Unlocking Continual Learning Abilities in Language Models

Anonymous ACL submission

Abstract

Language models (LMs) exhibit impressive performance and generalization capabilities. However, LMs struggle with the persistent challenge of catastrophic forgetting, which undermines their long-term sustainability in continual learning (CL). Existing approaches usually address the issue by incorporating old task data or task-wise inductive bias into LMs. However, old data and accurate task information are often unavailable or costly to collect, hindering the availability of current CL approaches for LMs. To address this limitation, we introduce “MIGU” (**M**agn**I**tude-based **G**radient **U**pdating for continual learning), a rehearsal-free and task-label-free method that only updates the model parameters with large magnitudes of output in LMs’ linear layers. MIGU is based on our observation that the normalized magnitude distribution of the output in LMs’ linear layers is different when the LM models deal with different task data. By imposing this simple constraint on the gradient update process, we can leverage the inherent behaviors of LMs, thereby unlocking their innate CL abilities. Our experiments demonstrate that MIGU is universally applicable to all three LM architectures (T5, RoBERTa, and Llama2), delivering state-of-the-art or on-par performance across continual finetuning and continual pre-training settings on four CL benchmarks. For example, MIGU brings a 15.2% average accuracy improvement over conventional parameter-efficient finetuning baselines in a 15-task CL benchmark. MIGU can also seamlessly integrate with all three existing CL types to further enhance performance. We include the code in the submission attachment.

1 Introduction

Neural networks suffer from catastrophic forgetting (McCloskey and Cohen, 1989), i.e. learning new knowledge and tasks at the cost of forgetting previously acquired ones. Recently, language models (LMs) have demonstrated impressive performance and generalization capabilities across the spectrum of NLP research (Liu et al., 2019; Brown et al., 2020; Touvron et al., 2023) and beyond (Zhang et al., 2023a). Nevertheless, they still suffer from catastrophic forgetting (Shi et al., 2024; Wu et al., 2024), undermining the capacity for continual learning (CL) (Wang et al., 2024a). In light of the large scale and high cost of training LMs (Achiam et al., 2023), models with strong continual learning capabilities would enable more economical reuse of these resource-intensive models, a vital trajectory for driving both scientific development and societal benefits.

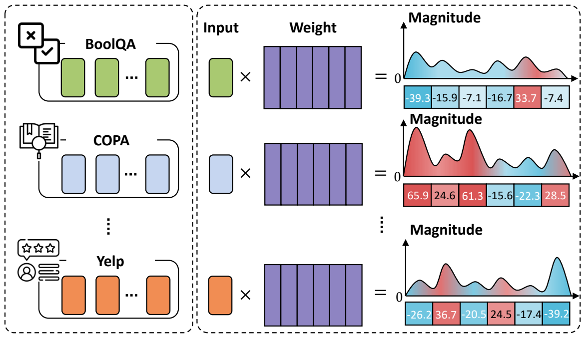


Figure 1: The different output values and the varying magnitude distributions are observed across the BoolQA, COPA, and Yelp datasets. The output values and normalized magnitude distributions are from the real sample of the first linear layer in FFN of the last Transformer block of T5. The detailed magnitude distributions are illustrated in Figure 17 of Appendix D.4.

To make LMs better continual learners, the research community pursues three main directions (Shi et al., 2024): (1) rehearsal-based approaches that mix new task data with a small buffer of past task examples (Scialom et al., 2022; Wang et al., 2024d), (2) architecture-based methods that introduce new components like adapters to incorporate new tasks (Gururangan et al., 2021; Qin et al., 2022; Zhao et al., 2024; Wang et al., 2024b), and (3) parameter-based approaches that either apply regularization to penalize changes in important parameters for old tasks (Zheng et al., 2023; Zhu et al., 2024) or update parameter gradients for each task into orthogonal subspaces (Wang et al., 2023b).

However, rehearsal-based methods require data from previously learned tasks, which is not always available (Touvron et al., 2023). The architecture and parameter-based approaches typically rely on task labels to design techniques to mitigate gradient conflicts between tasks by updating the parameters task-wise. However, obtaining accurate task labels can be challenging or infeasible in LMs' scenarios. This paper explores an alternative approach, examining whether the model's inherent features or behaviors can be utilized instead of task labels to mitigate gradient conflicts between tasks. Concretely, we examine the distribution of the normalized output magnitude of the linear layers in LMs. The output is computed as the dot product between the input $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$ and the weight $\mathbf{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$ of the layer, and then the output is normalized using the L1-norm, resulting in a vector $\mathbf{n} \in \mathbb{R}^{d_{\text{out}}}$. Our analysis reveals an intriguing finding: the normalized output $\mathbf{n}$ exhibits distinct magnitude distributions for different tasks[1]. The process and observation described above are illustrated in Figure 1, which presents real output data for the first linear layer of the Feedforward Network (FFN) in the last Transformer block of T5. We can observe that the magnitude distributions differ significantly for three example tasks - BoolQA, COPA, and Yelp. Motivated by this observation, we argue that the differences in magnitude distributions within LMs could serve as a natural, label-free alternative to replace the need for external task labels in mitigating gradient conflicts to update the model's parameters task-wise. However, this potential is locked during conventional continual learning settings.

To this end, we introduce "MIGU" (**M**agn**I**tude-based **G**radient **U**pdating for continual learning), leveraging the inherent differences in magnitude distributions of the normalized output in LMs' linear layers to enable continual learning without relying on task labels. Specifically, during the forward propagation phase, we cache and normalize the output of the linear layers using the L1-norm. Then, in the backward propagation phase, we only update the parameters with the T largest values in normalized magnitude, where T is a predefined threshold ratio. Since different tasks exhibit distinct magnitude distribution patterns, MIGU can effectively harness the LMs' inherent features to update the parameters with large magnitudes per task, alleviating gradient conflicts and unlocking their innate continual learning potential.

We evaluate MIGU across three main LM architectures: the encoder-only RoBERTa (Liu et al., 2019), the encoder-decoder T5 model (Raffel et al., 2023), and the decoder-only Llama2 (Touvron et al., 2023). Furthermore, we consider two continual pre-training settings for LMs: continual pretraining and continual finetuning, using four CL datasets. Notably, our approach can seamlessly integrate three mainstream CL approaches - rehearsal-based, architecture-based, and parameter-based - to further enhance the CL abilities of LMs. When evaluated on the four datasets, our experimental results achieve comparable or superior performance to the current state-of-the-art methods. For example, in a 15-task long sequence CL dataset, the MIGU leads to a 15.2% accuracy improvement over the conventional parameter-efficient finetuning baseline. Furthermore, combining MIGU with three types of CL methodologies substantially improves these individual CL approaches. We also provide detailed ablation studies and visualizations on MIGU, revealing that CL with MIGU pushes the magnitude distribution similarity between tasks farther apart and better avoids conflicts. We believe the work presents a novel perspective on exploring CL in LMs.

2 Related Work

Continual Learning for Language Models. Continual learning is a long-standing challenge through the history of machine learning and deep learning (McCloskey and Cohen, 1989; Wu et al., 2024). Recent studies for CL in LMs can be roughly categorized into three categories. 1. Rehearsal-based approach that mixes new task data with a small buffer of past task examples (Scialom et al., 2022; Wang et al., 2024d). 2. Architecture-based approach that expands new modules like adapters to incorporate new tasks (Gururangan et al., 2021; Qin et al., 2022; Zhao et al., 2024; Wang et al., 2024b). 3. Parameter-based method that updates parameters in a task-aware manner. Some literature (Wang et al., 2023a) splits the

[1]The term 'normalized output magnitude distribution' will be referred to interchangeably as 'magnitude distribution' for brevity throughout the paper.

parameter-based into either regularization-based approaches that add a regularization term to penalize changes in important weights of the earlier learned tasks (Zheng et al., 2023; Zhu et al., 2024), or optimization-based approaches that updates parameters gradients for each task into orthogonal subspaces to avoid conflicts (Wang et al., 2023b). These methods rely on either old task data or accurate task labels, which are hard or expensive to collect for LMs' continual training. In contrast, MIGU only leverages LMs' innate features for CL.

Partially Updating Parameters in Continual Learning. Among existing CL methods for LMs, our approach and regularization-based (Zheng et al., 2023; Zhu et al., 2024) approach both partially update parameters, but ours fundamentally diverges from the regularization-based methods in motivation and design. While they rely on backward gradients to identify and protect important weights for old tasks, we leverage the differences in magnitude distribution across tasks during the feed-forward phase. Additionally, our method's ability to freely mask at the sample level sets us apart from their fixed gradient mask approach. Furthermore, our method does not require task labels, enabling it to work in broader scenarios where task labels are unavailable. Lastly, the layer output distributions are naturally obtained during the feed-forward phase training, whereas they normally require an additional subset to derive the gradient mask before training on a task.

Finding Important Weights. One may classify our method as a broader research cluster centered on finding important weights, a topic that has been extensively explored in continual learning (Zhu et al., 2023), model pruning and compression (Frankle and Carbin, 2019), efficient training and inference (Ansell et al., 2024), as well as investigations into activation sparsity (Zhang et al., 2023b; Song et al., 2024), and other related areas. However, these works mostly use weight or gradient magnitude to define a fixed size of the important weights. A few works on activation sparsity use the sparsity patterns after the activation function for either efficient inference (Zhang et al., 2022) or performance improvements (Qiu et al., 2024). None of the above explore the general dot product of weights and layer input. The closest work to ours is an unstructured pruning work (Sun et al., 2024) using the dot product of weight and input, demonstrating a superior method to pure weight-based pruning. However, this prior work fails to consider the varying patterns of important weights across different tasks. In contrast, our method utilizes the normalized dot product of weight and input as an inherent indicator of importance in CL settings.

