Particles Observed in Loss Variations with Model Pre-Scaling

Anonymous ACL submission

Abstract

The latest advancements in foundational large language models (LLMs) have challenged the widely recognized scaling laws, primarily manifesting in the reinterpretation of the relationship between model scale, data scale, and model capabilities. This paper proposes a novel research perspective by treating the model's holistic weights as a system variable. Through preliminary subtle scaling of the model during supervised fine-tuning (SFT) — a method referred to as pre-scaling — we systematically investigate the relationship between performance evolution and model variations. Building on this approach, we conduct extensive experiments across various pre-trained language models (PLMs), revealing the discrete features of the model: loss particles and output particles. Through empirical investigation and theoretical analysis, we characterize the fundamental process and statistical properties of particle fission during SFT. According to the inherent properties of output particles, the coupling relationship between these particles and sample importance is established. Based on this insight, we propose a simple and efficient data selection method named Pre-Scaling Pruning (PSP), which comprises two strategies: $\mathrm{PSP}_{\mathrm{one-shot}}$ and $\mathrm{PSP}_{\mathrm{zero-shot}}$. Notably, at a pruning ratio of 50%, the data subset selected by $\mathrm{PSP}_{\mathrm{one-shot}}$ achieves a higher average GLUE score than the full dataset, demonstrating that high-quality data subsets can not only reduce computational overhead but also enhance the model's generalization capability.

1 Introduction

Large language models (LLMs) have emerged as a breakthrough in the field of natural language processing (NLP), exemplified by models such as PaLM (Chowdhery et al., 2023), GPT-4 (OpenAI, 2023), and LLaMa (Touvron et al., 2023). These models, enriched with extensive world knowledge (Zhou et al., 2024; Gekhman et al., 2024), have significantly improved performance across various downstream tasks, including complex reasoning. A crucial research focus since the advent of LLMs has been how to effectively perform supervised fine-tuning (SFT) to further unlock their latent reasoning capabilities (Zhang et al., 2024b).Influenced by the scaling law (Kaplan et al., 2020), early research was largely driven by continuous expansion of model sizes. As research progressed, different scaling principles and phenomena (Xiao et al., 2024; Muennighoff et al., 2025) were proposed. It was observed that even with a substantial reduction in model size (Zhang et al., 2024a; Zhao et al.) and a continuous refinement of post-training data (Wang et al., 2023; Yu et al., 2024), models could still maintain or even surpass existing baselines.

From a broader perspective, the advancement of model reasoning capabilities can be viewed as an iterative process of scaling up and compressing model size while progressively refining fine-tuning datasets. Therefore, it is crucial to systematically investigate the relationships among generalization, scaling variations, and data selection. Early studies provided valuable insights, suggesting that generalization is often associated with models exhibiting lower weight norms (Zhang et al., 2018; Ghiasi et al., 2024; Kobayashi et al., 2024; Bos and Chug, 1996; Krogh and Hertz, 1991). Techniques such as model pruning (Tang et al., 2024) and L_2 regularization based on weight decay (Xie et al., 2024) have been effective in improving generalization. Additionally, data pruning has been introduced to enable models to learn more essential features with fewer training samples.

However, studying the relationships between model size, data pruning, and generalization in isolation presents significant limitations. In particular, existing progress in data pruning (Fayyaz et al., 2022) often relies on multiple proxy models, leading to high computational costs. Although later research (Attendu and Corbeil) proposed an

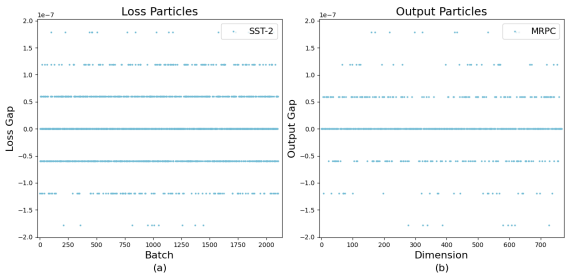

Figure 1: Loss particles (a) and output particles diagram (b). Figure (a) shows the loss particles of all batches of SST-2; Figure (b) shows the output particles of a single sample randomly selected in MRPC on 768 dimensions. They are obtained based on the loss difference and output difference of BERT-base before and after slight compression.

improved dynamic selection method, it requires data filtering throughout the entire training process, increasing the complexity of data selection.

We introduce a research prototype that prioritizes scaling adjustments, applying a subtle overall reduction in model size before analyzing the relationship between sample characteristics and generalization during SFT. This novel perspective reveals many intriguing properties within models. We observe the emergence of the discrete features in models after subtle scaling adjustments, including loss particles and output particles, as shown in Figure 1. Furthermore, we identify variations and correlations in output particles characteristics across different samples under subtle scaling. Not all training samples contribute equally to enhancing model generalization (Yang et al., 2022). Leveraging this observation, we propose Pre-Scaling Pruning (PSP), a novel method that utilizes output particles for data selection. PSP helps models identify high-quality data subsets, significantly reducing data evaluation costs and training overhead.Our contributions are:

- We discovered that pre-trained language models (PLMs) exhibit discrete characteristics fine-grained scaling — loss particles and output particles. Through empirical research and theoretical analysis, we provided a general process and statistical characteristics of particle splitting in SFT.

- We propose a simple and efficient data selection method named **Pre-Scaling Pruning**, which comprises two strategies : $\text{PSP}_{\text{one-shot}}$ and $\text{PSP}_{\text{zero-shot}}$. Experimental results show that the data subset selected by $\text{PSP}_{\text{one-shot}}$ achieves a higher average GLUE score than the full dataset at a pruning rate of up to 50%. $\text{PSP}_{\text{zero-shot}}$, using 50% of the SST-2 data, also produces results comparable to the full dataset, demonstrating the high quality and efficiency of the data subset selected by output particles.

- Based on the "pre-scaling" method, we established a technical route centered on changes in model weights. Along this route, starting with fine changes in the model, we can discover more fundamental microscopic features. For example, the sample particles manifest in the model's macro response to the loss. The "pre-scaling" method requires minimal resource overhead and can be easily integrated into the detailed research of related fields, with broad applicability and effectiveness.

2 Related Work

Recently, the quality of data has become particularly important in the context of LLMs (Zhao et al.; Minaee et al., 2024). During training, LLMs typically involve two fundamental steps (Ouyang et al., 2022): pre-training on large-scale corpora (Radford, 2018; Devlin, 2018; Raffel et al., 2020; Touvron et al., 2023) and fine-tuning on instruction datasets (Mishra et al., 2022; Sanh et al.; Longpre et al., 2023; Muennighoff et al., 2024) or other downstream NLP tasks. Training on high-quality data can lead to stronger performance (Du et al., 2022).

