2023; Yadav et al., 2023; Zhang et al., 2023). The superiority of model merging over multi-task learning (Crawshaw, 2020; Zhang & Yang, 2022) (which also intends to obtain one model with several abilities) is that model merging pays attention to the fusion of model parameters without accessing the original training data (Matena & Raffel, 2022; Jin et al., 2023). Average Merging (Wortsman et al., 2022) is one common model merging approach, which utilizes averaged parameters to construct the merged model. Task Arithmetic (Ilharco et al., 2023) employs a pre-defined scaling term to distinguish the importance of various models. Fisher Merging (Matena & Raffel, 2022) performs weighted fusions of parameters,

where the weights are calculated by the Fisher information matrix (Fisher, 1922). RegMean (Jin et al., 2023) masterly solves model merging by optimizing a linear regression problem with closed-form solutions. TIES-Merging (Yadav et al., 2023) tackles the task conflicts in Ilharco et al. (2023) by trimming low-magnitude parameters, resolving sign disagreements, and disjointly merging parameters with consistent signs. In this paper, we use DARE as a versatile plug-in 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 capabilities of all the SFT models.

3. Methodology

SFT Delta Parameters. Let $\boldsymbol{\theta}_{\text{PRE}} \in \mathbb{R}^d$ denote the parameters of a pre-trained LM (d is the parameter dimension), such as LLaMA (Touvron et al., 2023a) or Llama 2 (Touvron et al., 2023b). 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 ($\boldsymbol{\theta}^t_{\text{SFT}}$), delta parameters are defined as the difference between parameters of LMs before and after SFT, i.e., $\boldsymbol{\delta}^t = \boldsymbol{\theta}^t_{\text{SFT}} - \boldsymbol{\theta}_{\text{PRE}} \in \mathbb{R}^d$. Since delta parameters reflect the changes in parameters during the SFT process, analyzing the properties of delta parameters can offer a better understanding of SFT.

Model Merging Problem. Given a set of K tasks $\{t_1, t_2, \cdots, t_K\}$ and K corresponding SFT models 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 fuse the parameters of K models into a single model with parameters $\boldsymbol{\theta}_{\text{M}}$ that can well handle K tasks simultaneously. Following Matena & Raffel (2022); Jin et al. (2023); Yadav et al. (2023), we focus on merging fine-tuned models that are optimized from the same pre-trained backbone.

3.1. DARE: A Simple Approach for Reducing Delta Parameter Redundancy

In this work, we reveal the extremely redundant properties of the delta parameters of SFT LMs and propose DARE to effectively reduce delta parameter redundancy (see Figure 2(a)). DARE is conceptually simple and 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 $\boldsymbol{\delta}^t$ based on a drop rate p (setting their values to zeros) and then rescales the remaining ones by a factor of $1/(1-p)$ as follows,

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

Finally, we combine $\hat{\boldsymbol{\delta}}^t$ and $\boldsymbol{\theta}_{\text{PRE}}$ via addition to obtain the parameters for inference, i.e., $\boldsymbol{\theta}^t_{\text{DARE}} = \hat{\boldsymbol{\delta}}^t + \boldsymbol{\theta}_{\text{PRE}}$. We prove that even after removing most delta parameters, DARE can well preserve the model performance by approximating the original embeddings.

Theoretical Analysis. We discuss linear transformation since most parameters of LMs play a role in this basic operation (e.g., the computations in feed-forward networks, the projections of queries, keys, values, and outputs in self-attention modules). Let $\boldsymbol{W}/\Delta\boldsymbol{W} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b}/\Delta\boldsymbol{b} \in \mathbb{R}^m$ be the pre-trained/delta parameters. The input is a vector $\boldsymbol{x} \in \mathbb{R}^n$. Expectation of the i-th ($1 \le i \le m$) dimension of the original embeddings $\boldsymbol{h} \in \mathbb{R}^m$ is computed by

$$\begin{aligned} \mathbb{E}[h_i] &= \mathbb{E}[\sum_{j=1}^{n} \left(w_{ij} + \Delta w_{ij}\right) x_j + \left(b_i + \Delta b_i\right)] \\ &= \sum_{j=1}^{n} x_j \mathbb{E}[w_{ij}] + \mathbb{E}[b_i] + \sum_{j=1}^{n} x_j \mathbb{E}[\Delta w_{ij}] + \mathbb{E}[\Delta b_i] \\ &= \sum_{j=1}^{n} w_{ij} x_j + b_i + \sum_{j=1}^{n} \Delta w_{ij} x_j + \Delta b_i = h_i^{\text{PRE}} + \Delta h_i, \end{aligned}$$

where $w_{ij}/\Delta w_{ij}$ is the entry located at the intersection of the i-th row and j-th column within $\boldsymbol{W}/\Delta\boldsymbol{W}$. Similarly, $b_i/\Delta b_i$ denotes the element positioned at the i-th dimension of $\boldsymbol{b}/\Delta\boldsymbol{b}$. Assuming DARE randomly drops delta parameters with a ratio p and rescales others by a factor of γ. After using DARE, the delta parameters change to $\Delta\widehat{\boldsymbol{W}} \in \mathbb{R}^{m\times n}$ and $\Delta\widehat{\boldsymbol{b}} \in \mathbb{R}^m$. Therefore, the expectation of the i-th dimension of embeddings becomes

$$\begin{aligned} \mathbb{E}[\hat{h}_i] &= \mathbb{E}[\sum_{j=1}^{n} \left(w_{ij} + \Delta\hat{w}_{ij}\right) x_j + \left(b_i + \Delta\hat{b}_i\right)] \\ &= \sum_{j=1}^{n} x_j \mathbb{E}[w_{ij}] + \mathbb{E}[b_i] + \sum_{j=1}^{n} x_j \mathbb{E}[\Delta\hat{w}_{ij}] + \mathbb{E}[\Delta\hat{b}_i] \\ &= \sum_{j=1}^{n} w_{ij} x_j + b_i + \sum_{j=1}^{n} x_j((1-p)\cdot\gamma\cdot\Delta w_{ij} + p\cdot 0) \\ &\quad +((1-p)\cdot\gamma\cdot\Delta b_i + p\cdot 0) \\ &= h_i^{\text{PRE}} + (1-p)\cdot\gamma\cdot(\sum_{j=1}^{n}\Delta w_{ij}x_j + \Delta b_i) \\ &= h_i^{\text{PRE}} + (1-p)\cdot\gamma\cdot\Delta h_i. \end{aligned}$$

By setting $\gamma = 1/(1-p)$, we have $\mathbb{E}[h_i] = \mathbb{E}[\hat{h}_i]$, concluding that DARE can approximate the original embeddings.

Remark. We have given a rough proof of why DARE works. In practice, we find that DARE is applicable when the drop rate p is properly set, and the tolerance of p grows with LMs' parameter sizes. Moreover, removing fine-tuned rather than

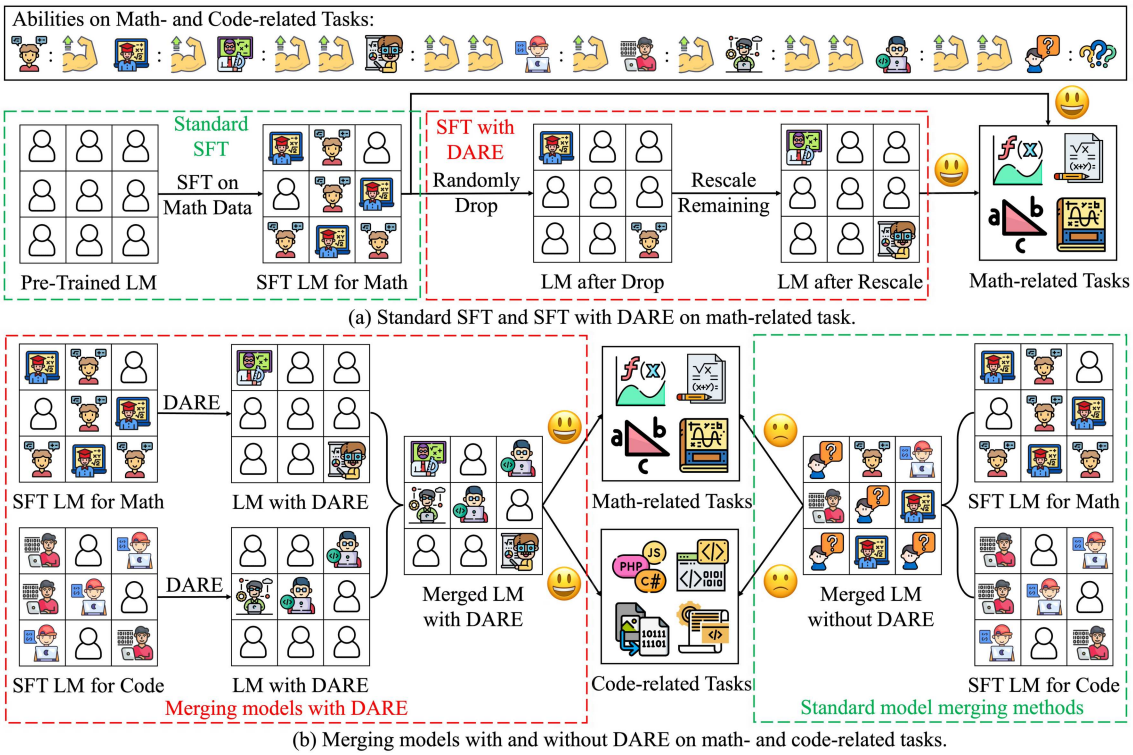

Figure 2: Illustrations of DARE and merging models with DARE. DARE can achieve comparable performance with standard SFT with 90% or even 99% delta parameters removed. Moreover, DARE tackles the parameter interference issue when merging models and yields consistent improvements. At the top, we mark each icon with one or two muscle logos, indicating its ability for specific tasks. For example, the first or second icon has one muscle logo for math-related tasks, while the third or fourth icon can perform better in math with two muscle logos. The rescale operation in DARE multiplies the remaining parameters by $1/(1-p)$, which enhances the task-specific abilities and leads to changes in icons after rescaling.

delta parameters would cause a catastrophically decreased performance. A promising future direction is to explore DARE more deeply, such as inferring the upper bound of p with respect to LM capacities and illustrating the intrinsic difference between fine-tuned and delta parameters.