	RF	**TIFT**	**CIT**	**CPT**
LFPT5 (Qin and Joty, 2021)			✓	
EPI (Wang et al., 2023d)	✓		✓	
O-LoRA(Wang et al., 2023b)	✓		✓	
MoCL (Wang et al., 2024c)	✓		✓	
SAPT (Zhao et al., 2024)	✓		✓	
DAS (Ke et al., 2023)	✓			✓
MIGU	✓	✓	✓	✓

Table 1: The comparison between MIGU and other CL methods. Specifically, **RF** indicates whether the method is rehearsal-free. **TIFT** indicates whether the method is task-id-free during training. **CIT** indicates whether the method supports instruction finetuning.**CPT** indicates whether the method supports continual pre-training.

3 Method

In Table 1, we compare MIGU with common CL methods. Our approach is only one rehearsal-free, task-id-free method that supports both continual pre-training and continual finetuning.

3.1 Preliminary - Continual Learning Setup

Continual learning (Ke and Liu, 2022; Wang et al., 2023c; Zhao et al., 2024) aims to tackle the challenges that arise within the ongoing sequence. Formally, tasks $\{\mathcal{T}_1, \ldots, \mathcal{T}_T\}$ arrive in sequentially. Each task $\mathcal{T}_t = \left\{\left(x_t^i, y_t^i\right)\right\}_{i=1}^{n_t}$ contains a separate target dataset with the size of n_t. For any time step t, the model is expected to not only adapt itself to the t-th task, but also retain its capabilities across all the previous tasks it has been trained on. This study explores two distinct CL settings. In the first setting, where only the MIGU method is employed, the task label is unavailable during the training and testing phases. Secondly, when combined with the three existing types of CL techniques, the model can be exposed to old task data or task information during the training phase.

3.2 MIGU - MagnItude-based Gradient Updating for Continual Learning.

Our approach employs a two-step process to leverage the inherent differences in magnitude distributions across various tasks for continual learning.: 1) Caching output magnitudes and 2) Updating gradient via a magnitude-based mask. We show the

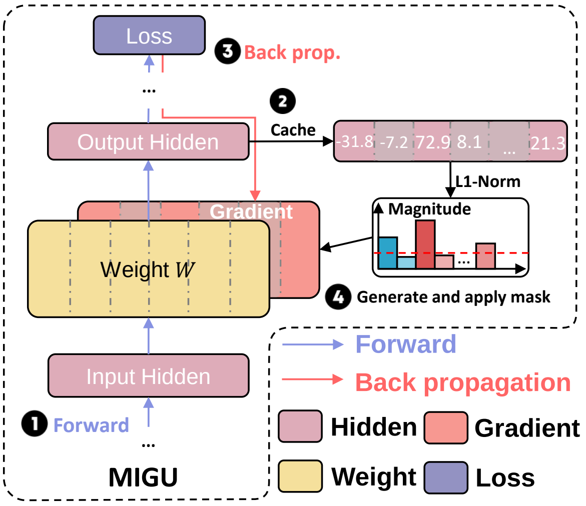


Figure 2: Proposed method: MIGU. During 1) the forward phase, our method 2) caches the output magnitude of the linear layers, and 3) after backpropagation, 4) MIGU masks the gradients by cached magnitudes to update parameters accordingly.

process in Figure 2. To illustrate our method, we first consider the fundamental component in LMs, a single linear layer with weight $\mathbf{W}$ [2] and only feed a token input into LMs.

Feedforward: Caching Output Magnitudes. Given the weight matrix $\mathbf{W} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}}}$, we interpret the columns of $\mathbf{W}$ as a set of d_{out} vectors, each with dimension d_{in}:

$$\mathbf{W} = [\mathbf{w_1}, \ldots, \mathbf{w_i}, \ldots \mathbf{w_{d_{out}}}], \text{where } \mathbf{w_i} \in \mathbb{R}^{d_{\text{in}}} \tag{1}$$

Given the input vector of the layer $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$, the operation of the layer can be viewed as the dot product between $\mathbf{x}$ and each weight vector $\mathbf{w_i}$:

$$h_i = \mathbf{x} \cdot \mathbf{w_i} \tag{2}$$

We then compute the normalized product magnitude n_i using the L1-norm by : $n_i = \|h_i\|_1$, where $\|\cdot\|_1$ denotes the L1-norm. Thus, we have the normalized magnitude product distribution vector $\mathbf{n}$ for $\mathbf{W}$.

Backward Propagation: Updating gradient via a magnitude-based mask. After calculating the gradient in the backward phase, we obtain the gradient matrix $\nabla\mathbf{W}$ for the weight $\mathbf{W}$, which presents the optimization direction given the input $\mathbf{x}$. We then define a mask matrix $\mathbf{M}$ to partially mask $\nabla\mathbf{W}$ using the normalized product magnitudes cached during the forward phase. Formally, we sort the product magnitudes in the descending order and mask the corresponding gradient as follows:

$$t = \lfloor T \times d_{\text{out}} \rfloor \tag{3}$$

$$\mathbf{M} = \text{BinaryTopT}(\mathbf{n}, t) \tag{4}$$

$$\text{BinaryTopT}(\mathbf{n_i}, t) = \begin{cases} 1 & \text{if } \mathbf{n_i} \text{ is in the top } 1 - t \text{ elements of } \mathbf{n}. \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where T is the threshold ratio to mask gradient, t is the actual number t to mask, $\lfloor . \rfloor$ is the floor rounding. The model update rule is then given by:

$$\mathbf{W}_{\text{new}} \leftarrow \mathbf{W} - \eta \cdot \mathbf{M} \odot \nabla\mathbf{W} \tag{6}$$

where η is the learning rate. This formulation ensures that only those weights with normalized magnitudes exceeding the threshold T are updated.

3.3 MIGU In Transformers

In practice, to apply MIGU, we average the product magnitudes of all tokens on a batch to generate the mask for simple implementation.

MIGU in Transformer Block. For a Transformer block, we apply our method from Section 3.2 to the Query, Key, Value, and Output linear layer of the multi-head attention (MHA) component, and two (for T5 and RoBERTa) or three (for Llama) linear layers in the FFN component.

MIGU in LoRA Implementation. We also implement MIGU for parameter-efficient finetuning (PEFT) of LMs, particularly we employ Low-Rank Adaptation (LoRA) (Hu et al., 2022). The standard LoRA is mathematically represented as follows:

$$\mathbf{x_A} = \mathbf{x} \cdot \mathbf{A} \tag{7}$$

$$\mathbf{x_B} = \mathbf{x_A} \cdot \mathbf{B} \tag{8}$$

$$\mathbf{x_O} = \mathbf{x} \cdot \mathbf{W} + \frac{\alpha}{r} \cdot \mathbf{x_B}, \tag{9}$$

where $\mathbf{x}$ denotes the input representation of the layer, $\mathbf{A} \in \mathbb{R}^{d_{in} \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times d_{out}}$ are the low-rank matrices, α is a scaling constant, $\mathbf{W}$ is the original weight matrix of the standard linear, and $\mathbf{x_O}$ is the output after applying the LoRA transformation.

To implement MIGU, we apply the same method in Section 3.2 for the matrix $\mathbf{A}$. But for the matrix $\mathbf{B}$, we use the output of $\mathbf{x_O}$ in Equation 8 rather than the output of $\mathbf{x_B}$ in Equation 9 to compute the magnitude distribution vector.

[2] For simplicity, we omit the bias term $\mathbf{b}$ here.

CL Benchmark	Standard	Long
LFPT5	72.7	69.2
EPI	65.3	-
MoCL	75.9	-
SAPT-LoRA	-	**82.0**
IncLoRA	68.8	64,7
+ MIGU	76.4(7.6↑)	68.7(4.0↑)
OIncLoRA	75.8	69.6
+ MIGU	**76.6**(0.8↑)	70.0(0.4↑)
LoRAReplay	74.5	75.2
+ MIGU	76.2(1.7↑)	76.5(1.3↑)
MoELora	54.1	27.6
FT	75.7	68.3
+ MIGU	**78.8**(3.1↑)	**73.8**(5.5↑)
LoRA	67.9	46.0
+ MIGU	73.3(5.4↑)	61.2(15.2↑)

Table 2: Average accuracy on two standard CL benchmarks with T5-large model. The top block contains CL methods with extra old task data or task labels, while the bottom does not.

4 Experiments

We use three language models adopted by the previous lines of works in CL for NLP: encoder-only RoBERTa (Liu et al., 2019), encoder-decoder T5 model (Raffel et al., 2023) and decoder-only Llama2 (Touvron et al., 2023). We start with continual finetuning T5-large (Raffel et al., 2020) on two CL datasets following the settings from (Qin and Joty, 2021; Wang et al., 2023b).We implement MIGU upon vanilla finetuning and PEFT with LoRA (Hu et al., 2022). We also combine our method with three main types of CL approaches to examine the seamless integration of our method with the existing CL methodologies. Next, we use encoder-only RoBERTa to continual pre-traning domain adaptive data, following the setting (Ke et al., 2023). We further scale our experiment to decoder-only Llama2-7B (Touvron et al., 2023) and test the trade-off between base model ability and new task ability. All experimental results are reported as the average of 3 runs. Please refer to the Appendix A.1 for more detailed settings.