Importance of Pre-training Data. The objective of pre-training is to train a "general-purpose" model, which requires vast amounts of text. However, it is well-known that raw text data from the internet can contain large amounts of template text, error messages, and offensive content (Raffel et al., 2020; Touvron et al., 2023; Elazar et al.). Therefore, removing redundant and harmful data while retaining "high-quality" data (Conneau and Lample, 2019; Raffel et al., 2020; Wenzek et al., 2020; Gao et al., 2020; Rae et al., 2021; Nystrom et al., 2022) is crucial for improving the efficiency of the pre-training phase and reducing training costs. Among the approaches, heuristic methods for data quality filtering are widely adopted. For example, (Raffel et al., 2020) removes lines that do not end with terminal punctuation and lines with fewer than four words, while (Penedo et al., 2023) eliminates

lines predominantly composed of uppercase or numerical characters.

Importance of Instruction Tuning Data. Typically, models acquire basic language abilities during pre-training. Instruction tuning, a primary post-training paradigm, aims to address the mismatch between the distribution of pre-training data and downstream use cases. While instruction tuning primarily relies on large datasets, studies such as LIMA (Zhou et al., 2024) demonstrate that data quality is more important than quantity. An increasing number of works have focused on the quality and efficiency of instruction datasets (Köpf et al., 2024; Muennighoff et al.; Zhuo et al., 2024; Lu et al., 2023; Lian et al., 2023). The IFD (Li et al., 2024) method significantly outperforms the Alpaca model by using only about 5% of the Alpaca dataset, also surpassing the WizardLM model by approximately 10%. (Li et al., 2023) propose the Nuggets framework, implementing a dual-phase approach that leverages the disparity between one-shot and zero-shot scores to compute a definitive score for each instruction.

NLP Classification Tasks. In addition to instruction tuning, post-training of LLMs also includes other NLP classification tasks, such as sentiment analysis and natural language inference. However, there has been limited research on the quality of datasets for these NLP tasks. (Fayyaz et al., 2022) adjusted GraNd and EL2N (Paul et al., 2021) for PLMs and applied them to NLP tasks. However, these methods require training multiple proxy models and conducting numerous experiments to estimate more accurate scores, which adds considerable cost—sometimes even exceeding the training time on large-scale datasets. (Attendu and Corbeil) leverages an EL2N metric extended to the joint intent and slot classification task, followed by an initial fine-tuning phase on the full training set. However, they need to score data in multiple fine-tuning stages, selecting different data subsets for training, further complicating the data selection process.

3 The Particle Phenomenon in Loss Variations

Given a downstream task dataset $D = \{z_1, z_2, \ldots, z_N\}$ containing N samples, where $z_i = (x_i, y_i)$. Here, x_i represents the input data, y_i is the truth label, and $i = 1, 2, \ldots, N$. We explore the generalization of a pre-trained model with a feed-forward head attached to downstream tasks. Different from existing work, our research motivation is to perform subtle scaling adjustments on the model θ in advance and examine the corresponding changes in loss based on the changing θ, attempting to generally characterize the correlation between the two.

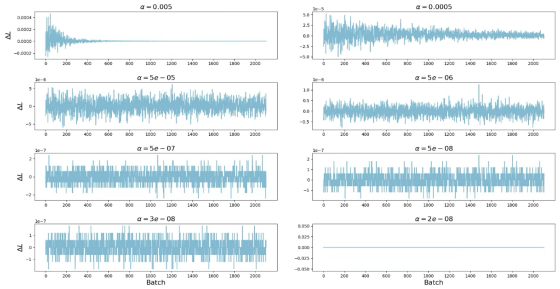

Figure 2: Distribution of model loss differences on each batch under different compression scales.

3.1 Discovery of Loss Particles

Since appropriate compression of model parameters often leads to an improvement in generalization ability (Zhang et al., 2018; Ghiasi et al., 2024; Kobayashi et al., 2024; Bos and Chug, 1996; Krogh and Hertz, 1991), we attempt to compress the model. That is, under the action of the hyperparameter α, we examine the loss difference ΔL generated on a batch before and after the model undergoes the change $\theta - \alpha\theta$, where $\alpha \in (0, 1)$. For the first time, we discovered that under fine-scale ($10^{-6} \sim 10^{-8}$) compression, ΔL exhibits particle characteristics, as shown in Figure 2. As can be seen from Figure 2, for most of the batch data in SST-2, ΔL undergoes discrete jumps when $\alpha = 3 \times 10^{-8}$. This means that minor model differences do not result in continuous loss values. The size of the loss particle represents the degree to which the model can distinguish fine loss differences. The larger the particle, the lower the model's ability to distinguish fine sample differences.

ΔL can be expressed as:

$$\Delta L = \begin{cases} 0, & \Delta\theta < \Delta\theta_{\alpha_{flip}} \\ kL_\delta, & \Delta\theta \geq \Delta\theta_{\alpha_{flip}} \end{cases} \quad (1)$$

$$L_\delta = \min_{batch \in D} (L(\theta, batch) - L(\theta - \Delta\theta_{\alpha_{flip}}, batch)) \quad (2)$$

where α_{flip} is the jump parameter, L_δ is the loss particle, $\Delta\theta_{\alpha_{flip}} = \alpha_{flip}\theta$, and k is a discrete integer variable. As α continuously in-

creases from α_{flip}, ΔL successively takes irregular jump values at jump points such as $\alpha_{flip(1)}$, $\alpha_{flip(2)}, \ldots, \alpha_{flip(n)}$. In the interval $\Delta\alpha_{flip} = [\alpha_{flip(i+1)} - \alpha_{flip(i)}]$, ΔL remains unchanged.

We define L_δ as the minimum batch loss particle. According to our findings, the size of L_δ is related to the form of the objective function $o \in \mathcal{O}$ and the specific task $t \in \mathcal{T}$. For related research, see Appendix B.1 for details.

It can be seen from Figure 1 that the ΔL values generated by different batches at the jump point α_{flip} are not the same. In the same dataset, ΔL on different samples is called the loss particle length, denoted as ΔL_{batch}. Assume that $\Delta L_{batchx} \neq \Delta L_{batchy}$, but $|\Delta L_{batchx} - \Delta L_{batchy}|$ has an integer multiple relationship with L_δ. In addition, the loss differences generated by different batches for model compression have positive and negative values.

As α increases, L_δ remains invariant and indivisible. For ΔL generated in the weight amplification direction, the above formula can still be used to summarize. Different from compression, the jump points α_{flip} in the amplification direction are generally larger, as shown in Appendix B.5.

3.2 Particle Fission during Fine-Tuning

Different from pure weight scaling, the model θ iteration during the fine-tuning process includes the compression and expansion of some weights in different proportions. Experiments have found that the shape of L_δ undergoes a binary fission change similar to that in the field of biological cells. The L_δ during the fine-tuning process can be regarded as an individual that has undergone τ binary fission cycles, denoted as $L_{\delta-\tau} = \frac{1}{2^\tau} L_\delta$, where the empirical τ is on the same scale as the number of epochs. In Figure 3, we show the fission process of L_δ when $\tau = 1$. The fission process of $L_{\delta-\tau}$ is indicated by the red arrow in Figure 3. Appendix B.2 provides a more detailed description of the fission of loss particles.