Last, we highlight the connections and differences between DARE and Dropout (Srivastava et al., 2014). Both methods involve random dropping and rescaling operations, but they differ in two key aspects: (1) DARE handles delta parameters while Dropout operates on model outputs; (2) DARE aims to reduce delta parameter redundancy *without training*, which *permanently* eliminates delta parameters and only retains others for inference. Dropout is used to prevent models from overfitting, which *temporarily* removes part of outputs during training but preserves all the outputs for inference.

3.2. Merging Models with DARE

As DARE effectively reduces the redundancy of delta parameters by setting most of them to zeros, we hypothesize that DARE can help address the interference of parameters when merging multiple models (Yadav et al., 2023). Take Figure 2(b) as an example, when merging math- and code-related models, DARE can assist existing model merging methods to better absorb the abilities of two models with less or no parameter interference.

Formally, given K models that are fine-tuned on K corresponding tasks with parameters $\{\boldsymbol{\theta}^{t_1}_{\text{SFT}}, \boldsymbol{\theta}^{t_2}_{\text{SFT}}, \cdots, \boldsymbol{\theta}^{t_K}_{\text{SFT}}\}$, we first apply DARE on each parameters $\boldsymbol{\theta}^{t_k}_{\text{SFT}}$ $(1 \le k \le K)$, and derive $\{\boldsymbol{\theta}^{t_1}_{\text{DARE}}, \boldsymbol{\theta}^{t_2}_{\text{DARE}}, \cdots, \boldsymbol{\theta}^{t_K}_{\text{DARE}}\}$. Then, we adopt established model merging methods to fuse the derived parameters and obtain the merged single model with parameters $\boldsymbol{\theta}_{\text{M}}$. Let us take Task Arithmetic (Ilharco et al., 2023) as an instance, whose official computation process is denoted by

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

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

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

The expression DARE $\left(\boldsymbol{\theta}_{\text{SFT}}^{t_k}, \boldsymbol{\theta}_{\text{PRE}}, p\right)$ signifies the process of deriving delta parameters from $\boldsymbol{\theta}_{\text{SFT}}^{t_k}$ and $\boldsymbol{\theta}_{\text{PRE}}$, eliminating delta parameters based on drop rate p following Equation (3.1), and finally combining the sparsified delta parameters with $\boldsymbol{\theta}_{\text{PRE}}$ to obtain $\boldsymbol{\theta}_{\text{DARE}}^{t_k}$. In Section 4.3, we find that DARE can effectively improve the performance of Task Arithmetic when merging multiple LMs. It is also worth noticing that DARE is a versatile plug-and-play module and can be applied to any model merging methods, such as Average Merging (Wortsman et al., 2022), Fisher Merging (Matena & Raffel, 2022), RegMean (Jin et al., 2023), and TIES-Merging (Yadav et al., 2023).

4. Experiments

We conduct extensive experiments on encoder- and decoder-based LMs to show the effectiveness of DARE in reducing SFT delta parameter redundancy and merging models.

4.1. Experimental Setup

Datasets and Pre-Trained Backbones for Decoder-based LMs. We choose AlpacaEval (Li et al., 2023) for evaluating instruction-following models (WizardLM (Xu et al., 2023)). We use GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b) for testing mathematical reasoning models (WizardMath (Luo et al., 2023a)). HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) are adopted for estimating code-generating models (WizardCoder-Python (Luo et al., 2023b) and llama-2-13b-code-alpaca (Chaudhary, 2023)). These models are fine-tuned based on pre-trained backbones including LLaMA (Touvron et al., 2023a), Llama 2 (Touvron et al., 2023b), and Code Llama (Rozière et al., 2023). Please see Table 3 in Section A.1 for their versions and correspondences with pre-trained backbones.

Datasets and Pre-Trained Backbones for Encoder-based LMs. For encoder-based LMs, the GLUE benchmark (Wang et al., 2019) is used, containing one sentence acceptability dataset CoLA (Warstadt et al., 2019), one sentiment detection dataset SST-2 (Socher et al., 2013), two paraphrase datasets MRPC (Dolan & Brockett, 2005) and QQP (Shankar et al., 2017), one sentence similarity dataset STS-B (Cer et al., 2017), and three natural language inference datasets MNLI (Bowman et al., 2015; Williams et al., 2018), QNLI (Rajpurkar et al., 2016), and RTE (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009). As the test labels of GLUE are not publicly available, we split the original training data into training and validation sets with ratios of 90% and 10%. The original validation data is used as the test set. We choose bert-base-uncased (Devlin et al., 2019) and roberta-base (Liu et al., 2019a) as pre-trained backbones, and further fine-tune them to get SFT models on the eight datasets.

Evaluation Metrics. We calculate win rate for AlpacaEval, zero-shot accuracy for GSM8K and MATH, pass@1 for HumanEval and MBPP, Matthews correlation coefficient for CoLA, accuracy for SST-2, QNLI, and RTE, matched accuracy for MNLI, accuracy and F1 score for MRPC and QQP, and Pearson and Spearman correlation for STS-B.

Implementation Details. Following Xu et al. (2023); Luo et al. (2023a;b), the inference of decoder-based LMs is implemented by vLLM (Kwon et al., 2023). Temperature is set to 0.0 for greedy decoding. The maximal number of generated tokens is 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. The weight decay is 0.01. We use 1e-5 and 5e-5 as learning rates and list the optimal setting of each fine-tuned model in Table 4 in Section A.2. Experiments are conducted on NVIDIA Tesla V100 and A100 GPUs.

4.2. Extreme Redundancy in SFT Delta Parameters

We show the extremely redundant property of SFT delta parameters of both decoder- and encoder-based LMs. We vary drop rate p in [0.0, 0.1, 0.2, $\cdots$, 0.9, 0.99] and apply DARE to get models after removing the corresponding ratio of delta parameters. When p is equal to 0.0, we actually obtain the standard SFT LMs. We report the performance of decoder-based LMs on GSM8K and HumanEval as well as encoder-based LMs on eight GLUE datasets in Figure 3 and Figure 4. Please see results of decoder-based LMs on AlpacaEval, MATH, and MBPP in Figure 12 in Section B.1.

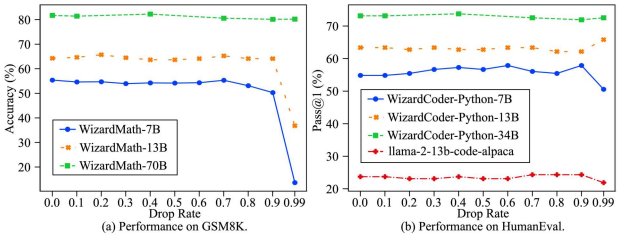

Figure 3: Performance of decoder-based LMs on GSM8K and HumanEval with various drop rates.

We conclude that: (1) *the SFT delta parameters of both encoder- and decoder-based LMs are highly redundant*. DARE can effectively remove 90% delta parameters without significantly decreasing the performance. In some cases,

Table 1: Performance of merging decoder-based WizardLM-13B (LM), WizardMath-13B (Math), and llama-2-13b-code-alpaca (Code) on all the datasets. The best and second-best results are marked in **bold** and underlined fonts.

Merging Methods	Models	Use DARE	Instruction-following	Mathematical Reasoning		Code-generating	
			AlpacaEval	GSM8K	MATH	HumanEval	MBPP
/	LM	No	67.20	2.20	0.04	36.59	34.00
	Math	No	/	64.22	14.02	/	/
	Code	No	/	/	/	23.78	27.60
Task Arithmetic	LM & Math	No	67.04	**66.34**	13.40	28.66	30.60
		Yes	**67.45**	66.26	12.86	26.83	32.40
	LM & Code	No	**68.07**	/	/	31.70	32.40
		Yes	67.83	/	/	**35.98**	**33.00**
	Math & Code	No	/	64.67	13.98	8.54	8.60
		Yes	/	**65.05**	13.96	10.37	9.80
	LM & Math & Code	No	69.03	58.45	9.88	18.29	29.80
		Yes	**69.28**	56.48	10.16	23.17	**31.60**
TIES-Merging	LM & Math	No	68.63	15.77	2.04	**37.80**	35.60
		Yes	**68.70**	36.16	4.56	36.59	**37.00**
	LM & Code	No	63.63	/	/	0.0	0.0
		Yes	67.15	/	/	18.29	26.40
	Math & Code	No	/	63.23	13.56	9.76	22.40
		Yes	/	**64.82**	13.88	10.37	23.60
	LM & Math & Code	No	65.91	62.55	9.54	21.95	30.40
		Yes	**72.50**	58.00	9.20	**29.27**	**31.40**

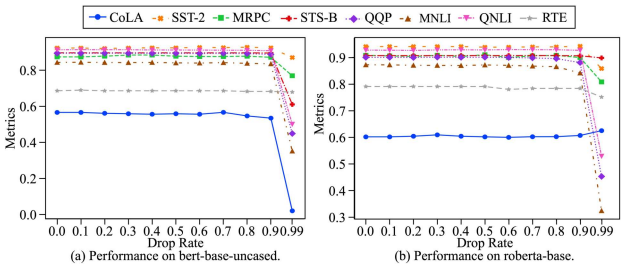

Figure 4: Performance of encoder-based LMs on GLUE with different drop rates.

the drop rate p can even reach 99%; (2) *the tolerance of drop rate increases with the sizes of LMs, i.e., LMs with more parameters can withstand higher drop rate*. For example, WizardMath-70B performs well when $p = 0.99$ while WizardMath-7B and WizardMath-13B fail. This depicts some connections with the scaling laws of LMs (Kaplan et al., 2020; Hoffmann et al., 2022), indicating that there may exist quantifiable correlations between model sizes and drop rates they can afford.