4.1 Continual Finetuning on T5-large

Two Benchmarks. We evaluate our approach to continual finetuning on T5-large using the standard CL benchmark and long sequence benchmark. We follow the setup from (Qin and Joty, 2021; Wang et al., 2023b) to shuffle the four text classification tasks from the LM dataset (Zhang et al., 2015) into three different orders to form Order 1, 2, 3 for standard CL benchmark. Similarly, we shuffle a mix of 15 tasks (five classification tasks, nine GLUE and SuperGLUE tasks, and the IMDB dataset) to form Orders 4, 5, and 6 for the long sequence benchmark. For the details on benchmark and sequence, please refer to the appendix C.1.

Baselines. We separate the baselines into two categories: without old data or task information and with old data or task information during training. For the first category, we include vanilla **FT**, which trains all model parameters on a sequence of tasks, and vanilla **LoRA**, in which fixed-size LoRA parameters are trained on a sequence of tasks. For the second category, we have rehearsal-based approaches: **LoRAReplay** that trains new tasks on LoRA with mixing a 2% past task, **LFPT5** (Qin and Joty, 2021) continuously trains a soft prompt that simultaneously learns to solve the tasks and generate training samples for experience replay; architecture-based approaches: **IncLoRA** that incremental learning of new LoRA parameters on a sequential series of tasks, **MoELora** (Luo et al., 2024), a vanilla MoE with LoRA number equals to the task number, **SAPT-LoRA** (Zhao et al., 2024) extends IncLoRA by aligning learning process and selection process of LoRA, and **MoCL** (Wang et al., 2024c) continually adds new modules and composes them with existing modules; parameter-based approaches **OIncLoRA** (Wang et al., 2023b)[3] extends IncLoRA to learn different LoRAs into orthogonal subspaces.

Metrics. Average Accuracy (Chaudhry et al., 2018). The average performance of all tasks after training on the last task, i.e., $A_{\mathcal{T}} = \frac{1}{\mathcal{T}} \sum_{t=1}^{\mathcal{T}} a_{\mathcal{T},t}$.

Results on T5. Table 2 shows that our proposed approach improves the performance of all five CL approaches. Notably, when our method is applied, the vanilla FT and LoRA baselines see substantial improvements. Some results obtained using our approach are comparable to the SOTA CL methods that leverage task labels or old task data. Notably, the LoRA+MIGU approach surpasses the vanilla LoRA method by a substantial

[3] O-LoRA is original name, we rename it to OIncLoRA to emphasize it is build upon IncLoRA and align with our notation.

15.2% on the long sequence benchmark, significantly mitigating the drawbacks of LoRA in the CL setting with long sequences. We chose to combine our method with three LoRA-based techniques to integrate with three CL approaches that leverage old data or additional labels. The parameter-based IncLoRA+MIGU exhibits the most significant improvement over the original IncLoRA, implying that our magnitude-based approach can effectively mitigate the conflicts among the sequentially learned LoRA parameters in IncLoRA. The relatively marginal improvement of parameter-based OIncLoRA+MIGU indicates a similar function between our approach and projecting LoRAs into orthogonal subspaces, but our method does not require task labels during the continual training process. SAPT-LoRA achieves the SoTA performance in long sequence benchmark, but it requires both task labels and past data, which are often infeasible or costly in LMs settings. We also report an efficiency study in Appendix D.2 Table 9 to show our approach only leads to a minor overhead over the vanilla methods, which is assumed to be more efficient than other CL methods. We provide a full experiment in Appendix D.1 Table 8. We also draw the Violin Plot to show the statistical significance of our approach in Appendix D.1.

4.2 Continual Pre-training on RoBERTa

Benchmark. In contrast to the previous continual finetuning setting, (Ke et al., 2023) introduces DAS, a new benchmark for continual pre-training of LMs. DAS is composed of 6 unlabeled domain corpora, which contain 3 reviews and 3 academic papers. It is then evaluated using 6 corresponding classification datasets. Please refer to the Appendix B.1 for the details.

Metrics. For continual pre-training, We utilize MF1 (Macro-F1) and ACC (Accuracy) following (Ke et al., 2023) to evaluate the performance after pre-training on the last domain.

Baselines. We choose top baselines range from vanilla methods that pre-train RoBERTa on domains sequentially with full parameters **FT** and with PEFT **Adapter** to rehearsal-based (**DER++** (Buzzega et al., 2020)), architecture-based (**DEMIX** (Gururangan et al., 2021)), and parameter-based **HAT-Adapter** (Serrà et al., 2018) and **DAS** (Ke et al., 2023).

Metrics	MF1	ACC
DEMIX	74.70	79.66
DER++	75.78	80.46
HAT-Adapter	74.63	79.78
DAS	**77.90**	**81.90**
DAS*	76.59	81.07
Adapter	74.05	79.48
FT	76.36	80.77
+ MIGU	**76.73**(0.37↑)	**81.19**(0.42↑)

Table 3: Average MF1, ACC on the DAS benchmark after continual pre-training on all domains and finetuning on their corresponding end-task datasets. DAS* is the result we reproduced.

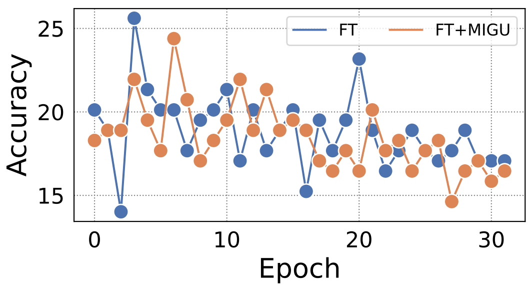
	Avg of 1-2	Avg of 5-6
FT	79.69	82.07
DAS	**80.30**(0.61↑)	81.16(0.91↓)
MIGU	80.14(0.45↑)	**82.41**(0.34↑)

Table 4: The average ACC of the first and last two learned domains in the DAS benchmark.

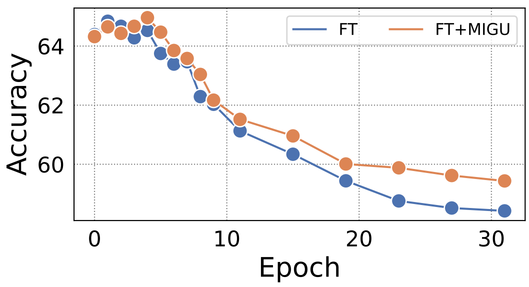
Results on RoBERTa. We further evaluate MIGU in another setting in which, instead of finetuning, we continually pre-train a RoBERTa model to six domains sequentially (domain-adaptive pre-training). Our experimental results in Table 3 also show promising results of our approach over or on par with the sophisticated CL methods with task labels or old data. For instance, FT+MIGU achieves 0.37% improvement in MF1 and 0.42% in ACC. We also explore the performance of the domains in different orders. We report the average ACC of the first and last two learned domains in Table 4. The results indicate that while the DAS model exhibits less forgetting in the earlier learned domains, but it also learns less in the last domains, possibly due to the strong regularization used to constrain its parameter updates during the CL process over a long sequence. In contrast, MIGU demonstrates a more sustainable method, exhibiting robust performance on the earlier and recently learned domains.

4.3 Forget Less and Learn the Same: Scaling to Llama2

Results on Llama2. We further assess our approach on a more demanding LLM continual instruction tuning setting. We finetune a base



(a) Learn the same. Instruction tuning results on Human eval. MIGU with LoRA learns the same as the valinna LoRA.



(b) Forget less. Average accuracy on HellaSwag, Winogrande, ARC-Challenge for Llama-2-7B. The results indicate that MIGU with LoRA forgets less than valinna LoRA.

Figure 3: Performance comparison of LoRA with MIGU and the baseline vanilla LoRA on Llama2-7B instruction tuning, evaluated using the Humaneval (Chen et al., 2021), as well as on LM benchmarks: HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2019), and ARC-Challenge (Clark et al., 2018).

Llama2-7B on Magicoder-Evol-Instruct-110K for 32 epochs. This dataset (Wei et al., 2024) contains 72.97M tokens of programming questions and answers. However, due to computation constraints, we sample 20% of data and conduct experiments on LoRA. We follow (Biderman et al., 2024) to assess LoRA+MIGU's capabilities on both the base ability (forgetting domain) and the code ability (learning domain). To evaluate code learning performance, we utilize the Humaneval benchmark (Chen et al., 2021)), which contains 164 problems that generate a Python program with a docstring and a function signature. A generation is considered correct if it passes all supplied unit tests. To quantify how much they have forgotten previous knowledge, we follow (Biderman et al., 2024) that utilizes average scores of three benchmarks, HellaSwag (Zellers et al., 2019), WinoGrade (Sakaguchi et al., 2019) and ARC-challenge (Clark et al., 2018). The experiments are shown in Figure 16. Compared to baseline FT, our method learns a similar level of new code knowledge but exhibits significantly less forgetting of previous knowledge. This suggests our approach achieves a better trade-off point on the Pareto frontier between learning plasticity and memory stability (Huang, 2003; Wang et al., 2024a). For example, after 32 training epochs, the average accuracy across the three benchmarks for our method is 59.4, while the baseline model only achieves 58.4.