3.3 Dynamic Characteristics of Output Particles

Obviously, due to the continuous mathematical properties of the linear layer function, the reason for the emergence of L_δ lies in the pre-trained model itself, rather than the feed-forward head added later. The high-dimensional output of the model should be discrete, which ultimately leads to the particle characteristics of the loss. We

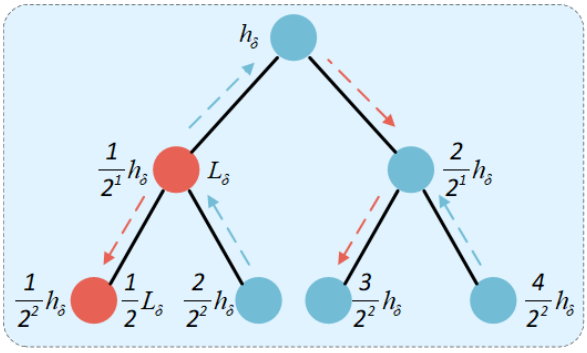

Figure 3: Binary fission of loss particles (L_δ) and binary tree range distribution of output particles(h_δ). The red arrows and blue arrows represent the fission and merging processes respectively.

present Property 1 and prove that the model output is also discrete. Let the output of the model be $h = f_{pre}(x)$, $h \in R^d$, where d is the dimension of the output of the pre-trained model. Similar to the case of loss particles, we experimentally verify the discreteness of h before and after fine-tuning. More details are shown in Appendix B.4.

Property 1. *Consider a downstream task in an n-class classification problem, where the model output is $h = [h_1, h_2, \ldots, h_d]$, and the corresponding classification labels are$Y = [y_1, y_2, \ldots, y_n]$. $\hat{Y} = softmax(h) = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n]$, where $\hat{y}_i$ is the predicted class probability. Suppose the loss generated by a certain batch during model inference is L. If there exists a non-zero minimum loss particle L_δ, then the model output vector h generated for this batch must be a discrete vector.*

The proof is in Appendix A.1.

Let the i-th dimension of the model output be $h^{(i)}$. Under the fine penalty hyperparameter α, the difference of $h^{(i)}$ will jump, that is:

$$\Delta h^{(i)} = \begin{cases} 0, & \Delta\theta < \Delta\theta_{\alpha_{flip}} \\ kh_\delta, & \Delta\theta \geq \Delta\theta_{\alpha_{flip}} \end{cases} \quad (3)$$

where α_{flip} is the jump parameter, $\Delta\theta_{\alpha_{flip}} = \alpha_{flip}\theta$, and k is a discrete integer variable. We define h_δ as the minimum sample output particle, denoted as $h_\delta = \min\{\Delta h^{(i)} | i = 1, 2, \ldots, d\}$.

The fission of loss particles during fine-tuning prompts us to further examine the dynamics of $\Delta h^{(i)}$.We found that:

1. The length change of the minimum sample output particle h_δ is related to the fine-tuning process. As the fine-tuning progresses, h_δ undergoes a binary fission change similar to $L_{\delta-\tau}$. While

$h_{\delta-\tau}$ continues to fission downward, the overall change trend of $\Delta h^{(i)}$ in each dimension of the sample is related to the downstream task. As can be seen from Figure 4(a), $\Delta h^{(i)}$ of MRPC mostly shows a divergent trend, while the output particles of SST-2 mostly show a convergent trend. For batch data, for a typical sample x with $x = \min\{h_\delta^{(i)}(x_j)|i = 1, 2, \ldots, d, x_j \in D_{batch}\}$, Property 2 proves that L_δ is equivalent to h_δ. Property 3 gives that the actually observed binary fission of h_δ (including L_δ) conforms to the statistical expectation of a uniform distribution.

Property 2. *Let the minimum loss particle of the pre-trained model before fine-tuning be L_δ, and the minimum output particle of the sample x be h_δ. Then $|L_\delta| = |h_\delta|$, where $|\cdot|$ represents the particle length.*

The proof is in Appendix A.2.

Property 3. *Let the minimum output particle of the sample before fine-tuning be h_δ. Suppose that after τ fissions at the end of fine-tuning, the minimum output particle becomes h_{δ_τ}. If the sequence of fission particle values $|h_{\delta_1}|, |h_{\delta_2}|, \ldots, |h_{\delta_\tau}|$ is uniformly distributed successively in the ranges $[0, |h_\delta|]$, $[0, |h_{\delta_1}|]$, …, $[0, |h_{\delta_{\tau-1}}|]$, then $E[|h_{\delta_i}|] = \frac{1}{2^i}|h_\delta|$, where $i \in [1, 2, \ldots, n]$.*

The proof is in Appendix A.3.

2. The characteristics of samples are significantly different. Taking the SST-2 dataset as an example, we randomly select 20 samples from the training set and observe their performance on the output particles of the pre-trained model before fine-tuning, as shown in Figure 4(b). It can be seen from the figure that the discrete degree of the output of some samples is relatively large, while that of other samples is relatively more concentrated. This reflects the characteristics and differences of samples, that is, the pre-trained model has different sensitivities to different samples on different features.

3. Different dimensions have corresponding jump sequences such as $\alpha^{(i)}_{flip(1)}, \alpha^{(i)}_{flip(2)}, \ldots, \alpha^{(i)}_{flip(j)}$, and each maintains an invariant interval such as $\Delta\alpha^{(i)}_{flip(j)} = [\alpha^{(i)}_{flip(j+1)} - \alpha^{(i)}_{flip(j)}]$. The jump sequences $\alpha^{(i)}_{flip(j)} \neq \alpha^{(i')}_{flip(j)}$, and the value changes of k are different.

4. During fine-tuning, $h_{\delta-\tau}$ and $L_{\delta-\tau}$ maintain equivalence. In the same round of fine-tuning, the output changes of all dimensions of each sample

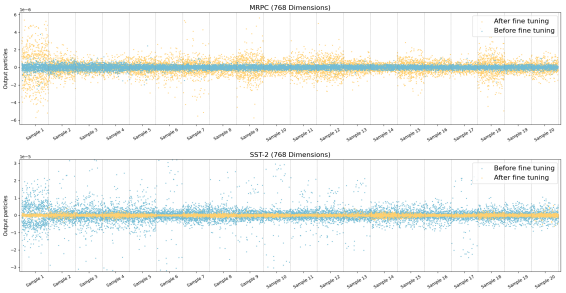

Figure 4: Output particles' distribution of different samples before and after fine-tuning BERT-base.

can still be summarized by Equation 3, except that h_δ in it becomes $h_{\delta-\tau}$. The degree of fission is related to the model and the task set. Property 4 gives an approximate calculation of batches that have not undergone fission during the fine-tuning process.

Property 4. *Assume that the probability of occurrence of particle fission whithin a sample is p, where a batch $D_{batch} \in D$, $|D_{batch}| = b$, $N = |D|$. After τ fissions, the probability of randomly selecting a non-fissioned batch from the dataset is $p_{batch} = \frac{C^b_{N_\tau}}{C^b_N}$, where $N_\tau = N \cdot (1-p)^\tau$.*

The proof is in Appendix A.4.