4.3. Merging Models with DARE on SFT LMs

We combine DARE with five model merging methods, including Average Merging (Wortsman et al., 2022), Task Arithmetic (Ilharco et al., 2023), Fisher Merging (Matena & Raffel, 2022), RegMean (Jin et al., 2023), and TIES-Merging (Yadav et al., 2023). Please see Section A.3 for more descriptions of the methods. For feasible computations, we merge decoder-based LMs based on Task Arithmetic and TIES-Merging. The scaling term in both methods is chosen from [0.5, 1.0], and the retain ratio of largest-magnitude parameters in TIES-Merging is selected from [0.5, 0.7, 0.9]. We merge WizardLM-13B, WizardMath-13B, and llama-2-13b-code-alpaca since all of them adopt Llama-2-13b as the pre-trained backbone. WizardCoder-Python-13B is not selected as it is fine-tuned from CodeLlama-13b-Python. We merge encoder-based LMs with all five methods and perform grid search on some hyperparameters (see Table 5 in Section A.4 for more details). Following Jin et al. (2023); Yadav et al. (2023), we also fine-tune the models under the multi-task learning setting and report the oracle results. We show the performance of merging decoder-based LMs in Table 1 and present partial results of merging encoder-based LMs in Figure 5. Please refer to Figure 13 in Section B.2 for the complete results.

From Table 1, we find that: 1) DARE often facilitates Task Arithmetic and TIES-Merging on merging decoder-based LMs, which even yields better results than the source model in many cases, offering a novel discovery unobserved in previous works. For instance, the improvements brought by Task Arithmetic with DARE are 3.10% for LM & Math & Code vs. LM on AlpacaEval, 3.18% for LM & Math vs. Math on GSM8K, and 19.57% for LM & Code vs. Code on MBPP; 2) Compared with Task Arithmetic, TIES-Merging tends to benefit more from DARE. This is because TIES-Merging first eliminates delta parameters with lower

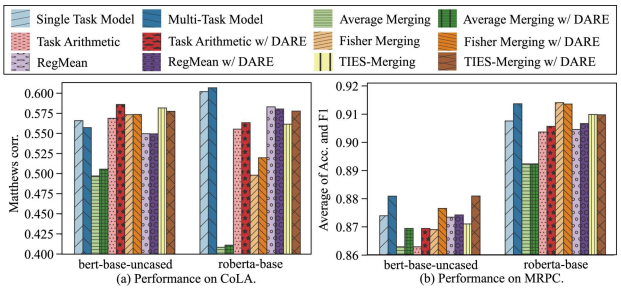

Figure 5: Performance of merging encoder-based bert-base-uncased and roberta-base on CoLA and MRPC.

magnitudes for each model, which potentially decreases the performance. When using DARE, delta parameters can be effectively removed by resetting them to zeros without adversely affecting the performance. Thus, TIES-Merging just drops delta parameters sparsified by DARE (with zero as the smallest magnitude), avoiding performance reduction in the first step; 3) It seems that llama-2-13b-code-alpaca is not well fine-tuned for generating codes since it performs worse than WizardLM-13B, which may affect the model merging performance. We additionally evaluate the code-generating ability of the merger of WizardLM-13B and WizardMath-13B, which obtains better results than llama-2-13b-code-alpaca, explaining the suboptimal performance of the amalgamation of WizardMath-13B and llama-2-13b-code-alpaca. Therefore, an essential prerequisite for effective model merging is that each source model to be merged should be well fine-tuned.

From Figure 5, we observe that DARE often yields 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. However, the merged model still struggles to surpass the single model in some cases, which is in line with the conclusions in Matena & Raffel (2022); Jin et al. (2023); Yadav et al. (2023).

Last but not least, from both Table 1 and Figure 5, we further conclude that the improvements caused by DARE are more pronounced in decoder-based LMs compared to encoder-based LMs. One possible reason is that decoder-based LMs are able to accommodate more abilities than encoder-based LMs due to their substantially larger sizes.

We further verify the effectiveness of DARE in merging decoder-based LMs apart from the Llama 2 backbone (e.g., Mistral-7B (Jiang et al., 2023)). We provide two merged decoder-based LMs with 7 billion parameters (namely, supermario_v1 and supermario_v2) and evaluate them on Open LLM Leaderboard (Beeching et al., 2023). Please see Section A.5 for more details of the source models and benchmarks. From Table 2, we find that the merged LMs beat

Table 2: Results of 7B LMs on the Open LLM Leaderboard.

Models	Average	ARC	Hella.	MMLU	TQA	Wino.	GSM8K
NeuralBeagle14-7B	74.74	72.95	88.34	64.55	69.93	82.40	70.28
Beagle14-7B	74.76	72.95	87.95	64.70	68.88	82.64	71.42
supermario_v1	**74.85**	73.72	88.71	64.57	68.23	85.64	68.23
WildMarcoroni-7B	75.29	73.98	88.61	64.81	69.76	84.29	70.28
WestSeverus-7B	75.29	71.42	88.27	64.79	72.37	83.27	71.65
supermario_v2	**75.49**	72.95	88.53	64.99	71.22	83.90	71.34

the source models they are built upon, achieving a certain degree of improvement. Notably, until January 28th, 2024, *supermario_v2 achieves the first rank on the Open LLM Leaderboard*. It is thrilling that these benefits can be cheaply acquired by merely utilizing CPUs.

4.4. Importance of the Rescale Operation

As analyzed in Section 3.1, the rescale operation in DARE is essential to approximate the original embeddings. To verify this, we introduce DropOnly which randomly drops delta parameters without rescaling. We calculate the similarities of embeddings between the original LM and LM with DARE or DropOnly. Specifically, we obtain the embeddings of each input token layer-by-layer and report the average cosine similarities. Results of WizardMath-7B on GSM8K and bert-base-uncased on CoLA are shown in Figure 6.

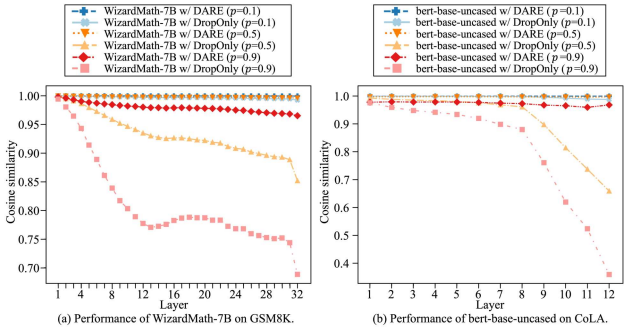

Figure 6: Cosine similarities of each layer's embeddings between the original LM and LM with DARE or DropOnly.

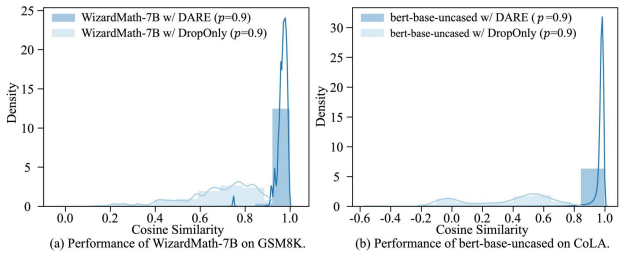

Figure 7: Distributions of cosine similarities of the last layer's embeddings between the original LM and LM with DARE or DropOnly.

We observe that DARE can perfectly maintain the original embeddings in each layer with similarities higher than 0.95 even when removing 90% delta parameters. However, DropOnly just preserves the original embeddings with $p = 0.1$ and the similarities sharply decline when p is higher. For example, the similarities on WizardMath-7B decrease to about 0.85/0.68 when p is 0.5/0.9). We further show the distributions of embeddings' cosine similarities in the last layer in Figure 7, demonstrating the ability of DARE in approximating original embeddings. Note that similar findings can be obtained on other LMs and datasets but they are not presented due to page limits.

We also report the performance of LMs with DARE and DropOnly in Figure 8. See Figure 14 and Figure 15 in Section B.3 for additional results. We observe that discarding the rescale operation usually leads to worse results, and the performance gaps between DARE and DropOnly become more significant with the increase of p. This validates the effectiveness of the rescale operation in DARE once again.