5 Discussions

We then provide ablations on gradient mask threshold and components as well as a visualization.

5.1 Ablation on Gradient Mask Threshold

We plot all five curves of our approach for gradient mask threshold from 0.0 to 0.9 for our methods in T5-large experiments. The optimal threshold value for FT, LoRA, and IncLoRA settings is 0.7 while LoRAReplay is 0.4 as shown in Figure 4[4]. OIncLoRA is only 0.1, which is plausible due to the parameter updating regularized by the OIncLoRA method itself. The optimal value for IncLoRA is 0.6, close to FT, LoRA, and IncLoRA settings. Surprisingly, with only 5% ($T = 0.95$) or 1% ($T = 0.99$) parameters updating, LoRA+MIGU still beats LoRA by a wide margin. This interesting finding may indicate that only a small proportion

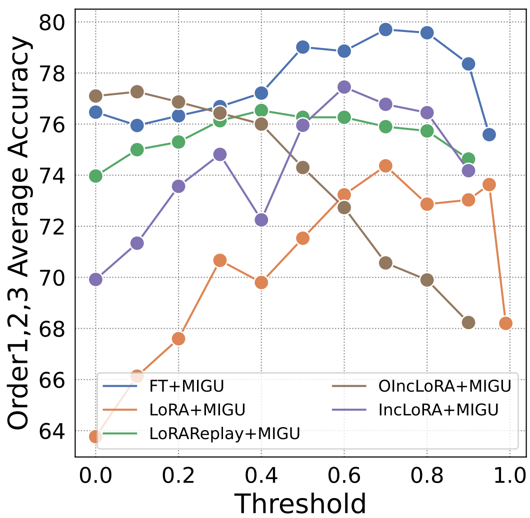


Figure 4: Ablation study on the gradient mask threshold. The curves illustrate that the optimal value is concentrated around 0.7 for FT, LoRA, and IncLoRA, 0.4 and 0.1 for the LoRAReplay and OIncLoRA settings respectively.

[4]The ablation on threshold search only reports one run, so it does not align with the experiments in Section 4.1.

	Order1	Order2	Order3	Avg
FT + MIGU	79.6	**80.3**	**79.2**	**79.7**
FT	75.3	76.1	78.0	76.5
+ FFN 1_{st}-L	76.9	75.9	75.8	76.2
+ FFN all	77.2	77.2	76.7	77.0
+ Attention Q	77.2	76.4	78.3	77.3
+ Attention K	76.9	73.4	75.6	75.3
+ Attention V	75.4	76.3	78.0	76.6
+ Attention O	76.2	75.9	76.0	76.0
+ Attention all	**80.2**	78.8	79.0	79.3

Table 5: The ablation results from applying MIGU to different LM components."+ FFN all" means only applying MIGU to all the linear operators in FFN layers. The results demonstrate implementing MIGU across all linear layers leads to the most benefits.

of proportional weights with large magnitudes is crucial for successful CL settings, which may be worth future investigation.

5.2 Ablation on Gradient Mask Components

We further investigate which components within a transformer block should utilize MIGU. Typically, a transformer block consists of six linear layers: the query, key, and value (QKV) linear layers and the output linear layer (O) in the MHA module, as well as the two linear layers in the FFN. Our analysis shows that employing MIGU across all these linear layers achieves the best overall performance, suggesting that the magnitude-based approach is effective for linear layers in different parts of the transformer architecture.

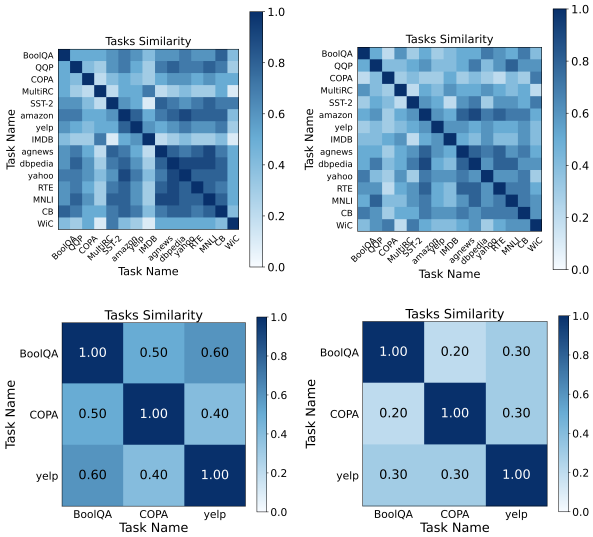
5.3 Visualization

We plot task similarity by counting the overlapping ratio of updated parameters (large magnitudes) positions by using 100 samples per task. In Figure 5, we visualize the task similarity for the first layer of FFN in the last Transformer block of T5-large, comparing FT and FT+MIGU in the Order1 setting. The results clearly show that MIGU increases the degree of parameter isolation across tasks, achieving a similar effect by using task information but without relying on such explicit task labels. We further highlight the similarity between the BoolQA, COPA, and Yelp tasks and the notable decrease in similarity among these three tasks. Analyzing the performance results shown in Table 6, we find that the significant reduction in overlapping ratio across tasks considerably alleviates the task conflicts, resulting in much more significant performance gains.

	BoolQA	COPA	Yelp
FT	67.3	45.0	39.1
FT + MIGU	78.3(11.0$\uparrow$)	55.0(10.0$\uparrow$)	47.6(8.5$\uparrow$)

Table 6: The improvement on BoolQA, COPA and Yelp in Order 1.

For example, the accuracy improvement for the COPA dataset is exactly 10%. We put the full visualization of all linear layers in Appendix D.4.



(a) FT (b) FT + MIGU

Figure 5: The visualization of magnitude distribution similarity across different tasks. FT+ MIGU is lower, indicating that MIGU reduces the possibility of weight conflicts between tasks. The two sub-figures at the bottom are three highlighted task samples: BoolQA, COPA and Yelp.

6 Conclusion

We propose MIGU, a rehearsal-free and task-label-free method that only updates the model parameters with large output magnitudes in LM's linear layers. By imposing this simple constraint on the gradient update process, we can leverage the inherent behaviors of LMs, thereby unlocking their innate CL abilities. Our experiments, applied to all three LM architectures (T5, RoBERTa and Llama2), on two CL scenarios (continual finetuning and continual pre-training) and four CL benchmarks, consistently deliver better performance. Our method can also be seamlessly integrated with existing CL solutions to further improve their performance.

7 Limitations

We acknowledge two limitations for this work. Due to computation limitations, although we finetune Llama2-7B with LoRA, we are unable to scale our experiments to LM continual pre-training or full tuning. However, our experimental performance on continual pre-training RoBERTa indicates the great potential for the scalability of this general approach. Another limitation is we only explore an approach for unlocking the inherent CL potential of LMs through updating the gradient by the magnitude of output. There exists more discussions on exploiting innate features such as activation sparsity as discussed in the Related Work section. These limitations can be further addressed in future work.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Alan Ansell, Ivan Vulić, Hannah Sterz, Anna Korhonen, and Edoardo M. Ponti. 2024. Scaling sparse fine-tuning to large language models. *Preprint*, arXiv:2401.16405.

Dan Biderman, Jose Gonzalez Ortiz, Jacob Portes, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. 2024. Lora learns less and forgets less. *Preprint*, arXiv:2405.09673.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. 2020. Dark experience for general continual learning: a strong, simple baseline. *arXiv preprint arXiv:2004.07211*.

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

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. 2018. *Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence*, page 556–572. Springer International Publishing.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri 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, Josh 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. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Cyprien de Masson D'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. *Advances in Neural Information Processing Systems*, 32.

Xiaowen Ding, Bing Liu, and Philip S Yu. 2008. A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 international conference on web search and data mining*.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *Preprint*, arXiv:1803.03635.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A Smith, and Luke Zettlemoyer. 2021. Demix layers: Disentangling domains for modular language modeling. *arXiv preprint arXiv:2108.05036*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of ACM SIGKDD*.

Guang-Bin Huang. 2003. Learning capability and storage capacity of two-hidden-layer feedforward networks. *IEEE Transactions on Neural Networks*, 14(2):274–281.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

David Jurgens, Srijan Kumar, Raine Hoover, Daniel A. McFarland, and Dan Jurafsky. 2018. Measuring the evolution of a scientific field through citation frames. *TACL*.

Zixuan Ke and Bin Liu. 2022. Continual learning of natural language processing tasks: A survey. *ArXiv*, abs/2211.12701.

Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. 2023. Continual pre-training of language models. *arXiv preprint arXiv:2302.03241*.

Zixuan Ke, Hu Xu, and Bing Liu. 2021. Adapting bert for continual learning of a sequence of aspect sentiment classification tasks. In *NAACL*, pages 4746–4755.

Jens Kringelum, Sonny Kim Kjaerulff, Søren Brunak, Ole Lund, Tudor I Oprea, and Olivier Taboureau. 2016. Chemprot-3.0: a global chemical biology diseases mapping. *Database*, 2016.

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

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel S. Weld. 2020. S2ORC: the semantic scholar open research corpus. In *ACL*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *ACL*.