4 Sample Selection with Pre-Scaling on Model

In the previous section, we first performed a slight compression on the model weight θ and then examined the change in loss under this slight compression. As a result, we discovered the loss particle L_δ. Superficially, L_δ seems similar to the approximate expression of the gradient $\Delta L/\Delta\theta$, but in fact, there is an essential difference between them. L_δ is obtained on the premise of compressing the entire θ, which is a response of the loss to the overall change of the model. While the gradient is the response of the loss to the change of a single weight. Especially in the field of NLP, there are significant limitations in directly using methods based on the first - order response of the loss (such as gradients) to characterize features. Furthermore, from Property 1, we obtained the equivalence between the minimum batch particle and the minimum loss particle. They are not only numerically equal but also maintain this equivalence throughout the entire fine - tuning stage. The method of pre-scaling on θ can profoundly describe the relationship between samples and loss. Thus, we proposed the

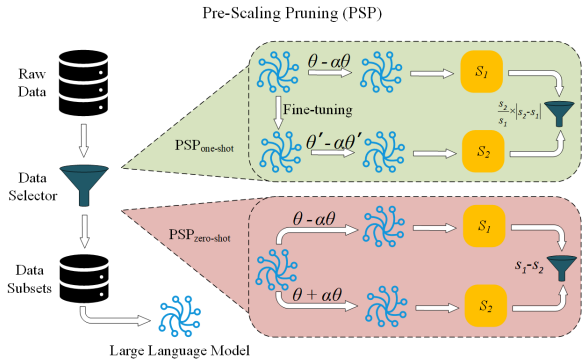

Figure 5: The overview of proposed PSP. Two data selection schemes are considered: $\text{PSP}_{\text{one-shot}}$ and $\text{PSP}_{\text{zero-shot}}$.

PSP method for sample selection. Section 4.1 provides an overview of PSP, including two schemes: $\text{PSP}_{\text{one-shot}}$ and $\text{PSP}_{\text{zero-shot}}$. Section 4.2 details the two schemes.

4.1 Overview

The overview of proposed PSP is provided in the Figure 5. Based on the pre-scaling operations, two data selectors are designed, including $\text{PSP}_{\text{one-shot}}$ and $\text{PSP}_{\text{zero-shot}}$. Detailed descriptions of the two solutions will be presented in Section 4.2. In addition, the pre-scaling operations performed on the model θ includes slight compression, i.e. $\theta - \alpha\theta$, and slight expansion, i.e. $\theta + \alpha\theta$. The coefficient α is the scaling factor.

4.2 Pre-Scaling Pruning

For a sample x, we construct the output particle vector $v(x)$ as follows:

$$v(x) = (\Delta h^{(1)}, \Delta h^{(2)}, \ldots, \Delta h^{(d)}) \tag{4}$$

$$\|v(x)\|_1 = \sum_{i=1}^{d} |\Delta h^{(i)}| \tag{5}$$

$v(x)$ reflects the sensitivity of the model to the sample x in different output dimensions, and the vector $v(x)$ is sensitive to fine - tuning. Let s represent the sensitivity score of the model to the sample x, and $s = \|v(x)\|_1$, where $\|\cdot\|_1$ represents the L_1 - norm value of the vector. According to the changes of s in different situations, we proposed two schemes for the PSP method.

$\text{PSP}_{\text{one-shot}}$. 1. Use D to fine - tune the model once, and obtain the output particles $v_1(x_i)$ and $v_2(x_i)$ of the sample $x_i \in D$ before and after fine - tuning respectively. 2. Calculate the sensitivity scores of the model to the sample x_i before and after fine - tuning, denoted as s_1 and s_2 respectively. 3. Construct the scoring function $\phi(x_i)$.

The key is how to establish the scoring function $\phi(\cdot)$ based on s_1 and s_2, and then rank the samples according to the pruning criterion expressed by the function $\phi(\cdot)$ to obtain the sequence of retained samples. Empirically, the output particles of some samples show a "divergent" trend, with $s_1 < s_2$, while those of other samples show a "compressed" trend, with $s_1 > s_2$. If we measure the importance of samples only through s_1 or s_2, it will reduce the generalization ability of the model. Therefore, the information of s_1 and s_2 needs to be comprehensively considered. We construct the scoring function $\phi(x_i)$ for the sample $x_i \in D$ by using the difference and ratio between s_1 and s_2:

$$\phi(x_i) = \frac{s_2}{s_1} \cdot |s_2 - s_1| \tag{6}$$

$\phi(\cdot)$ reflects a relative stability of the model's sensitivity to certain features of the sample before and after fine - tuning. This stability reflects the ability of the pre-trained model in knowledge acquisition and also provides an explanation for the fine - tuning itself.

Due to the large number of task sets, it is not advisable to simply apply Equation 6 in practical applications because different tasks have different biases towards the ratio or difference. For complex situations, we propose a gold-panning selection method. The specific details are analyzed in Section 5.5.

$\text{PSP}_{\text{zero-shot}}$. The model also shows output particles under slight compression, as detailed in Appendix B.4. Empirically, the model is not sensitive to slight compression. $\text{PSP}_{\text{zero-shot}}$ constructs $s_1 = \|v_1(x_i)\|_1$ and $s_2 = \|v_2(x_i)\|_1$ for the sample $x_i \in D$ by slightly compressing and expanding the model, that is, $\theta - \alpha\theta$ and $\theta + \alpha\theta$. The scoring function $\phi(x_i)$ of $\text{PSP}_{\text{zero-shot}}$ is as follows:

$$\phi(x_i) = s_1 - s_2 \tag{7}$$

5 Experiments

5.1 Experimental Setup

Baselines. We compared five methods to prove the effectiveness of output particles. A) Full training set: The baseline method of standard training using the full training set. B) Random pruning: Randomly delete data points with ratio $(1 - \rho)$ at once and use the remaining data for subsequent training.

C) GraNd: Use the expected value of the gradient norm of the model as a measure of the importance of samples. Consistent with the settings in (Fayyaz et al., 2022), we compute the GraNd scores only for the randomly initialized classifier layer on top of the PLMs. D) EL2N: An estimated variant of GraNd. The importance of each sample is the average EL2N scores of five independently trained models, and then we retain the data with a higher score for subsequent training. E) Single EL2N: First, train with the full dataset for E_{pre} epochs, and then use the EL2N scores to measure the importance of samples. Retain the data subset with a higher score for subsequent training. F) $\text{PSP}_{\text{one-shot}}$: The proposed method which trains on the full train-set for τ epochs, followed by single-data pruning using $\text{PSP}_{\text{one-shot}}$ scores. The implementation details of each method are shown in Appendix C.1.

Tasks & Datasets. We used eight datasets from the GLUE (Beven and Binley, 2014) benchmark. The GLUE benchmark is a suite designed to evaluate the performance of natural language understanding (NLU) systems. Table 6 in Appendix D provides the general characteristics of the datasets we used.