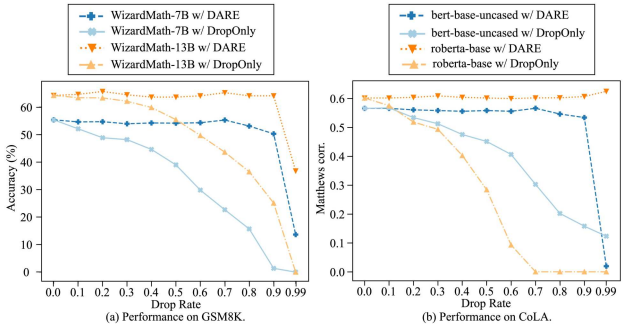

Figure 8: Comparisons between DARE and DropOnly on GSM8K and CoLA on various LMs.

4.5. Comparison with Magnitude-based Pruning

We compare DARE with the commonly used Magnitude-based Pruning (MP) (Han et al., 2015; Li et al., 2018; Lee et al., 2021), which chooses parameters based on their magnitudes. For more fair and credible comparisons, we adapt MP to operate on delta parameters and discard the retraining process. We show partial results of LMs with DARE and MP in Figure 9. Please refer to Figure 16 and Figure 17 in Section B.4 for extra 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 superiority of DARE in abandoning delta parameters. The reason is that MP fails to preserve the original embeddings since it neglects the contributions of delta parameters with lower magnitudes. We have also tried to combine MP with the rescale operation but got worse results than using MP separately. For example, when

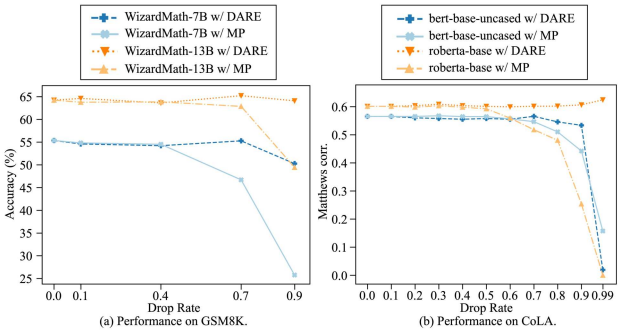

Figure 9: Comparisons between DARE and MP on GSM8K and CoLA on various LMs.

the drop rate is 0.7, the performance of MP on 7B LMs decreases from 43.85 to 10.61 on AlpacaEval, from 46.70 to 0.37 on GSM8K, and from 21.34 to 3.05 on HumanEval. This is because MP removes parameters with smaller magnitudes and retains certain parameters with larger magnitudes. Simply rescaling the remaining parameters would result in unpredictable performance.

4.6. When Can DARE Be Used?

We 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 metric on HumanEval/MBPP drastically decreases 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-related data (Rozière et al., 2023), 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 infeasible. To verify this, we depict the *absolute values* of SFT delta parameters of 13B decoder-based LMs vs. various pre-trained 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. Additionally, we present the statistics on the percentiles of delta parameter ranges of both decoder- and encoder-based LMs in Table 6 in Section B.5.

From the results, we observe the absolute values of delta parameters of WizardCoder-Python-13B vs. Llama-2-13b (often greater than 0.01) are several orders of magnitude bigger than those of WizardCoder-Python-13B vs. CodeLlama-13b-Python (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 parame-

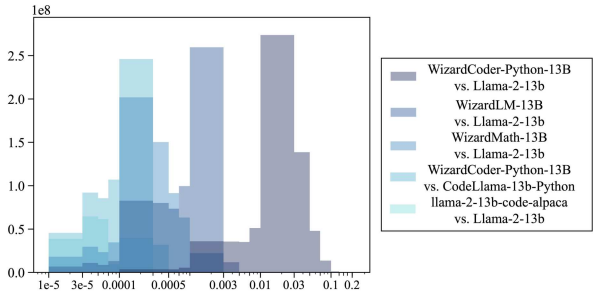

Figure 10: Delta parameter absolute values of 13B decoder-based LMs vs. the pre-trained backbones.

ters are less than 0.002, 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., within 0.002). Otherwise, DARE may fail.

4.7. Can DARE Drop Fine-tuned Parameters?

As previous network pruning methods mainly operate on the fine-tuned instead of delta parameters, we also conduct experiments under this setting with both decoder- and . For decoder-based LMs, we find they perform badly when removing fine-tuned parameters even with 0.1 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. Similar observations can also be found on MP or decoder-based LMs with 7B, 34B, or 70B sizes. Partial results on encoder-based LMs are shown in Figure 11 and please see Figure 21 in Section B.6 for additional results. We observe that directly

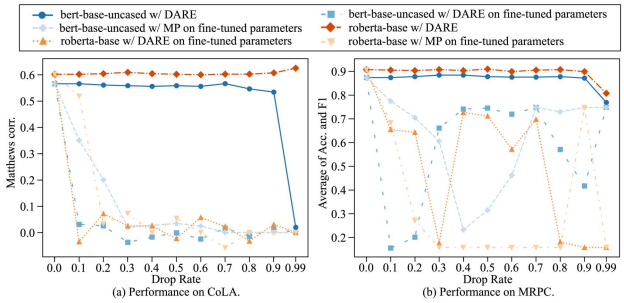

Figure 11: Results of DARE and MP by dropping fine-tuned parameters on CoLA and MRPC on encoder-based LMs.

eliminating the fine-tuned parameters by either DARE or MP would lead to worse performance on encoder-based LMs. The above results confirm that the knowledge is inherent in pre-trained LMs, and SFT is responsible for unlocking instead of introducing new capabilities. Moreover, decoder-based LMs are more susceptible than encoder-based LMs when removing fine-tuned parameters. This could be attributed to the fact that decoder-based LMs exhibit a higher degree of capability and have a stronger correlation with the fine-tuned parameters. Consequently, even the removal of a relatively small proportion of fine-tuned parameters can significantly degrade their performance.

5. Conclusion

In this work, we first 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 any data, retraining, or even GPUs. DARE can impressively drop 90% or even 99% SFT delta parameters without sacrificing much performance compared with using all SFT delta parameters. We further employed DARE as a versatile plug-and-play approach 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 encoder- and decoder-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 that our findings can advance the understanding of model alignment from the perspective of analyzing model parameters.

Impact Statement

Recently, merging language models has become a promising research direction. Our work allows researchers to obtain a single model with diverse capabilities at a low cost. Thanks to our method, *hundreds of models with different functionalities have been created on the Hugging Face community*[1]. *Several popular toolkits on the GitHub platform have also integrated our work, including huggingface/peft* [2] *and arcee-ai/mergekit*[3]. Even though this work has no direct social impacts, the potentially harmful information generated by LLMs (e.g., gender bias, racial discrimination) may still exist when using our approach. It is necessary to advocate for careful regulation by the communities as well as authorities on this matter.

Acknowledgements

We would like to express our sincere gratitude to the anonymous reviewers for their insightful comments and suggestions, which have significantly enriched this paper.

[1] https://huggingface.co/models?other=arxiv:2311.03099

[2] https://github.com/huggingface/peft

[3] https://github.com/arcee-ai/mergekit

References

Austin, J., Odena, A., Nye, M. I., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C. J., Terry, M., Le, Q. V., and Sutton, C. Program synthesis with large language models. *CoRR*, abs/2108.07732, 2021.

Beeching, E., Fourrier, C., Habib, N., Han, S., Lambert, N., Rajani, N., Sanseviero, O., Tunstall, L., and Wolf, T. Open llm leaderboard, 2023.

Bentivogli, L., Clark, P., Dagan, I., and Giampiccolo, D. The fifth pascal recognizing textual entailment challenge. *TAC*, 7:8, 2009.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642. The Association for Computational Linguistics, 2015.

Cer, D. M., Diab, M. T., Agirre, E., Lopez-Gazpio, I., and Specia, L. Semeval-2017 task 1: Semantic textual similarity - multilingual and cross-lingual focused evaluation. *CoRR*, abs/1708.00055, 2017.

Chaudhary, S. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca, 2023.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code. *CoRR*, abs/2107.03374, 2021.

Cheng, Y., Wang, D., Zhou, P., and Zhang, T. A survey of model compression and acceleration for deep neural networks. *CoRR*, abs/1710.09282, 2017.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.

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

Dagan, I., Glickman, O., and Magnini, B. The PASCAL recognising textual entailment challenge. In Candela, J. Q., Dagan, I., Magnini, B., and d'Alché-Buc, F. (eds.), *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*, pp. 177–190. Springer, 2005.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. 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*, pp. 4171–4186. Association for Computational Linguistics, 2019.

Ding, N., Qin, Y., Yang, G., Wei, F., Yang, Z., Su, Y., Hu, S., Chen, Y., Chan, C., Chen, W., Yi, J., Zhao, W., Wang, X., Liu, Z., Zheng, H., Chen, J., Liu, Y., Tang, J., Li, J., and Sun, M. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nat. Mac. Intell.*, 5(3):220–235, 2023.

Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., and Smith, N. A. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *CoRR*, abs/2002.06305, 2020.

Dolan, W. B. and Brockett, C. 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.

Fisher, R. A. 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.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *7th International Conference on Learning Representations*. OpenReview.net, 2019.

Gale, T., Elsen, E., and Hooker, S. The state of sparsity in deep neural networks. *CoRR*, abs/1902.09574, 2019.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L.,

Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, 12 2023.

Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, W. B. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pp. 1–9, 2007.

Haim, R. B., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B., and Szpektor, I. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, volume 7, pp. 785–794, 2006.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations*. OpenReview.net, 2021a.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1*, 2021b.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. Training compute-optimal large language models. *CoRR*, abs/2203.15556, 2022.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 2019.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations*. OpenReview.net, 2022.