Tongxu Luo, Jiahe Lei, Fangyu Lei, Weihao Liu, Shizhu He, Jun Zhao, and Kang Liu. 2024. Moelora: Contrastive learning guided mixture of experts on parameter-efficient fine-tuning for large language models. *arXiv preprint arXiv:2402.12851*.

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

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Jianmo Ni, Jiacheng Li, and Julian J. McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *EMNLP*, pages 188–197. Association for Computational Linguistics.

Chengwei Qin and Shafiq Joty. 2021. Lfpt5: A unified framework for lifelong few-shot language learning based on prompt tuning of t5. *arXiv preprint arXiv:2110.07298*.

Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. Elle: Efficient lifelong pre-training for emerging data. *Preprint*, arXiv:2203.06311.

Zihan Qiu, Zeyu Huang, and Jie Fu. 2023. Emergent mixture-of-experts: Can dense pre-trained transformers benefit from emergent modular structures? *arXiv preprint arXiv:2310.10908*.

Zihan Qiu, Zeyu Huang, and Jie Fu. 2024. Unlocking emergent modularity in large language models. *Preprint*, arXiv:2310.10908.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. 2023. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. 2022. Fine-tuned language models are continual learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6107–6122.

Joan Serrà, Didac Suris, Marius Miron, and Alexandros Karatzoglou. 2018. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*.

Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, and Hao Wang. 2024. Continual learning of large language models: A comprehensive survey. *arXiv preprint arXiv:2404.16789*.

Yixin Song, Haotong Xie, Zhengyan Zhang, Bo Wen, Li Ma, Zeyu Mi, and Haibo Chen. 2024. Turbo sparse: Achieving llm sota performance with minimal activated parameters. *Preprint*, arXiv:2406.05955.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. 2024. A simple and effective pruning approach for large language models. *Preprint*, arXiv:2306.11695.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2023a. A comprehensive survey of continual learning: Theory, method and application. *ArXiv*, abs/2302.00487.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. 2024a. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Mingyang Wang, Heike Adel, Lukas Lange, Jannik Strötgen, and Hinrich Schütze. 2024b. Rehearsal-free modular and compositional continual learning for language models. *arXiv preprint arXiv:2404.00790*.

Mingyang Wang, Heike Adel, Lukas Lange, Jannik Strötgen, and Hinrich Schütze. 2024c. Rehearsal-free modular and compositional continual learning for language models. *Preprint*, arXiv:2404.00790.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuan-Jing Huang. 2023b. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10658–10671.

Xiao Wang, Yuansen Zhang, Tianze Chen, Songyang Gao, Senjie Jin, Xianjun Yang, Zhiheng Xi, Rui Zheng, Yicheng Zou, Tao Gui, et al. 2023c. Trace: A comprehensive benchmark for continual learning in large language models. *arXiv preprint arXiv:2310.06762*.

Yifan Wang, Yafei Liu, Chufan Shi, Haoling Li, Chen Chen, Haonan Lu, and Yujiu Yang. 2024d. Inscl: A data-efficient continual learning paradigm for fine-tuning large language models with instructions. *arXiv preprint arXiv:2403.11435*.

Zhicheng Wang, Yufang Liu, Tao Ji, Xiaoling Wang, Yuanbin Wu, Congcong Jiang, Ye Chao, Zhencong Han, Ling Wang, Xu Shao, and Wenqiu Zeng. 2023d. Rehearsal-free continual language learning via efficient parameter isolation. *ArXiv*.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2024. Magicoder: Empowering code generation with oss-instruct. *Preprint*, arXiv:2312.02120.

Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. 2024. Continual learning for large language models: A survey. *arXiv preprint arXiv:2402.01364*.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. 2019. BERT post-training for review reading comprehension and aspect-based sentiment analysis. In *NAACL-HLT*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Hang Zhang, Xin Li, and Lidong Bing. 2023a. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. 2022. Moefication: Transformer feed-forward layers are mixtures of experts. *Preprint*, arXiv:2110.01786.

Zhengyan Zhang, Zhiyuan Zeng, Yankai Lin, Chaojun Xiao, Xiaozhi Wang, Xu Han, Zhiyuan Liu, Ruobing Xie, Maosong Sun, and Jie Zhou. 2023b. Emergent modularity in pre-trained transformers. *Preprint*, arXiv:2305.18390.

Weixiang Zhao, Shilong Wang, Yulin Hu, Yanyan Zhao, Bing Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. 2024. Sapt: A shared attention framework for parameter-efficient continual learning of large language models. *Preprint*, arXiv:2401.08295.

Junhao Zheng, Shengjie Qiu, and Qianli Ma. 2023. Learn or recall? revisiting incremental learning with pre-trained language models. *arXiv preprint arXiv:2312.07887*.

Didi Zhu, Zhongyi Sun, Zexi Li, Tao Shen, Ke Yan, Shouhong Ding, Kun Kuang, and Chao Wu. 2024. Model tailor: Mitigating catastrophic forgetting in multi-modal large language models. *arXiv preprint arXiv:2402.12048*.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. 2023. A survey on model compression for large language models. *Preprint*, arXiv:2308.07633.

A Experimental Details

All experiments are run on an A100 $\times$ 8 DGX-machine.

A.1 Continual finetuning on T5

We adapted the code-base from O-LORA.

Finetuning (FT) and FT with MIGU.

- The batch size is set to 64.
- The optimization is performed using the AdamW algorithm with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay coefficient of 0.01.
- The initial learning rate is set to 1×10^{-4}, alongside a static learning rate scheduler.
- The threshold for mask selection is set at 0.7 across orders 1 to 6 in the FT+MIGU configuration.

Low-Rank Adaptation (LoRA) and LoRA with MIGU.

- LoRA configuration: $r = 8, \alpha = 32$, dropout $= 0.05$.
- The learning rate is set to 1×10^{-3}, with all other hyperparameters being consistent with the FT+MIGU configuration.

Incremental LoRA (IncLoRA) and IncLoRA with MIGU.

- For each LoRA module: $r = 8, \alpha = 32$, dropout $= 0.05$.
- Hyperparameters are identical to those specified in the LoRA and LoRA with MIGU settings.

Order-Incremental LoRA (OIncLoRA) and OIncLoRA with MIGU.

- The threshold for mask selection is set at 0.05 across orders 1 to 6 in the FT+MIGU configuration.
- All remaining hyperparameters are consistent with the LoRA and LoRA with MIGU settings.

LoRA Replay and LoRA Replay with MIGU.

- The threshold for mask selection is set at 0.4 across orders 1 to 6 in the FT+MIGU configuration.
- All remaining hyperparameters are consistent with the LoRA and LoRA with MIGU settings.

A.2 Continual pre-training finetune on RoBERTa

We adapted the code-base from DAS.

pre-training.

- The batch size is set to 248.
- The optimization is performed using the AdamW algorithm with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay coefficient of 0.
- The initial learning rate is set to 1×10^{-4}, alongside a linear learning rate scheduler.
- The threshold for mask selection is set at 0.7 on the sequence of tasks.

Tuning.

- The batch size is set to 16.
- The optimization is performed using the AdamW algorithm with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay coefficient of 0.01.
- The initial learning rate is set to 3×10^{-5}, alongside a linear learning rate scheduler.

A.3 Instruct finetuning on Llama2.

- The optimization is performed using the AdamW algorithm with hyperparameters $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a weight decay coefficient of 0.
- The initial learning rate is set to 5×10^{-4}, alongside a cosine learning rate scheduler with warmup $= 0.1$ of the total duration.
- LoRA configuration: $\alpha = 32$, dropout $= 0.05$.

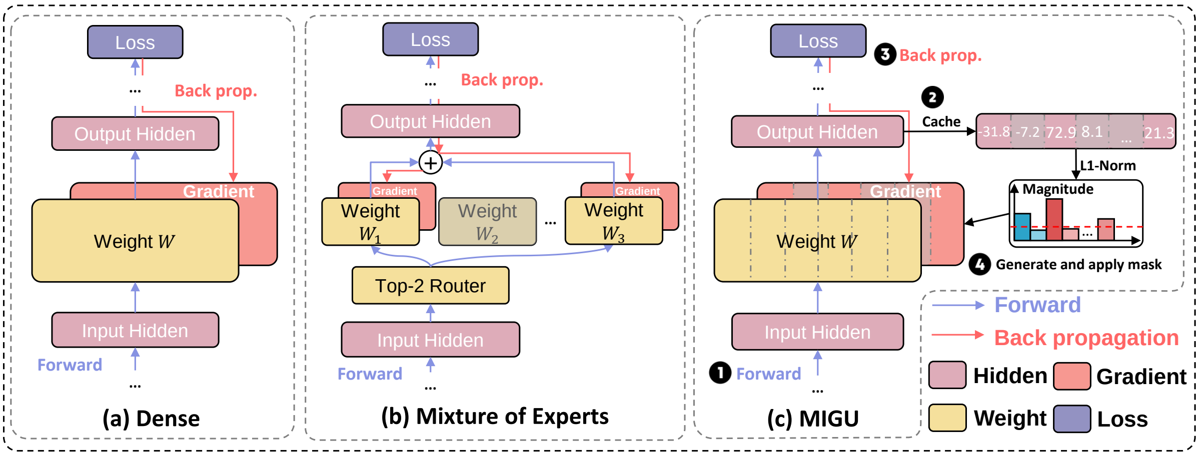


Figure 6: Differences between (a)Dense, (b)MoE and MIGU

Benchmark	Order	Task Sequence
Standard CL	1	dbpedia → amazon → yahoo → ag
	2	dbpedia → amazon → ag → yahoo
	3	yahoo → amazon → ag → dbpedia
Long sequence	4	mnli → cb → wic → copa → qqp → boolqa → rte → imdb → Yelp → amazon → sst-2 → dbpedia → ag → multirc → yahoo
	5	multirc → boolqa → wic → mnli → cb → copa → qqp → rte → imdb → sst-2 → dbpedia → ag → Yelp → amazon → yahoo
	6	Yelp → amazon → mnli → cb → copa → qqp → rte → imdb → sst-2 → dbpedia → ag → yahoo → multirc → boolqa → wic
DAS	7	Restaurant → ACL → AI → Phone → PubMed → Camera

Table 7: Task Sequence Orders for Continual Learning Experiments. Orders 1-3 represent the conventional task sequences employed in standard continual learning benchmarks (Zhang et al., 2015). Orders 4-6 extend to longer sequences, encompassing 15 tasks each (Razdaibiedina et al., 2023). Order 7 comprises a sequence of 6 tasks derived from unsupervised pre-training domains, in accordance with (Ke et al., 2023).