Models. We used two PLMs: BERT-base-uncased (110M parameters) (Devlin, 2018) and RoBERTA-base (125M parameters) (Liu, 2019). The pre-trained models were all provided by Hugging Face (Wolf, 2019). For BERT and RoBERTA, their high-dimensional outputs are both 768-dimensional, which is exactly the feature vector for calculating output particles.

5.2 Results of the $\text{PSP}_{\text{one-shot}}$ Method

RoBERTA-base. The experimental results of $\text{PSP}_{\text{one-shot}}$ on RoBERTA-base (Liu, 2019) a are reported in Table 1. The GraNd and EL2N scores are the average scores of five independently trained proxy models. Since STS-B is a regression task, we did not calculate its EL2N and Single EL2N scores. In most cases, the results of $\text{PSP}_{\text{one-shot}}$ are comparable to those of training with the full dataset. It is worth noting that for the RTE task with high learning difficulty, $\text{PSP}_{\text{one-shot}}$ exceeds the full dataset by 2.89%. The subsets selected by GraNd and EL2N (Paul et al., 2021) scores show a phenomenon of difficult fitting, which also indicates the effectiveness of $\text{PSP}_{\text{one-shot}}$ in the NLP field.

In addition, to further verify the effectiveness of $\text{PSP}_{\text{one-shot}}$, we conducted experiments on other models such as BERT-base pre-trained model (Devlin, 2018). And the same efficient performance is obtained. The detailed results can be found in the Appendix C.2.

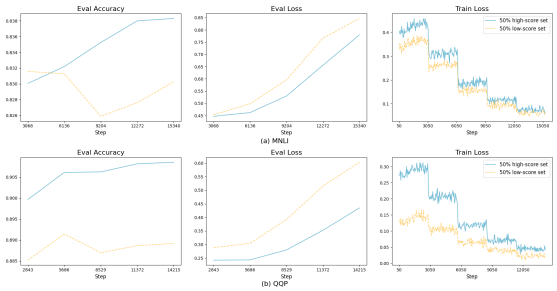

Figure 6: The results of fine-tuning BERT using the 50% of data with higher (blue solid) and lower (yellow solid) scores selected by $\text{PSP}_{\text{one-shot}}$ on MNLI (a) and QQP (b) respectively. The three graphs in each row from left to right are the evaluation accuracy, evaluation loss, and training loss.

5.3 Results of the $\text{PSP}_{\text{zero-shot}}$ Method

We evaluated the effectiveness of $\text{PSP}_{\text{zero-shot}}$ on the SST-2 and MNLI tasks using the BERT-base model. As can be seen from Table 2, by selecting only 50% of the data, $\text{PSP}_{\text{zero-shot}}$ can achieve the same result as using the full dataset on the SST-2 task. Remarkably $\text{PSP}_{\text{zero-shot}}$ selects data without any training on the model and is independent of data labels.

5.4 Generalization Analysis

To further analyze the impact of the data selected by $\text{PSP}_{\text{one-shot}}$ on the model's generalization ability, the dataset was divided into two subsets: D_{high} and D_{low}. D_{high} represents the top 50% of data with higher scores selected by $\text{PSP}_{\text{one-shot}}$, and D_{low} represents the remaining 50% of data with lower scores. As can be seen from Figure 6, for MNLI (a) and QQP (b), the training loss of D_{high} is higher than that of D_{low}. However, the evaluation loss shows the opposite trend. And for QQP (b), the evaluation accuracy of the model trained on D_{high} is always higher than that of the model trained on D_{low}. This indicates that the subset selected by $\text{PSP}_{\text{one-shot}}$ can not only ensure the final convergence of the model but also avoid the model falling into a local optimum in the early stage, enabling the model to significantly improve its generalization ability even with only 50% of the data.

Dataset	MRPC	RTE	CoLA	STS-B	SST-2	QNLI	MNLI	QQP	AVG
FULL	88.73	75.09	61.81	90.80	94.38	92.71	87.71	91.52	85.34
Random	87.38	74.37	55.76	89.84	94.09	91.91	86.70	90.30	83.79
GraNd	85.05	50.54	55.92	89.11	93.35	92.31	86.64	90.53	80.43
EL2N	85.66	48.38	56.82	–	93.58	92.56	86.97	90.63	79.23
EL2N*	**89.33**	76.53	61.73	–	94.27	92.36	86.81	**90.63**	84.52
$\text{PSP}_{\text{one-shot}}$	88.97$^{\uparrow}$	**77.98**$^{\uparrow}$	**63.84**$^{\uparrow}$	**90.71**$^{\uparrow}$	**94.61**$^{\uparrow}$	91.82$^{\uparrow}$	**87.24**$^{\uparrow}$	90.58$^{\uparrow}$	**85.72**

Table 1: Comparison results of $\text{PSP}_{\text{one-shot}}$ and other methods on GLUE tasks using RoBERTA-base. The retention ratio of all datasets is 50%. The superscript "↑" indicates selecting data points with larger scores. To exclude the influence of random factors, all experimental results are the average of five independent runs. "FULL" represents Full training, "EL2N*" represents Single EL2N.

Dataset	SST-2	MNLI
FULL	92.78	84.37
Random	92.77	83.14
GraNd	**92.83**	82.24
EL2N	92.78	82.16
EL2N*	92.55	**83.45**
$\text{PSP}_{\text{one-shot}}$	92.78$^{\uparrow}$	83.3$^{\uparrow}$

Table 2: Comparison of $\text{PSP}_{\text{zero-shot}}$ and other methods. The superscript "↑" indicates selecting the 50% of data points with larger scores. "FULL" represents Full training, "EL2N*" represents Single EL2N.

task	$\rho_1(\frac{s_2}{s_1})$	$\rho_2(\lvert s2 - s1\rvert)$	ρ	Accuracy
QNLI	0.6	0.5	0.5	91.19
	0.5	0.4	0.4	**91.52**
	0.4	0.3	0.3	90.54
SST-2	0.6	0.5	0.5	92.55
	0.5	0.4	0.4	**93.35**
	0.4	0.3	0.3	92.89

Table 3: Gold-panning selection. We used the BERT-base pre-trained model, taking the QNLI and SST-2 tasks as examples. We conducted two-stage screening: 1) Select data with a ratio of ρ_1 using the $\frac{s_2}{s_1}$ score, and set the importance of unselected data points to 0; 2) Select data with a ratio of ρ_2 using $|s_2 - s_1|$, and the final retention ratio of data is $\rho = \rho_2$.

5.5 Gold-panning Selection

To verify whether there are samples in the data subset that are not conducive to improving the model's generalization ability, we used a strategy called "gold-panning selection". After obtaining the s and h scores described in Method 1 Specifically, after obtaining the s_1 and s_2 scores of each sample described in $\text{PSP}_{\text{one-shot}}$, the data is initially purified using indicator $\frac{s_2}{s_1}$, retaining data with the ratio ρ_1. Then we used $|s_2 - s_1|$ to select data with the target ratio ρ. We experimented with the results of two-stage purification under three ratios ρ, as shown in Table 3. We found that it is not always better to retain more data samples. For QNLI and SST-2, the model trained on 40% of the samples achieves higher accuracy than the one trained on 50% of the samples. This suggests that some samples in the 50% dataset may not be conducive to the model's learning, and removing them can enhance the model's generalization ability.