Ilharco, G., Ribeiro, M. T., Wortsman, M., Schmidt, L., Hajishirzi, H., and Farhadi, A. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*. OpenReview.net, 2023.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de Las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b. *CoRR*, abs/2310.06825, 2023.

Jin, X., Ren, X., Preotiuc-Pietro, D., and Cheng, P. Dataless knowledge fusion by merging weights of language models. In *The Eleventh International Conference on Learning Representations*. OpenReview.net, 2023.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626. ACM, 2023.

Lee, J., Park, S., Mo, S., Ahn, S., and Shin, J. Layer-adaptive sparsity for the magnitude-based pruning. In *9th International Conference on Learning Representations*. OpenReview.net, 2021.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059. Association for Computational Linguistics, 2021.

Li, G., Qian, C., Jiang, C., Lu, X., and Tang, K. Optimization based layer-wise magnitude-based pruning for DNN compression. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 2383–2389. ijcai.org, 2018.

Li, X., Zhang, T., Dubois, Y., Taori, R., Gulrajani, I., Guestrin, C., Liang, P., and Hashimoto, T. B. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 2023.

Li, X. L. and Liang, P. 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*, pp. 4582–4597. Association for Computational Linguistics, 2021.

Liang, T., Glossner, J., Wang, L., Shi, S., and Zhang, X. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

Lin, S., Hilton, J., and Evans, O. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pp. 3214–3252. Association for Computational Linguistics, 2022.

Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., and Tang, J. GPT understands, too. *CoRR*, abs/2103.10385, 2021.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019a.

Liu, Z., Sun, M., Zhou, T., Huang, G., and Darrell, T. Rethinking the value of network pruning. In *7th International Conference on Learning Representations*. OpenReview.net, 2019b.

Luo, H., Sun, Q., Xu, C., Zhao, P., Lou, J., Tao, C., Geng, X., Lin, Q., Chen, S., and Zhang, D. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *CoRR*, abs/2308.09583, 2023a.

Luo, Z., Xu, C., Zhao, P., Sun, Q., Geng, X., Hu, W., Tao, C., Ma, J., Lin, Q., and Jiang, D. Wizardcoder: Empowering code large language models with evol-instruct. *CoRR*, abs/2306.08568, 2023b.

Matena, M. and Raffel, C. Merging models with fisher-weighted averaging. In *Advances in Neural Information Processing Systems*, 2022.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392. The Association for Computational Linguistics, 2016.

Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Remez, T., Rapin, J., Kozhevnikov, A., Evtimov, I., Bitton, J., Bhatt, M., Canton-Ferrer, C., Grattafiori, A., Xiong, W., Défossez, A., Copet, J., Azhar, F., Touvron, H., Martin, L., Usunier, N., Scialom, T., and Synnaeve, G. Code llama: Open foundation models for code. *CoRR*, abs/2308.12950, 2023.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pp. 8732–8740. AAAI Press, 2020.

Shankar, I., Nikhil, D., and Kornel, C. First quora dataset release: question pairs (2017). *URL https://www. quora. com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs*, 2017.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642. ACL, 2013.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 2014.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Canton-Ferrer, C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023b.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations*. OpenReview.net, 2019.

Warstadt, A., Singh, A., and Bowman, S. R. Neural network acceptability judgments. *Trans. Assoc. Comput. Linguistics*, 7:625–641, 2019.

Williams, A., Nangia, N., and Bowman, S. R. A broad-coverage challenge corpus for sentence understanding through inference. In Walker, M. A., Ji, H., and Stent, A. (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, pp. 1112–1122. Association for Computational Linguistics, 2018.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Lopes, R. G., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., and Schmidt, L. 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*, pp. 23965–23998. PMLR, 2022.

Xia, M., Zhong, Z., and Chen, D. Structured pruning learns compact and accurate models. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1513–1528. Association for Computational Linguistics, 2022.

Xu, C., Sun, Q., Zheng, K., Geng, X., Zhao, P., Feng, J., Tao, C., and Jiang, D. Wizardlm: Empowering large language models to follow complex instructions. *CoRR*, abs/2304.12244, 2023.

Yadav, P., Tam, D., Choshen, L., Raffel, C., and Bansal, M. Resolving interference when merging models. In *Advances in Neural Information Processing Systems*, 2023.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pp. 4791–4800. Association for Computational Linguistics, 2019.

Zhang, J., Chen, S., Liu, J., and He, J. Composing parameter-efficient modules with arithmetic operations. In *Advances in Neural Information Processing Systems*, 2023.

Zhang, Y. and Yang, Q. A survey on multi-task learning. *IEEE Trans. Knowl. Data Eng.*, 34(12):5586–5609, 2022.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J., and Wen, J. A survey of large language models. *CoRR*, abs/2303.18223, 2023.

Zhu, M. and Gupta, S. 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 3 shows the versions and correspondences with pre-trained backbones of SFT decoder-based LMs.

Table 3: Versions and correspondences with pre-trained backbones of SFT decoder-based LMs.

Tasks	SFT Decoder-based LMs	Pre-Trained Backbones
Instruction-following	WizardLM-7B[4]	llama-7b[5]
	WizardLM-13B[6]	Llama-2-13b[7]
	WizardLM-70B[8]	Llama-2-70b[9]
Mathematical Reasoning	WizardMath-7B[10]	Llama-2-7b[11]
	WizardMath-13B[12]	Llama-2-13b[7]
	WizardMath-70B[13]	Llama-2-70b[9]
Code-generating	WizardCoder-Python-7B[14]	CodeLlama-7b-Python[15]
	WizardCoder-Python-13B[16]	CodeLlama-13b-Python[17]
	WizardCoder-Python-34B[18]	CodeLlama-34b-Python[19]
	llama-2-13b-code-alpaca[20]	Llama-2-13b[7]

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

The optimal settings of the learning rate of each fine-tuned encoder-based LM are presented in Table 4.

Table 4: 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[21]	5e-5	1e-5	5e-5	5e-5	1e-5	1e-5	1e-5	1e-5
roberta-base[22]	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 experiment with five model merging methods:

- **Average Merging** simply averages the parameters of multiple models to get the merged model (Wortsman et al., 2022).

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

- **Task Arithmetic** uses a scaling term to control the contributions between the pre-trained backbone and the models to be merged (Ilharco et al., 2023).

- **Fisher Merging** first estimates the importance of parameters by calculating the Fisher information matrix, and then fuses parameters based on their importance (Matena & Raffel, 2022).

- **RegMean** recasts the model merging task as a linear regression problem and derives closed-form solutions to solve the problem (Jin et al., 2023).

- **TIES-Merging** aims to address parameter conflicts in model merging. It first trims parameters with lower magnitudes, and then resolves sign disagreements. Parameters with consistent signs are finally merged (Yadav et al., 2023).

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

Table 5 shows the searched ranges of model merging methods' hyperparameters for encoder-based LMs. For DARE, we search the drop rate p in [0.1, 0.2, $\cdots$, 0.9] and select the optimal setting with the best performance.

Table 5: Searched ranges of hyperparameters 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]

A.5. Details of Our Merged 7B LMs and the Open LLM Leaderboard

We offer two merged LMs with 7 billion parameters, namely supermario_v1 and supermario_v2. Specifically, we choose NeuralBeagle14-7B[23] and Turdus[24] to build supermario_v1, where both of them all derived from Beagle14-7B[25]. We set the drop rate p in DARE to 0.3, and merge NeuralBeagle14-7B and Turdus by Task Arithmetic with 0.8 as the scaling term. We select WildMarcoroni-Variant1-7B[26] and WestSeverus-7B-DPO-v2[27] to obtain supermario_v2, where both of them adopt Mistral-7B-v0.1[28] (Jiang et al., 2023) as the backbone. The drop rate p in DARE is set to 0.5, and the scaling term in Task Arithmetic is also 0.5.

The Open LLM Leaderboard[29] is established to evaluate open-sourced LLMs based on Eleuther AI Language Model Evaluation Harness (Gao et al., 2023), which contains six benchmarks including AI2 Reasoning Challenge (ARC) (Clark et al., 2018), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021a), TruthfulQA (Lin et al., 2022), Winogrande (Sakaguchi et al., 2020), and GSM8K (Cobbe et al., 2021). The average score on the six datasets is used for ranking models on the leaderboard. We refer interested readers to the original papers for detailed information on the datasets.

Note that the results of Turdus on Open LLM Leaderboard are not available and we instead report the performance of Beagle14-7B in Table 2. Moreover, due to space limits, we use Hella., TQA, and Wino. as the abbreviations for HellaSwag, TruthfulQA, and Winogrande. WildMarcoroni-7B and WestSeverus-7B are the abbreviations for WildMarcoroni-Variant1-7B and WestSeverus-7B-DPO-v2.

[23] https://huggingface.co/mlabonne/NeuralBeagle14-7B
[24] https://huggingface.co/udkai/Turdus
[25] https://huggingface.co/mlabonne/Beagle14-7B
[26] https://huggingface.co/BarryFutureman/WildMarcoroni-Variant1-7B
[27] https://huggingface.co/FelixChao/WestSeverus-7B-DPO-v2
[28] https://huggingface.co/mistralai/Mistral-7B-v0.1
[29] https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

B. Additional Experimental Results

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

Figure 12 shows results of decoder-based LMs on AlpacaEval, MATH, and MBPP with different drop rates. We notice that the performance of WizardLM-70B drastically declines on AlpacaEval when the drop rate is 0.9 (different from the observations of WizardMath-70B and WizardCoder-Python-34B). One possible reason is that the instruction-following task on AlpacaEval is harder 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.