B Benchmark Instruction

B.1 Dataset Information

Our experimental section encompasses datasets including the **Standard CL benchmark** and **Long sequence benchmark**, both of which are utilized for instruction finetuning on the T5-large model; the **DAS benchmark**, which is used for continual pre-training on RoBERTa; the **Magicoder-Evol-Instruct-110K**, which pertains to instruction tuning on Llama-2-7B; and the datasets **Hellaswag**, **WinoGrande**, and **ARC-Challenge** for evaluating the finetuned Llama-2-7B.

Standard CL benchmark. For continual finetuning, we use MTL5 dataset introduced by (Zhang et al., 2015), and follow the setup from LFPT5 and O-LoRA (Qin and Joty, 2021; Wang et al., 2023b) to pick four text classification datasets (AG News, Amazon reviews, DBpedia and Yahoo Answers) and shuffle the tasks into three different orders.

Long sequence benchmark. (Razdaibiedina et al., 2023) extends the Standard CL benchmark by introducing a long sequence benchmark for continual learning benchmark with 15 datasets. This includes five tasks from CL benchmark, four from GLUE benchmark (MNLI, QQP, RTE, SST2) (Wang et al., 2018), five from SuperGLUE benchmark (WiC, CB, COPA, MultiRC, BoolQ) (Wang et al., 2018), and the IMDB movie reviews dataset (Maas et al., 2011). Following (Razdaibiedina et al., 2023), we select 1000 random samples for training each task and hold out 500 samples per class for validation.

DAS Benchmark. (Ke et al., 2023) introduce a new benchmark for continual pre-training of LMs, which is more challenging as the data required to pre-train is much larger and LMs are easier to forget previous knowledge. DAS is composed of 6 unlabeled domain corpora, which contain 3 reviews: Yelp Restaurant (Xu et al., 2019), Amazon Phone (Ni et al., 2019), Amazon Camera (Ni et al., 2019); 3 of them are academic papers: ACL Papers (Lo et al., 2020), AI Papers (Lo et al., 2020), and PubMed Papers [5]. and evaluated by 6 corresponding classification datasets are: Restaurant [6], Phone(Ding et al., 2008; Hu and Liu, 2004), Camera (Ding et al., 2008; Hu and Liu, 2004), ACL (ACL-ARC in (Jurgens et al., 2018)), AI (SCIERC in (Luan et al., 2018)), and PubMed (CHEMPORT in (Kringelum et al., 2016)).

Magicoder-Evol-Instruct-110K. This dataset (Wei et al., 2024) contains 72.97M tokens of programming questions and answers. It reproduces the "Evol-Instruct" dataset of WizardCoder (Luo et al., 2023): an LLM (GPT-4) is iteratively prompted to increase the difficulty of a set of question-answer pairs (from Code Alpaca (Chaudhary, 2023)). Due to computation constraints, we pick contain 20% the samples to instruct tuning the Llama-2-7B model.

HellaSwag, WinoGrade and ARC-challenge. For how much they forget the old knowledge, we follow the (Biderman et al., 2024) that averages three benchmarks, HellaSwag (Zellers et al., 2019), WinoGrade (Sakaguchi et al., 2019) and ARC-challenge (Clark et al., 2018). HellaSwag benchmark includes 70K problems, each describing an event with multiple possible continuations. The task is to pick the most plausible continuation, requiring inferences about nuanced everyday situations. WinoGrande benchmark also assesses commonsense reasoning. It includes 44K problems with sentences that require ambiguous pronoun resolution. ARC-Challenge benchmark consists of 7,787 grade-school level, multiple-choice science questions, testing capabilities in complex reasoning and understanding scientific concepts.

B.2 Training orders

The training orders in 3 benchmarks on T5-large and RoBERTa models are shown in table 7.

[5] https://pubmed.ncbi.nlm.nih.gov/

[6] https://alt.qcri.org/semeval2014/task4/

C Baselines for all settings

C.1 Baselines on Standard CL benchmark and Long sequence benchmark[7]

- **FT** (de Masson D'Autume et al., 2019): train all model parameters on a sequence of tasks (without adding any regularization or replaying samples from the previous tasks).

- **LoRA**: fixed-size LoRA parameters are trained on a sequence of tasks (without adding any regularization or replaying samples from the previous tasks).

- **IncLoRA**: incremental learning of new LoRA parameters on a sequential series of tasks (without adding any regularization or replaying samples from the previous tasks).

- **Replay**: finetune the whole model with a memory buffer, and replay samples from old tasks when learning new tasks to avoid forgetting.

- **LFPT5** (Qin and Joty, 2021): continuously train a soft prompt that simultaneously learns to solve the tasks and generate training samples, which are subsequently used in experience replay.

- **OIncLoRA** (Wang et al., 2023b): learns tasks in different LoRA subspaces that are kept orthogonal to each other and sums all LoRA weights up at testing time.

- **MoCL** (Wang et al., 2024c): MoCL continually adds new modules to language models and composes them with existing modules.

- **SAPT** (Zhao et al., 2024): In the SAPT method, a Shared Attentive Learning and Selection Module (SALS) is employed to guide training samples through optimal PET blocks for task-specific learning, using a unique instance-level attention mechanism. This process ensures efficient continual learning for large language models.

- **MoELORA** (Luo et al., 2024): MoELoRA considers LoRA as a Mixture of Experts, leveraging the modeling capabilities of multiple experts for complex data domains, as well as utilizing LoRA's parameter-efficient characteristics.

[7] We reuse some baseline descriptions from OLoRA

	Standard CL Benchmark (4 tasks)				Longer CL Benchmark (15 tasks)			
	Order-1	Order-2	Order-3	avg	Order-4	Order-5	Order-6	avg
SeqFT	18.9	24.9	41.7	28.5	7.4	7.4	7.5	7.4
SeqLoRA	44.6	32.7	53.7	43.7	2.3	0.6	1.9	1.6
IncLoRA	66	64.9	68.3	66.4	63.3	58.5	61.7	61.2
Replay	55.2	56.9	61.3	57.8	55	54.6	53.1	54.2
EWC	48.7	47.7	54.5	50.3	45.3	44.5	45.6	45.1
LwF	54.4	53.1	49.6	52.3	50.1	43.1	47.4	46.9
L2P	60.3	61.7	61.1	60.7	57.5	53.8	56.9	56.1
LFPT5	67.6	72.6	77.9	72.7	70.4	68.2	69.1	69.2
MoELoRA	52.8	49.6	59.8	54.1	36.3	31.4	15.1	27.6
O-LoRA	77.1	76.2	76.6	76.6	68.4	68.8	71.4	69.5
MoCL	75.6	75.4	76.7	75.9	-	-	-	-
FT	74.4	75.0	77.5	75.7	70.6	69.7	65.6	68.3
MIGU + FT	78.3	79.8	78.3	78.8	77.1	73.6	70.7	73.8
LORA	60.7	70.0	73.1	67.9	53.7	44.4	39.8	46.0
MIGU + LORA	74.8	71.6	73.5	73.3	66.9	64.8	51.8	61.2
OIncLoRA	77.1	76.2	76.6	76.6	68.4	68.8	71.4	69.5
MIGU + OIncLoRA	77.1	77.0	75.6	76.6	67.3	68.5	74.2	70.0
LORAReplay	77.1	73.4	73.2	74.5	74.5	75.4	75.7	75.2
MIGU + LORAReplay	77.8	75.1	75.9	76.2	75.4	76.8	77.2	76.5
ProgPrompt	75.2	75	75.1	75.1	78.0	77.7	77.9	77.9
PerTaskFT	70.0	70.0	70.0	70.0	78.1	78.1	78.1	78.1
MTL	80.0	80.0	80.0	80.0	76.5	76.5	76.5	76.5

Table 8: Summary of the results on two standard CL benchmarks with T5-large model. Averaged accuracy after training on the last task is reported. All results are averaged over 3 runs. (We reuse the table template and some results from OLoRA). **It is noticeable some baselines in some previous literature show significant lower performance than ours, we assume this may due to our strict parameter grid search for baseline methods.**

C.2 Baselines on DAS benchmark

- **NCL** (Naive CL) continually DAP-trains the RoBERTa;
- **NCL-Adapter** continually DAP-trains a set of adapters (Houlsby et al., 2019)
- **DER++** (Buzzega et al., 2020) is a replay method based on knowledge distillation. 16.4K tokens are saved for each domain in the replay memory.
- **DEMIX** (Gururangan et al., 2021) adds a new adapter for each new domain and initializes it with a previous adapter nearest to the new domain;
- **HAT-Adapter** (Serrà et al., 2018): HAT is an effective parameter-isolation method. HAT is applied to Transformer layers (i.e., self-attention, intermediate and output layers).
- **HAT-Adapter** (Ke et al., 2021): HAT-Adapter uses HAT within adapters.
- **DAS** (Ke et al., 2023) DAS proposes a soft-masking method to overcome CF and to encourage KT, and a constrative learning-based method for knowledge integration.