6 Conclusion

In this work, we first conduct a fine-grained scaling on the model and identify common discrete features of PLMs: loss particles and output particles. We then propose a novel metric to measure the sensitivity of PLMs to data and exploit changes in output particles to select high-quality data. Our method is applied to NLP classification tasks, achieving excellent performance with minimal overhead.

The "particle phenomenon" is a microscopic reflection of PLMs. Further exploration of loss particles and output particles may bring new enlightening perspectives to the fields of data pruning, model compression, and the interpretability research of LLMs.

Limitations

In this work, we identified the particle phenomenon in PLMs and leveraged output particles for data selection in NLP classification tasks. However, why

do particles emerge in PLMs but not in traditional deep networks? At present, we can only interpret this from the perspective of model intelligence, suggesting that minor variations in input samples are insufficient to elicit noticeable differences in model responses. Moreover, research on particles in simple SFT tasks may still be limited. Moving forward, we aim to conduct an in-depth investigation into this phenomenon within complex reasoning tasks in large models. In the future, we will further explore its implications in other pretraining and instruction-tuning tasks.

References

Jean-Michel Attendu and Jean-Philippe Corbeil. Nlu on data diets: Dynamic data subset selection for nlp classification tasks.

Keith Beven and Andrew Binley. 2014. Glue: 20 years on. *Hydrological processes*, 28(24):5897–5918.

Siegfried Bos and E Chug. 1996. Using weight decay to optimize the generalization ability of a perceptron. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 1, pages 241–246. IEEE.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. *Advances in neural information processing systems*, 32.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR.

Yanai Elazar, Akshita Bhagia, Ian Helgi Magnusson, Abhilasha Ravichander, Dustin Schwenk, Alane Suhr, Evan Pete Walsh, Dirk Groeneveld, Luca Soldaini, Sameer Singh, et al. What's in my big data? In *The Twelfth International Conference on Learning Representations*.

Mohsen Fayyaz, Ehsan Aghazadeh, Ali Modarressi, Mohammad Taher Pilehvar, Yadollah Yaghoobzadeh, and Samira Ebrahimi Kahou. 2022. Bert on a data diet: Finding important examples by gradient-based pruning. *arXiv preprint arXiv:2211.05610*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Zorik Gekhman, Gal Yona, Roee Aharoni, Matan Eyal, Amir Feder, Roi Reichart, and Jonathan Herzig. 2024. Does fine-tuning llms on new knowledge encourage hallucinations? *arXiv preprint arXiv:2405.05904*.

Mohammad Amin Ghiasi, Ali Shafahi, and Reza Ardekani. 2024. Improving robustness with adaptive weight decay. *Advances in Neural Information Processing Systems*, 36.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Seijin Kobayashi, Yassir Akram, and Johannes Von Oswald. 2024. Weight decay induces low-rank attention layers. *arXiv preprint arXiv:2410.23819*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2024. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36.

Anders Krogh and John Hertz. 1991. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7595–7628.

Yunshui Li, Binyuan Hui, Xiaobo Xia, Jiaxi Yang, Min Yang, Lei Zhang, Shuzheng Si, Junhao Liu, Tongliang Liu, Fei Huang, et al. 2023. One shot learning as instruction data prospector for large language models. *arXiv preprint arXiv:2312.10302*.

W Lian et al. 2023. Slimorca: An open dataset of gpt-4 augmented flan reasoning traces, with verification.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 364.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *International Conference on Machine Learning*, pages 22631–22648. PMLR.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. # instag: Instruction tagging for analyzing supervised fine-tuning of large language models. In *The Twelfth International Conference on Learning Representations*.

Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large language models: A survey. *arXiv preprint arXiv:2402.06196*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *60th Annual Meeting of the Association for Computational Linguistics, ACL 2022*, pages 3470–3487. Association for Computational Linguistics (ACL).

Niklas Muennighoff, Qian Liu, Armel Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro Von Werra, and Shayne Longpre. Octopack: Instruction tuning code large language models. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative representational instruction tuning. *arXiv preprint arXiv:2402.09906*.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*.

Andrew Nystrom, Chiyuan Zhang, Chris Callison-Burch, Daphne Ippolito, Douglas Eck, Katherine Lee, and Nicholas Carlini. 2022. Deduplicating training data makes language models better.

R OpenAI. 2023. Gpt-4 technical report. arxiv 2303.08774. *View in Article*, 2(5).

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. 2021. Deep learning on a data diet: Finding important examples early in training. *Advances in neural information processing systems*, 34:20596–20607.

Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. 2023. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*.

Alec Radford. 2018. Improving language understanding by generative pre-training.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

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(140):1–67.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, et al. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

Yehui Tang, Yunhe Wang, Jianyuan Guo, Zhijun Tu, Kai Han, Hailin Hu, and Dacheng Tao. 2024. A survey on transformer compression. *arXiv preprint arXiv:2402.05964*.

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*.

Zige Wang, Wanjun Zhong, Yufei Wang, Qi Zhu, Fei Mi, Baojun Wang, Lifeng Shang, Xin Jiang, and Qun Liu. 2023. Data management for large language models: A survey. *CoRR*.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Édouard Grave. 2020. Ccnet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012.

T Wolf. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Chaojun Xiao, Jie Cai, Weilin Zhao, Guoyang Zeng, Xu Han, Zhiyuan Liu, and Maosong Sun. 2024. Densing law of llms. *arXiv preprint arXiv:2412.04315*.

Zeke Xie, Zhiqiang Xu, Jingzhao Zhang, Issei Sato, and Masashi Sugiyama. 2024. On the overlooked pitfalls of weight decay and how to mitigate them: A gradient-norm perspective. *Advances in Neural Information Processing Systems*, 36.

Shuo Yang, Zeke Xie, Hanyu Peng, Min Xu, Mingming Sun, and Ping Li. 2022. Dataset pruning: Reducing training data by examining generalization influence. *arXiv preprint arXiv:2205.09329*.

Ruonan Yu, Songhua Liu, and Xinchao Wang. 2024. Dataset distillation: A comprehensive review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(1):150–170.

Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. 2018. Three mechanisms of weight decay regularization. *arXiv preprint arXiv:1810.12281*.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024a. Tinyllama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

Yulin Zhang, Yanhua Li, and Junhan Liu. 2024b. Unified efficient fine-tuning techniques for open-source large language models.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2024. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36.

Terry Yue Zhuo, Armel Zebaze, Nitchakarn Suppattarachai, Leandro von Werra, Harm de Vries, Qian Liu, and Niklas Muennighoff. 2024. Astraios: Parameter-efficient instruction tuning code large language models. *arXiv preprint arXiv:2401.00788*.