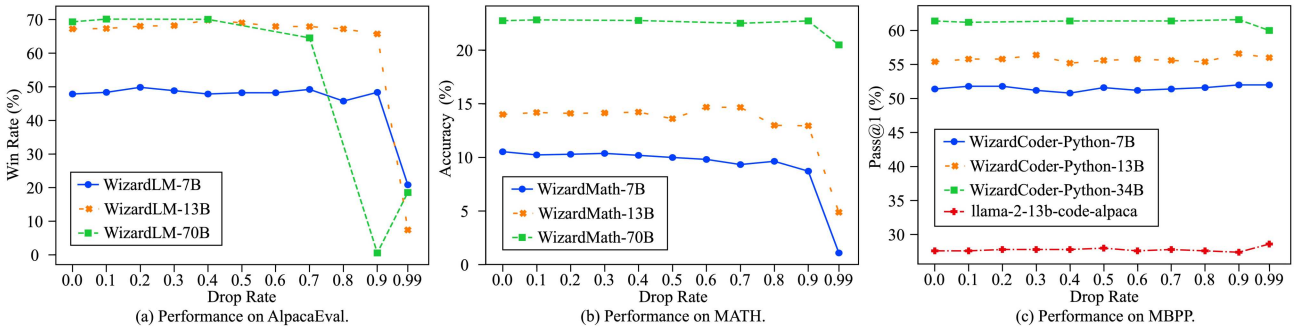

Figure 12: Performance of decoder-based LMs on AlpacaEval, MATH, and MBPP with various drop rates.

B.2. Additional Results of Merging Encoder-based LMs

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

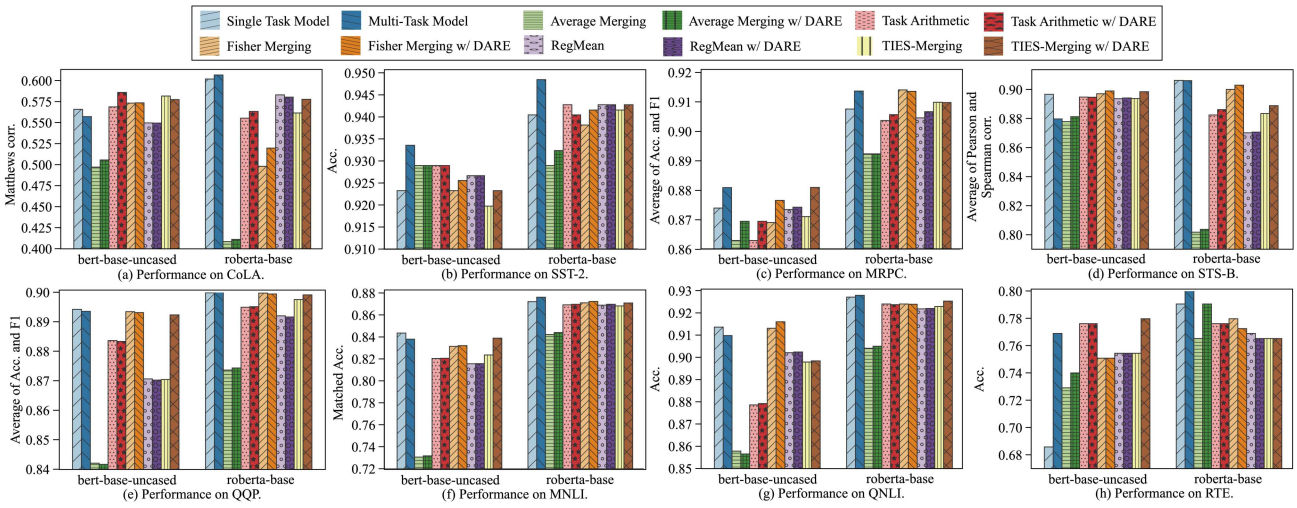

Figure 13: Performance of merging encoder-based LMs on GLUE.

B.3. Additional Results of Comparisons between DARE and DropOnly

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

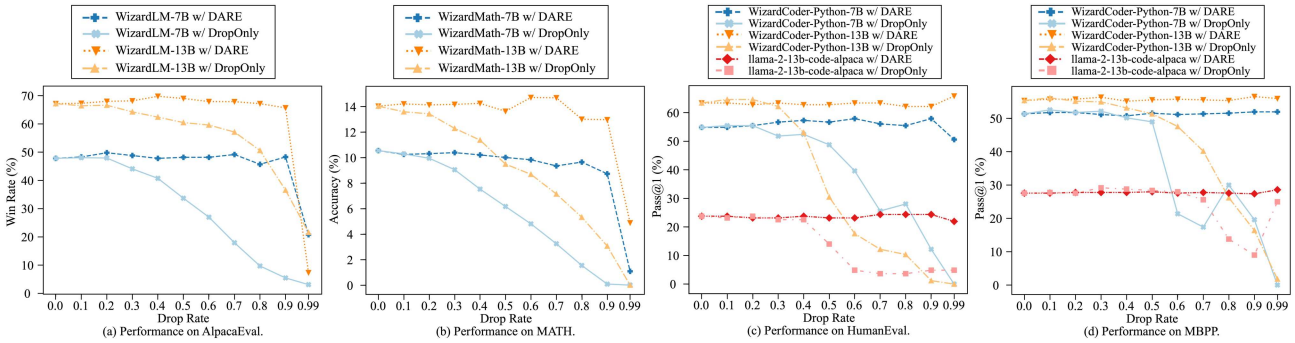

Figure 14: Comparing DARE and DropOnly on AlpacaEval, MATH, HumanEval, and MBPP on decoder-based LMs.

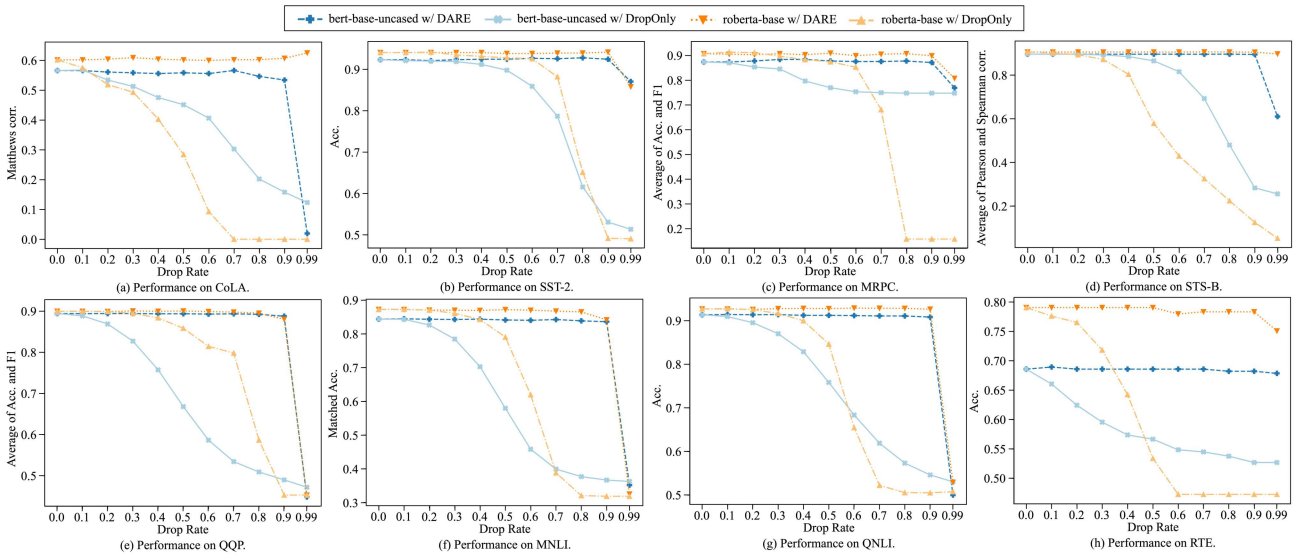

Figure 15: Comparisons between DARE and DropOnly on GLUE on encoder-based LMs.

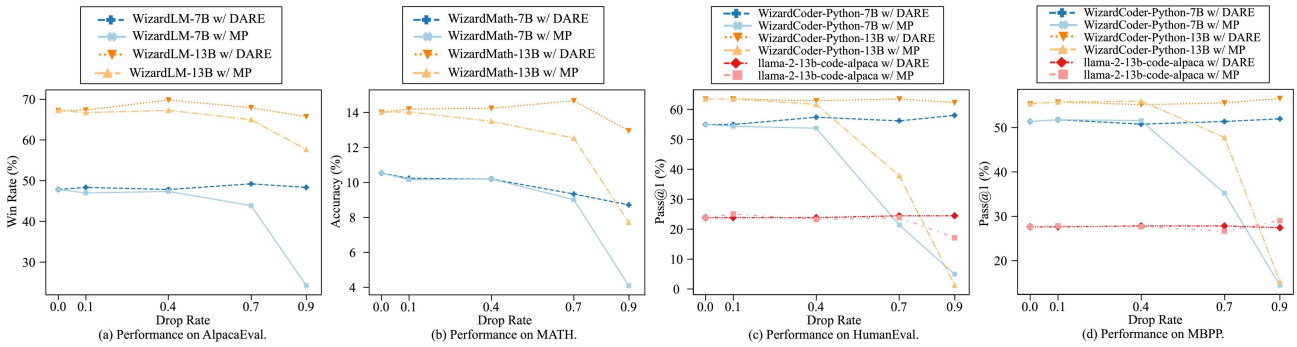

Figure 16: Comparisons between DARE and MP on AlpacaEval, MATH, HumanEval, and MBPP on decoder-based LMs.