D Experimental Results

D.1 Experiment on T5

We report more detailed results on the Standard CL benchmark and Long sequence benchmark in table 8, including each order results and their corresponding average results. To more intuitively display our results compared to the baseline, we plotted violin graphs showing the performance with and without our method under the condition of full finetuning as Figure 7 8 9 10.

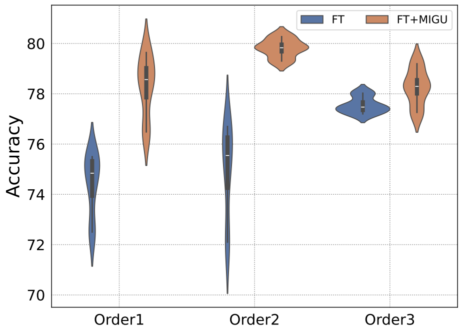


Figure 7: Performance comparison on the standard cl benchmark under full finetuning setting, with and without the implementation of our method..

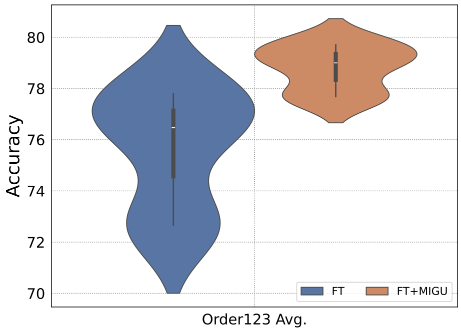


Figure 8: Average performance comparison on the standard cl benchmark under full finetuning setting, with and without the implementation of our method..

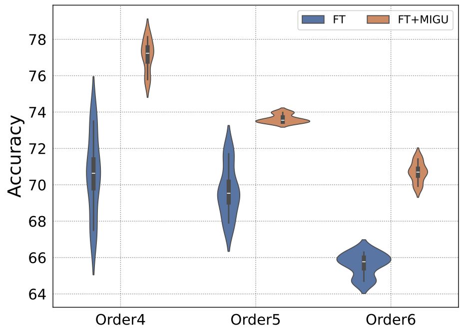


Figure 9: Performance comparison on the standard cl benchmark under full finetuning setting, with and without the implementation of our method..

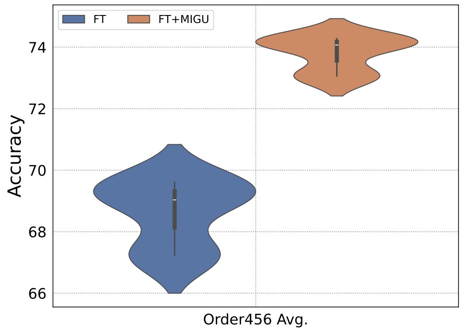


Figure 10: Average performance comparison on the standard cl benchmark under full finetuning setting, with and without the implementation of our method..

D.2 Experiment on RoBERTa

Detailed experiment results The violin graphs results is shown as Figure 11.

Efficiency We also conduct a efficiency test by comparing FT, FT+MIGU and DAS because continual-pre-training is a relatively computation-intensive setting so efficiency is important. DAS is a typical parameter-based regularization methods. We record the time required for the first three dataset given the same computing facilities.

As shown in the figure, FT+MIGU only occur a 1% overhead, due to the extra masking step in the backward propagarion phase while DAS achieves a series overhead due to . wall time DAS vs. ours. vs. FT

D.3 Experiment on Llama2

The detailed violin graphs results about ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019) and Winogrande (Sakaguchi et al., 2019) are seperately shown in Figure 12, 13, 14.

D.4 Visualization

To investigate how our method enhances model performance, we visualized the variation in product magnitudes between an FT model and an FT model augmented with our MIGU technique. We employed heatmaps to depict the similarity in product magnitude distributions across different tasks. Our findings reveal that task similarity in the FT model with MIGU implementation is markedly

	Restaurant	**ACL**	**AI**
FT	25.3(+0.0%)	26.7(+0.0%)	15.5(+0.0%)
FT + MIGU	27.3(+7.9%)	29.2(+9.4%)	17.0(+12.9%)
DAS	78.0(+208%)	66.5(+154%)	45.0(+190%)

Table 9: The wall time(min) on three domain pre-training dataset.

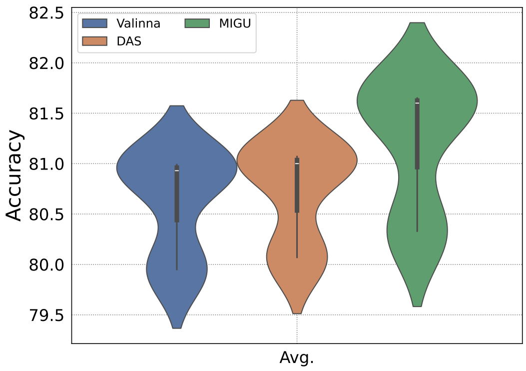


Figure 11: Performance comparison on the DAS benchmark under continual domain pre-training setting, with and without the implementation of our method.

reduced. This suggests that the models trained with our method exhibit more distinctive weight activations for different tasks, thereby mitigating their conflict. This distinction in activation patterns indicates our method's ability to foster more task-specific representations within the model, contributing to its improved performance across varied learning scenarios.

Magnitude Distribution.

E Ablation

Informed by the works on Mixture of Experts (MoE) (Jiang et al., 2024), Emergent MoE (EMoE) (Qiu et al., 2023), and MoE*fication* (Zhang et al., 2022), we investigate explicit clustering of weight vectors in language models (LMs) to construct expert groups. Technically, we treat the linear layer's weight matrix $\mathbf{W}$ as a set of d_{out} vectors, each of dimension d_{in}. These vectors are then partitioned into N clusters, analogous to MoE experts.

E.1 Implementation

As detailed in § 3.2, our method encompasses four core processes in cluster-based implementation. During the data forward phase, the product magnitudes of the weight vectors are computed and tracked. Subsequently, in the second phase, MIGU caches these magnitudes and employs an L1-norm normalization to derive a gradient mask. This mask is pivotal for modulating the gradients in the subsequent phases. The third phase involves the standard backpropagation to calculate the gradients of the parameters. Finally, in the fourth phase, the earlier computed gradient mask is applied to the obtained

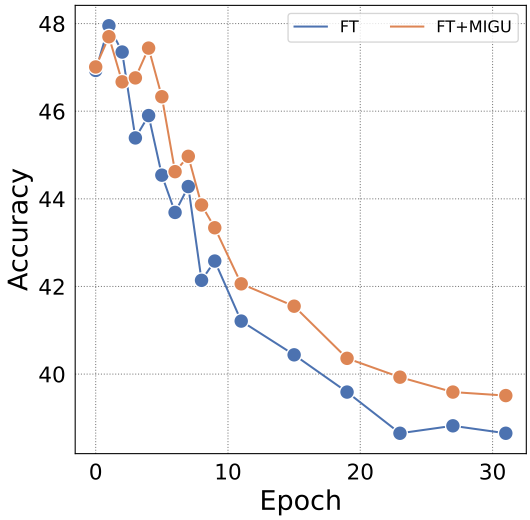


Figure 12: Accuracy on ARC-Challenge (Clark et al., 2018), evaluating on Llama-2-7B by MIGU with LoRA and valinna LoRA instruct tuning on Magicoder-Evol-Instruct-110k (Wei et al., 2024).

gradients, ensuring a modulated update of the parameters. This modulation is consistent within each cluster, thereby maintaining the integrity of the expert groupings and enhancing the model's learning efficacy.

We explored two distinct clustering strategies:

- Weight Cluster Combination: The weight vectors are clustered into N groups based on their proximity in the weight space.
- Co-magnitude Guided Combination: Using a subset of the dataset, we group weight vectors into clusters based on the similarity of their product magnitudes.

E.2 Result & Analysis

The outcomes of two distinct clustering approaches, alongside our implementation within LoRA, are illustrated in Figure 18. It is evident that, except for the second order, the "Weight Cluster" method surpasses the 'No Cluster' approach, which does not employ explicit clustering. However, the 'No Cluster' method demonstrates superior performance across the remaining orders, highlighting its robustness and effectiveness. Nonetheless, the other two explicit clustering techniques still significantly outperform the baseline vanilla continual learning LoRA, indicating their potential for further exploration.