A Proofs

A.1 Proof of Property 1

Property 1. *Consider a downstream task in an n-class classification problem, where the model output is $h = [h_1, h_2, \ldots, h_d]$, and the corresponding classification labels are$Y = [y_1, y_2, \ldots, y_n]$. $\hat{Y} = softmax(h) = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n]$, where $\hat{y}_i$ is the predicted class probability. Suppose the loss generated by a certain batch during model inference is L. If there exists a non-zero minimum loss particle L_δ, then the model output vector h generated for this batch must be a discrete vector.*

Proof. Proof by contradiction. Assume that $\exists i \in [1, 2, \ldots d]$, and h_i is a continuous variable. Assume that in this task, the loss function defined by cross - entropy is $L = -\sum_{i=1}^{n} y_i \ln \hat{y}_i$. And a certain batch contains data of all n classes. Suppose there is a small change $\varepsilon \neq 0$ in h_i. Since h_i is continuous, the model output changes to $h' = [h_1, h_2, \ldots h_i + \varepsilon, \ldots, h_d]$. Then the resulting $\hat{y}'_i = \frac{e^{h_i+\varepsilon}}{\sum_{j=1, j\neq i}^{n} e^{h_j} + e^{h_i+\varepsilon}} \neq \hat{y}_i$, which contradicts the proposition assumption $L_\delta \neq 0$. Thus, the proof is completed.

A.2 Proof of Property 2

Property 2. *PLet the minimum loss particle of the pre-trained model before fine-tuning be L_δ, and the minimum output particle of the sample x be h_δ. Then $|L_\delta| = |h_\delta|$, where $|\cdot|$ represents the particle length.*

Proof. Continue to use the assumptions in Property 1,the loss function defined by cross-entropy is $L = -\sum_{i=1}^{n} y_i \ln \hat{y}_i$.

Suppose that under a fine penalty α, the output change of the model is Δh. From Equation 3, there exists an integer vector $k = [k_1, k_2, \ldots, k_d]$ such that $\Delta h = |h_\delta| \cdot [k_1, k_2, \ldots, k_d]$. Since $\hat{y}_i = \frac{e^{h_i}}{\sum_{j=1}^{n} e^{h_j}}$, the predicted probability after the change is $\hat{y}'_i = \frac{e^{h_i+\Delta h_i}}{\sum_{j=1}^{n} e^{h_j+\Delta h_j}}$, where $\Delta h_i = k_i|h_\delta|$. Then

$$\Delta L = -\sum_{i=1}^{n} y_i(\ln \hat{y}'_i - \ln \hat{y}_i) \tag{8}$$

$$= -\sum_{i=1}^{n} y_i \ln \left(\frac{e^{h_i+\Delta h_i}}{e^{h_i}} \cdot \frac{\sum_{j=1}^{n} e^{h_j}}{\sum_{j=1}^{n} e^{h_j+\Delta h_j}} \right) \tag{9}$$

Note that $\Delta h_i \to 0$, where $e^{\Delta h_j} \approx 1$. Then $\Delta L = -\sum_{i=1}^{n} y_i \ln e^{\Delta h_i} = -\sum_{i=1}^{n} y_i k_i |h_\delta|$.

A.3 Proof of Property 3

Property 3. *Let the minimum output particle of the sample before fine-tuning be h_δ. Suppose that after τ fissions at the end of fine-tuning, the minimum output particle becomes h_{δ_τ}. If the sequence of fission particle values $|h_{\delta_1}|, |h_{\delta_2}|, \ldots, |h_{\delta_\tau}|$ is uniformly distributed successively in the ranges $[0, |h_\delta|]$, $[0, |h_{\delta_1}|]$, …, $[0, |h_{\delta_{\tau-1}}|]$, then $E[|h_{\delta_i}|] = \frac{1}{2^i}|h_\delta|$, where $i \in [1, 2, \ldots, n]$.*

Proof. According to the assumption, the particle value $|h_{\delta_i}|$ is uniformly distributed in the interval $[0, |h_{\delta_{i-1}}|]$. By the law of total probability, $E[|h_{\delta_i}|] = \frac{1}{|h_{\delta_{i-1}}|} \int_0^{|h_{\delta_{i-1}}|} x dx = \frac{1}{2}|h_{\delta_{i-1}}|$. By continuous upward recursion, the proof is completed.

The actually observed binary fission phenomenon confirms the uniformity of the output particle fission values.

A.4 Proof of Property 4

Property 4. *Assume that the probability of occurrence of particle fission whithin a sample is p, where a batch $D_{batch} \in D$, $|D_{batch}| = b$, $N = |D|$. After τ fissions, the probability of randomly selecting a non-fissioned batch from the dataset is $p_{batch} = \frac{C_{N_\tau}^b}{C_N^b}$, where $N_\tau = N \cdot (1-p)^\tau$.*

Proof. According to the problem, after one fission, the number of non-fissioned samples in D is $N_1 = N \cdot (1-p)$. Then after τ fissions, $N_\tau = N \cdot (1-p)^\tau$. Then the probability that any batch does not contain fissioned samples is $p_{batch} = \frac{C_{N_\tau}^b}{C_N^b}$. Since the above steps are in line with the requirements, the proof is completed.

B Further Research on Loss Particles and Output Particles

B.1 Universality of Loss Particles in PLMs

Loss particles are commonly found in PLMs. Figure 7 shows the characteristics of loss particles in different PLMs without fine-tuning. The models used are BRTE, RoBERTA, GPT2, and T5 respectively, the dataset used is SST-2, and the batch size is 32, and the penalty coefficient $\alpha = 5 \times 10^{-8}$. Unless otherwise specified, in this paper, BERT refers to BRTE-base-uncased, RoBERTA refers to RoBERTA-base, T5 refers to T5-base, and all models are provided by Hugging face (Wolf, 2019). It can be seen from Figure 7 that for most of the batches in SST-2, each model has loss particles. However, for different models, the distribution of loss particle sizes with batches is different.

The loss particles of different tasks are shown in Figure 8. Among them, SST-2 (a) is a binary sentiment analysis task, using the cross-entropy loss function. STS-B (b) is a regression task for judging the similarity of semantic texts, using the mean square error loss function. It can be seen from Figure 8 that there are significant differences in the loss particles of different tasks.

In addition, we studied the loss particles of each sample in the BERT model. Figure 9 shows the loss particles of each sample in the MRPC dataset on the BERT model, where the penalty coefficient $\alpha = 5 \times 10^{-8}$. We found that most samples have loss particles, and there are also positive and negative differences between the loss particles of samples, and there is a multiple relationship in size. At the same time, in addition to the smaller BERT model, we also found the existence of loss particles in the larger LLaMa2-7b model, as shown in Figure 10.