B.4. Additional Results of Comparisons between DARE and MP

Comparisons between DARE and magnitude-based pruning on AlpacaEval, MATH, HumanEval, and MBPP on decoder-based LMs and all results on GLUE on encoder-based LMs are shown in Figure 16 and Figure 17, respectively.

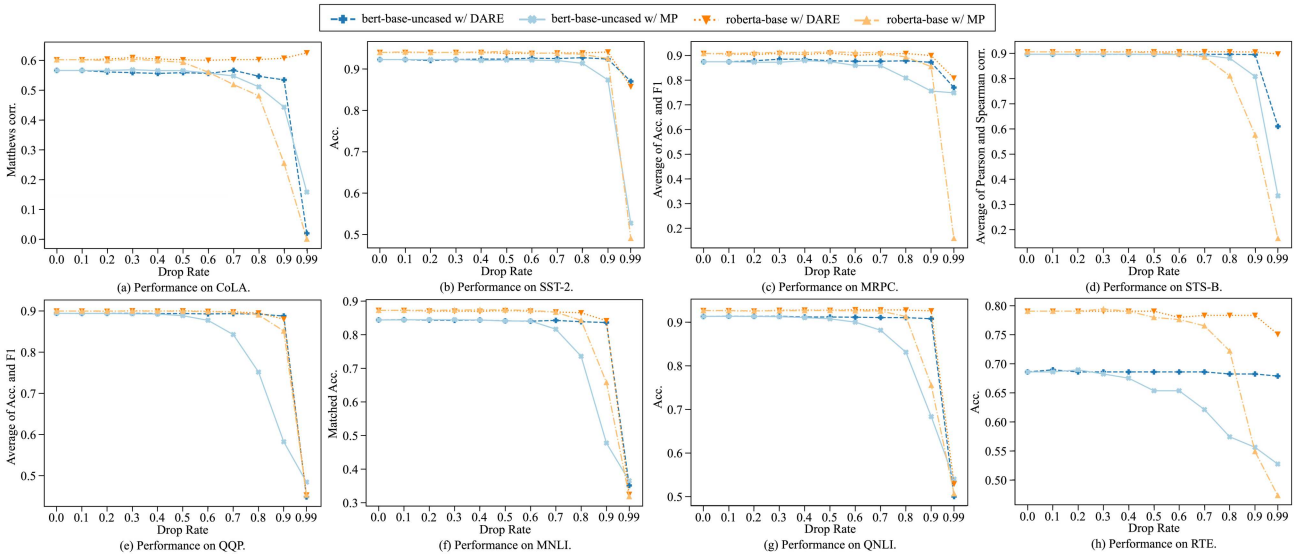

Figure 17: Comparisons between DARE and MP on GLUE on encoder-based LMs.

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. Note that for decoder-based LMs, the results are obtained by randomly selecting 10% delta parameters, whereas for encoder-based LMs, all delta parameters are included. We also provide the statistics on the percentiles of delta parameter ranges in Table 6, which are derived by sorting the entire ranges and indexing at positions corresponding to 0, 10%, 20%, ..., 100%.

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.

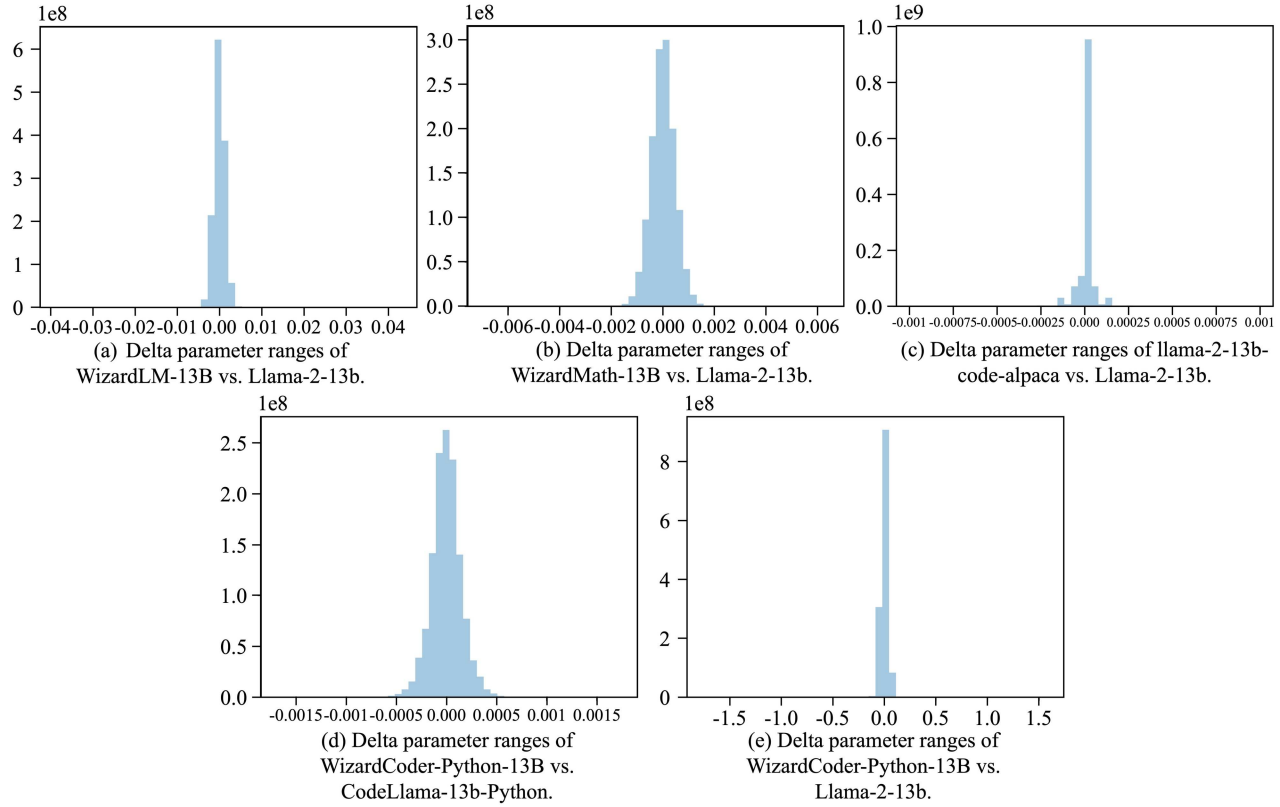

Figure 18: Delta parameter ranges of 13B decoder-based LMs vs. the pre-trained backbones.

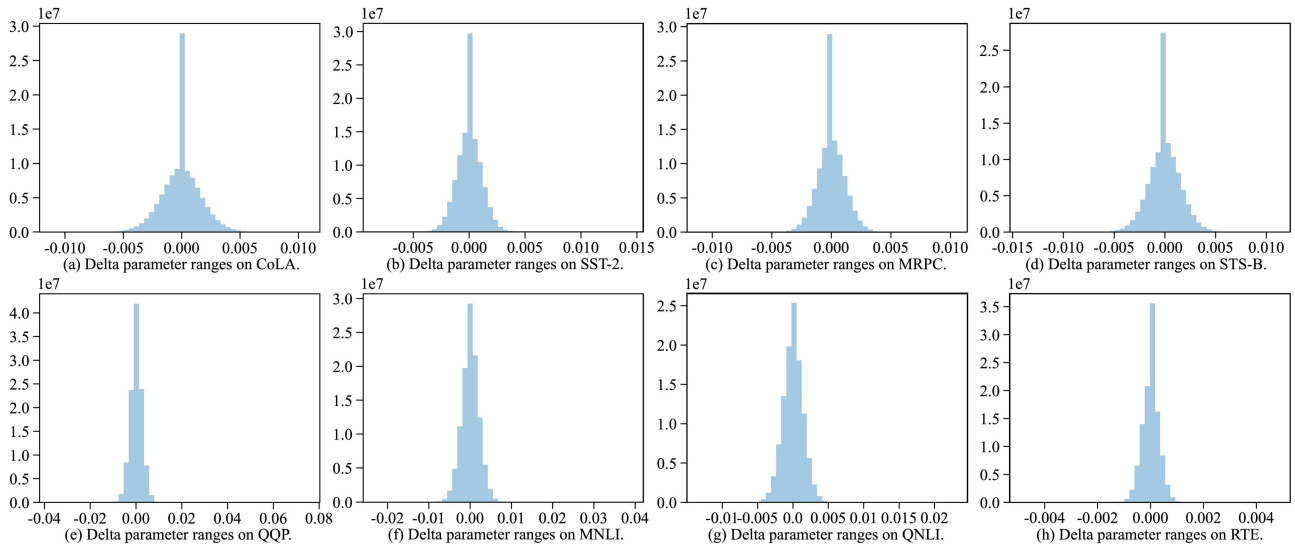

Figure 19: Delta parameter ranges of bert-base-uncased after SFT on GLUE.