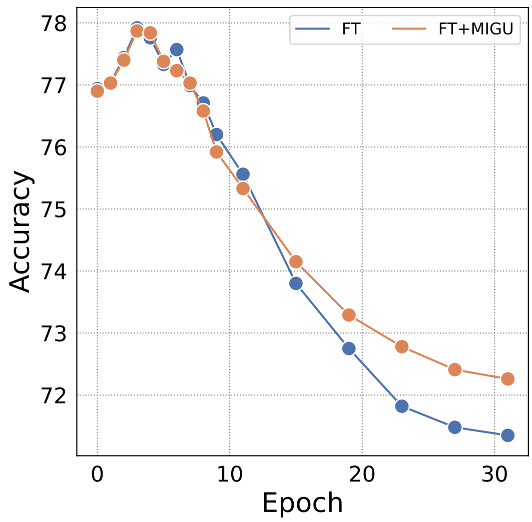


Figure 13: Accuracy on HellaSwag (Zellers et al., 2019), evaluating on Llama-2-7B by MIGU with LoRA and valinna LoRA instruct tuning on Magicoder-Evol-Instruct-110k (Wei et al., 2024).

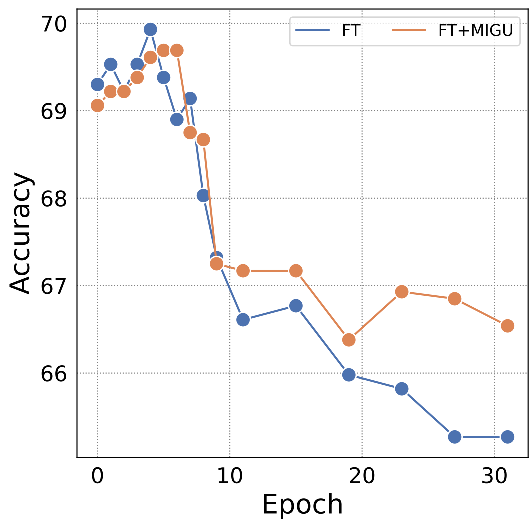
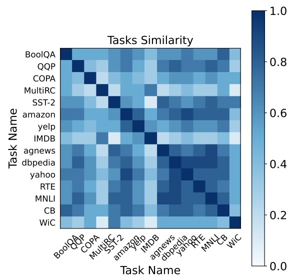
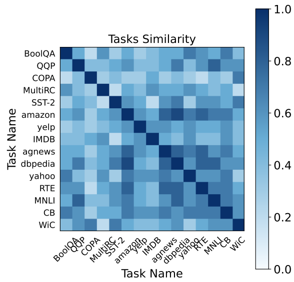


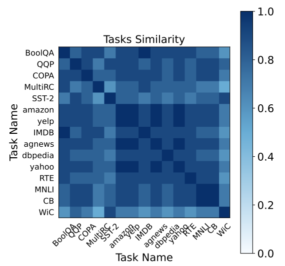
Figure 14: Accuracy on Winogrande (Sakaguchi et al., 2019), evaluating on Llama-2-7B by MIGU with LoRA and valinna LoRA instruct tuning pre-trained on Magicoder-Evol-Instruct-110k (Wei et al., 2024).



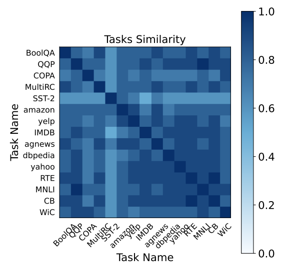
(a) full tuning



(b) full tuning with MIGU

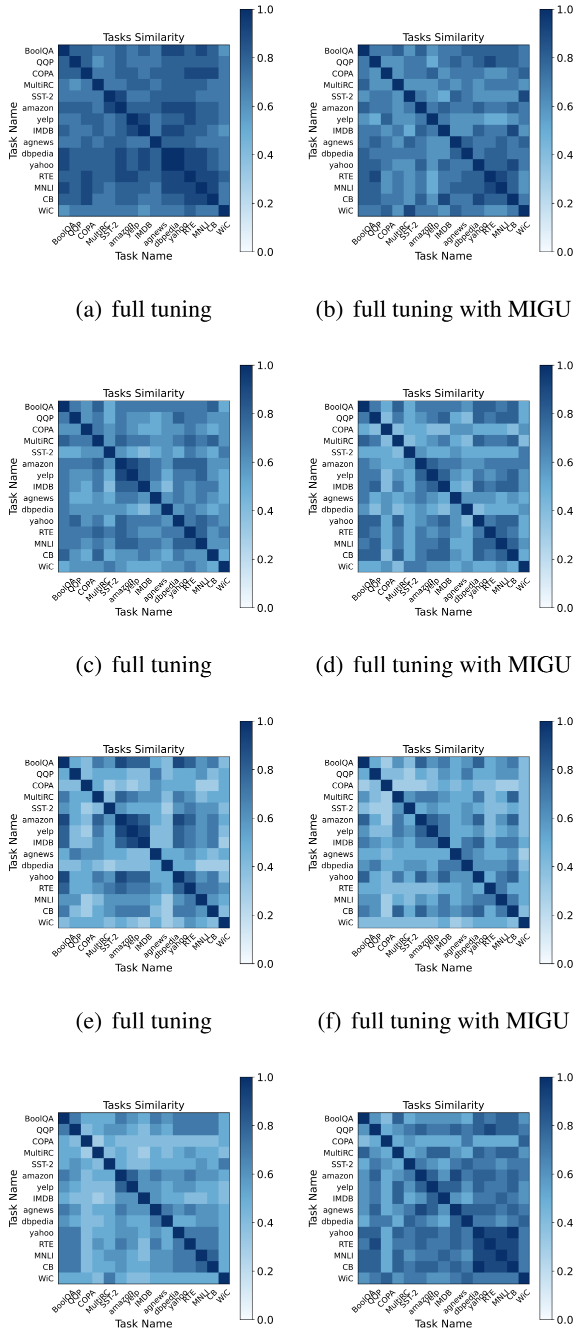


(c) full tuning



(d) full tuning with MIGU

Figure 15: The product magnitude distribution similarity of different tasks.



(a) full tuning (b) full tuning with MIGU

(c) full tuning (d) full tuning with MIGU

(e) full tuning (f) full tuning with MIGU

(g) full tuning (h) full tuning with MIGU

Figure 16: The product magnitude distribution similarity of different tasks.

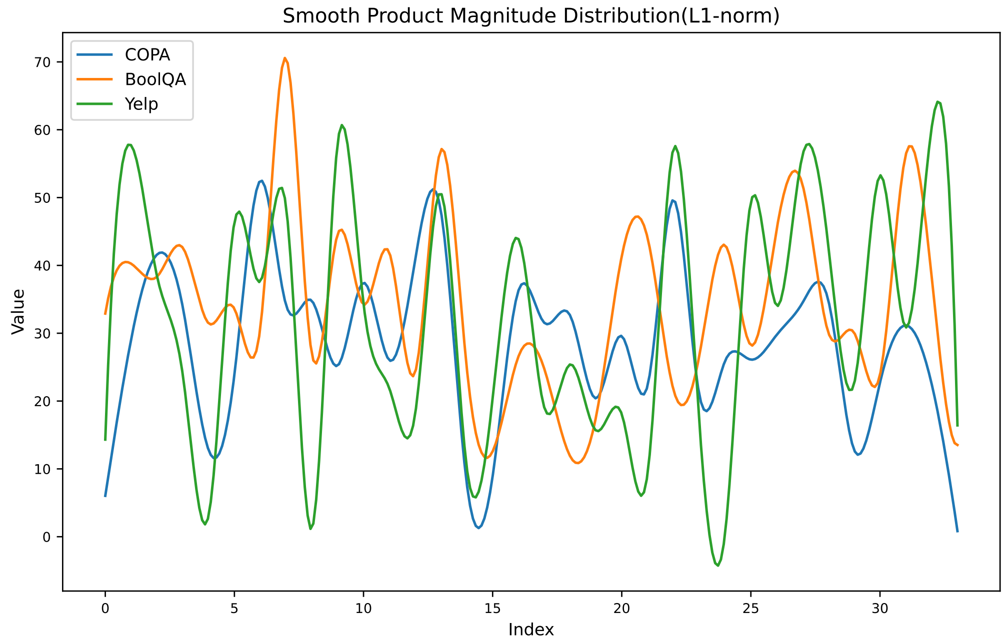


Figure 17: The Magnitudes distribution of COPA sample, BoolQA sample, and Yelp sample on the first linear layer of 23-th FFN layer of T5-large model.

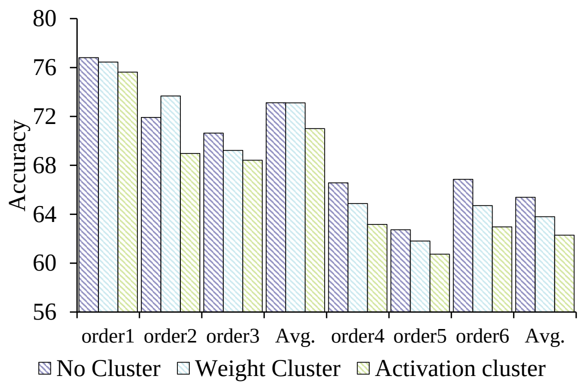


Figure 18: Ablation on LoRA modularity design