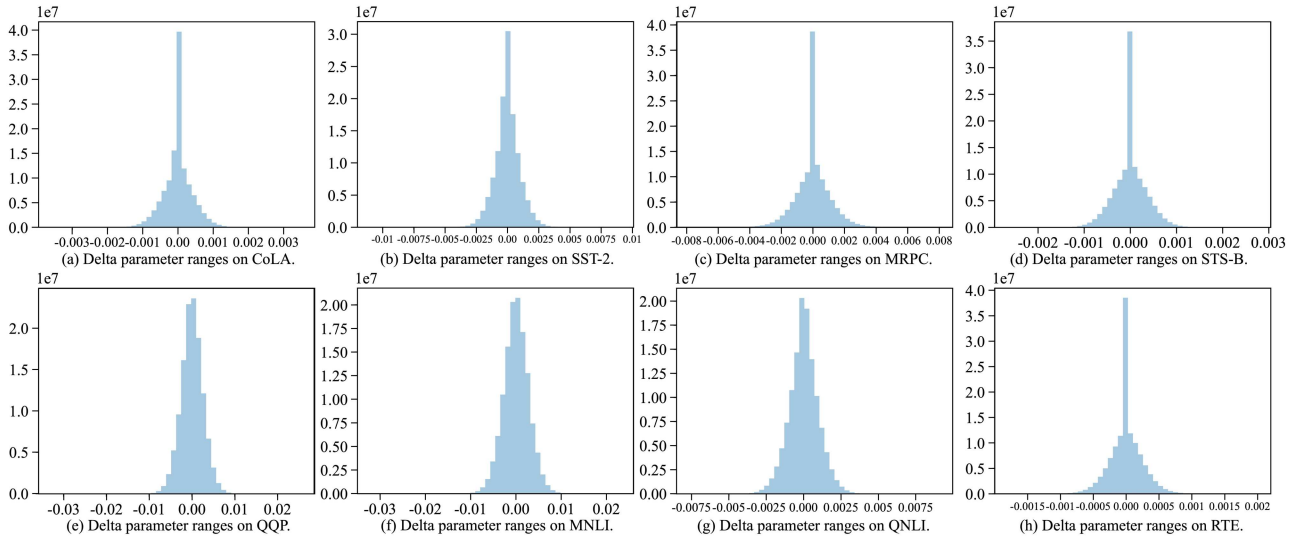

Figure 20: Delta parameter ranges of roberta-base after SFT on GLUE.

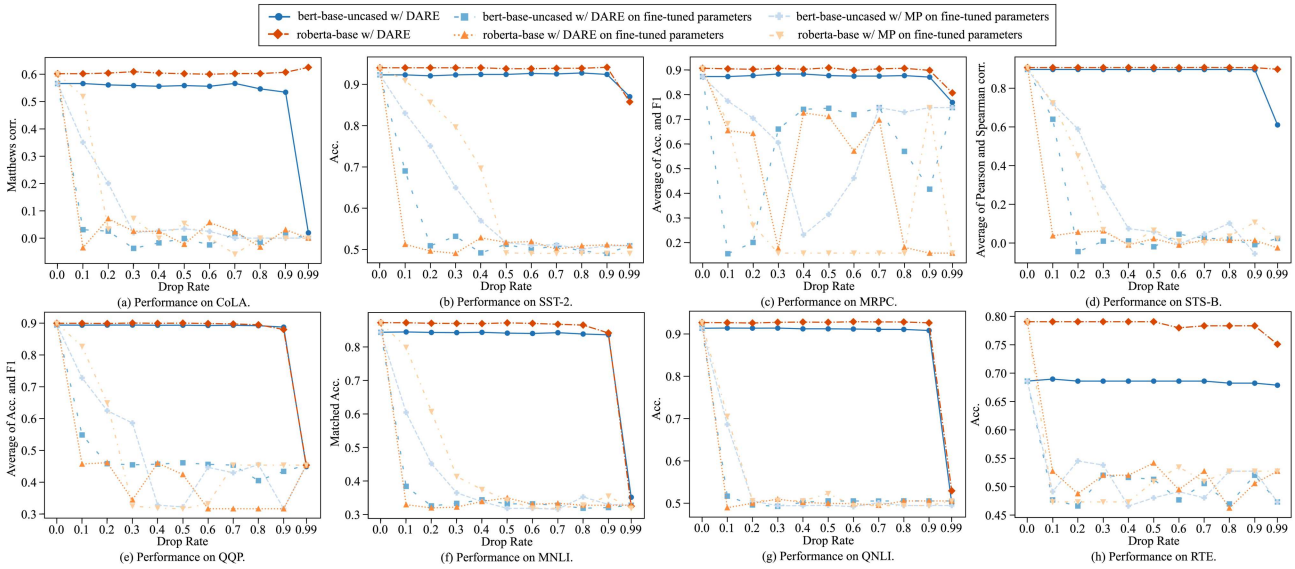

Figure 21: Performance of DARE and MP when dropping fine-tuned parameters on GLUE on encoder-based LMs.

Table 6: Statistics about the deciles of delta parameter ranges of both decoder- and encoder-based LMs.

Models		0% (min)	10%	20%	30%	40%	50%	60%	70%	80%	90%	100% (max)
WizardLM-13B vs. Llama-2-13b		-3.93e-02	-0.16e-02	-0.10e-02	-0.06e-02	-0.03e-02	0.00	0.03e-02	0.06e-02	0.10e-02	0.16e-02	4.81e-02
WizardMath-13B vs. Llama-2-13b		-0.69e-02	-0.06e-02	-0.04e-02	-0.02e-02	-0.01e-02	0.00	0.01e-02	0.02e-02	0.04e-02	0.06e-02	0.74e-02
llama-2-13b-code-alpaca vs. Llama-2-13b		-8.42e-02	-3.05e-05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3.05e-05	7.98e-02
WizardCoder-Python-13B vs. CodeLlama-13b-Python		-1.86e-03	-1.81e-04	-1.07e-04	-6.87e-05	-3.34e-05	0.00	3.34e-05	6.87e-05	1.07e-04	1.81e-04	1.82e-03
WizardCoder-Python-13B vs. Llama-2-13b		-2.40	-3.81e-02	-2.47e-02	-1.53e-02	-7.31e-03	7.63e-06	7.32e-03	1.53e-02	2.47e-02	3.81e-02	2.40
bert-base-uncased	CoLA	-1.10e-02	-1.99e-03	-1.16e-03	-5.70e-04	-7.35e-05	3.08e-05	1.07e-04	5.74e-04	1.17e-03	1.99e-03	1.08e-02
	SST-2	-8.33e-03	-1.33e-03	-8.09e-04	-4.29e-04	-1.11e-04	4.81e-05	1.93e-04	4.42e-04	8.21e-04	1.34e-03	1.44e-02
	MRPC	-1.10e-02	-1.41e-03	-8.45e-04	-4.49e-04	-1.01e-04	8.55e-06	1.06e-04	4.54e-04	8.50e-04	1.41e-03	1.03e-02
	STS-B	-1.40e-02	-1.88e-03	-1.13e-03	-5.93e-04	-1.27e-04	2.11e-05	1.27e-04	5.93e-04	1.13e-03	1.88e-03	1.11e-02
	QQP	-3.66e-02	-0.32e-02	-0.19e-02	-0.11e-02	-0.04e-02	0.01e-02	0.06e-02	0.13e-02	0.21e-02	0.32e-02	7.47e-02
	MNLI	-2.28e-02	-2.61e-03	-1.63e-03	-9.51e-04	-3.90e-04	7.63e-05	4.81e-04	1.03e-03	1.70e-03	2.65e-03	3.87e-02
	QNLI	-1.32e-02	-1.77e-03	-1.11e-03	-6.57e-04	-2.76e-04	4.43e-05	3.31e-04	7.08e-04	1.16e-03	1.79e-03	2.29e-02
	RTE	-4.86e-03	-3.94e-04	-2.42e-04	-1.31e-04	-3.43e-05	2.12e-06	3.56e-05	1.33e-04	2.43e-04	3.95e-04	4.81e-03
roberta-base	CoLA	-3.60e-03	-4.94e-04	-2.65e-04	-1.01e-04	-2.52e-05	1.81e-06	2.89e-05	1.01e-04	2.66e-04	4.97e-04	3.52e-03
	SST-2	-1.10e-02	-1.18e-03	-6.69e-04	-3.43e-04	-1.44e-04	1.04e-05	1.64e-04	3.81e-04	6.82e-04	1.19e-03	9.08e-03
	MRPC	-7.82e-03	-1.26e-03	-6.87e-04	-2.95e-04	-5.84e-05	2.92e-06	7.03e-05	2.94e-04	6.86e-04	1.26e-03	8.06e-03
	STS-B	-2.68e-03	-4.45e-04	-2.54e-04	-1.11e-04	-2.10e-05	1.22e-06	3.39e-05	1.12e-04	2.55e-04	4.46e-04	2.77e-03
	QQP	-3.29e-02	-3.16e-03	-2.01e-03	-1.18e-03	-5.33e-04	6.27e-05	6.51e-04	1.32e-03	2.14e-03	3.30e-03	2.56e-02
	MNLI	-3.22e-02	-3.39e-03	-2.17e-03	-1.28e-03	-5.78e-04	6.97e-05	7.09e-04	1.43e-03	2.31e-03	3.54e-03	2.31e-02
	QNLI	-7.69e-03	-1.22e-03	-7.51e-04	-4.35e-04	-1.83e-04	8.53e-06	2.10e-04	4.54e-04	7.67e-04	1.23e-03	9.11e-03
	RTE	-1.81e-03	-2.90e-04	-1.63e-04	-7.32e-05	-1.01e-05	3.86e-07	1.24e-05	7.38e-05	1.63e-04	2.91e-04	2.01e-03