B.2 Variations of Loss Particles during Fine-Tuning

We further studied the change rules of the BERT model during fine-tuning using MNLI and QQP. It was found that during the fine-tuning process, the loss particles gradually underwent discrete fission, as shown in Figure 11. Since the calculation amount of calculating the loss particles at each step is huge because the model needs to be scaled at each step. Therefore, we used a simple and effective method to replace it. The specific steps are as follows: 1) Obtain the minimum loss particle L_δ of the model before fine-tuning; 2) Record the loss of each step of training to form a set $L = \{L_{step0}, L_{step1}, \ldots, L_{stepN}\}$, where N represents the total number of training steps; 3) Calculate the remainder of dividing the loss of each step by L_δ to form a new set $R = \{R_{step0}, R_{step1}, \ldots, R_{stepi}, \ldots, R_{stepN}\}$, where $R_{stepi} = L_{stepi} \% L_\delta$. In Figure 11, we scaled R_{stepi} by R_{stepi}/L_δ. As the fine-tuning progresses, the number of values in R increases, indicating that the number of particles fissioned from L_δ is increasing. We represent the set of particles generated during the entire process as $l_\delta = \{l_{step0}, l_{step1}, \ldots, l_{stepi}, \ldots, l_{stepN}\}$. Where $l_{stepi} = \frac{T}{2^\tau} L_\delta, T \in [0, 1, 2, \ldots, 2^\tau]$.

B.3 Research on Output Particles of Different PLMs

Figure 12 shows the s_1 scores of the output particles of different PLMs. The calculation method of the s_1 score is consistent with the settings in $\text{PSP}_{\text{oneshot}}$. s_1 represents the initial sensitivity of the PLMs to the sample, which corresponds to the $Score$ in Figure 12. Taking the datasets QNLI and MNLI as examples, we compared the output particles of the BRTE, RoBERTA, GPT2, and T5 models. Specifically, we first calculated the s_1 score of each sample for the model, then sorted them, and then compared the distribution of different models. At the same time, since the s_1 scores of a small number of samples were too large, we performed a logarithmic scaling on $Score$: $\log(Score) + C$. It can be seen from Figure 12 that for QNLI and MNLI, there are significant differences between the output particles of different models, which also reflects the differences in the knowledge mastery

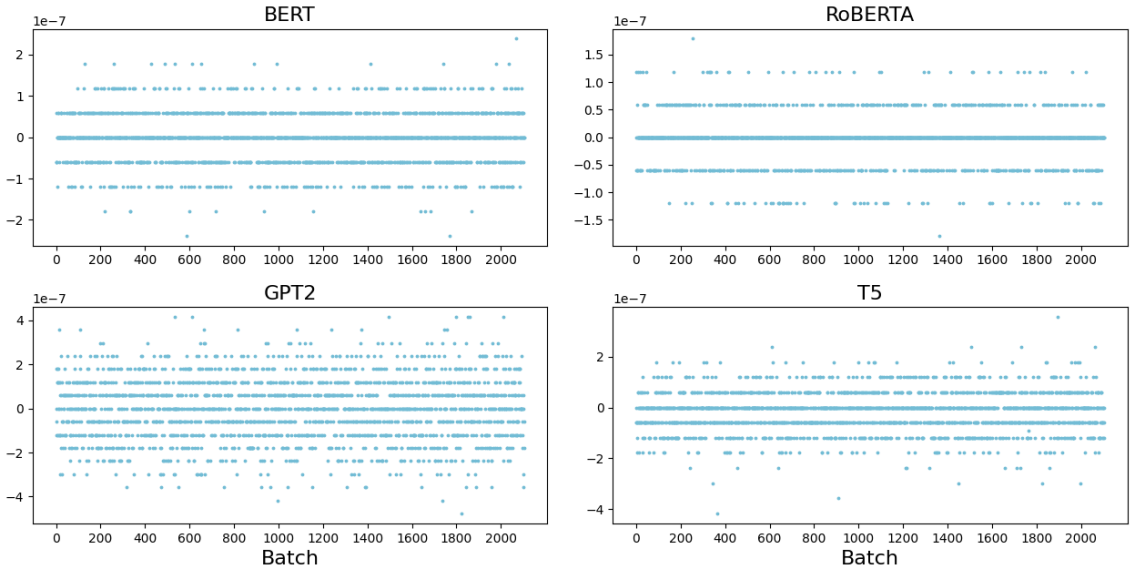

Figure 7: Loss particles of different PLMs. The models used include BERT, RoBERTA, GPT-2, and T5 from Hugging Face. The dataset used is SST-2.

of these two downstream tasks by different PLMs. And for the RoBERTA, GPT2, and T5 models, the overall performance of their output particles is relatively stable, while the changes of the BRTE model are more drastic.

B.4 Research on Output Particles after Slight Compression of PLMs

Figure 13 shows that when the penalty hyperparameter $\alpha < 3 \times 10^{-8}$, $\Delta h = \mathbf{0}$, where $\mathbf{0} \in R^d$. When $\alpha = 3 \times 10^{-8}$, the output differences of some dimensions begin to jump and remain unchanged in the interval $[3 \times 10^{-8}, 8 \times 10^{-8}]$. That is, in the interval $[3 \times 10^{-8}, 8 \times 10^{-8}]$, the output particle of the pre-fine-tuning model $h_{\delta/without} = 5.960464 \times 10^{-8}$, and the output particle of the post-fine-tuning model $h_{\delta/with} = 1.117587 \times 10^{-8}$. Among them, $h_{\delta/with} = \frac{3}{2^4} h_{\delta/without}$, indicating that some output particles have undergone fission after fine-tuning.

B.5 Research on Output Particles after Slight Expansion of PLMs

In addition, we used the BERT model and the MRPC dataset to study the output particles after a slight expansion of the model θ. It can be seen from Figure 14 that when the BERT model is slightly expanded and the scaling coefficient $\alpha = 6 \times 10^{-8}$, output particles appear. In Figure 13 (a), when the model is slightly compressed, the output particles appear when the scaling coefficient $\alpha = 3 \times 10^{-8}$. This shows that both slight compression and expansion of the PLMs will result in output particles. However, the model is more sensitive to slight compression.

C Experimental Details

C.1 Experimental Details

We divided the 8 tasks in GLUE into two categories during training. One category is tasks with larger datasets, including QNLI, MNLI, and QQP. The other category is tasks with smaller training sets, including CoLA, SST-2, MRPC, STS-B, and RTE. For these two categories, our training settings are shown in Table 4. At the same time, for all tasks, we used a unified pruning rate of 50%. We used the AdamW as our optimizer. For learning rate adjustment, we used a linear scheduler with a warm-up ratio of 0.1. For $\text{PSP}_{\text{one-shot}}$, the scaling coefficient $\alpha = 5 \times 10^{-8}$. And in $\text{PSP}_{\text{zero-shot}}$, $\alpha = 1 \times 10^{-7}$. All experiments were implemented on NVIDIA RTX3090 GPUs.

C.2 More Results of the $\text{PSP}_{\text{one-shot}}$ Method on BERTA-base

BERT-base. Table 5 reports the experimental results of using the BERT-base pre-trained model (Devlin, 2018). It is not difficult to see that $\text{PSP}_{\text{one-shot}}$ leads in most cases and even outperforms the results of training with the entire dataset. Especially for tasks such as MRPC, RTE, and CoLA, $\text{PSP}_{\text{one-shot}}$ exceeds the results of using the full dataset for training by more than 1%. At the

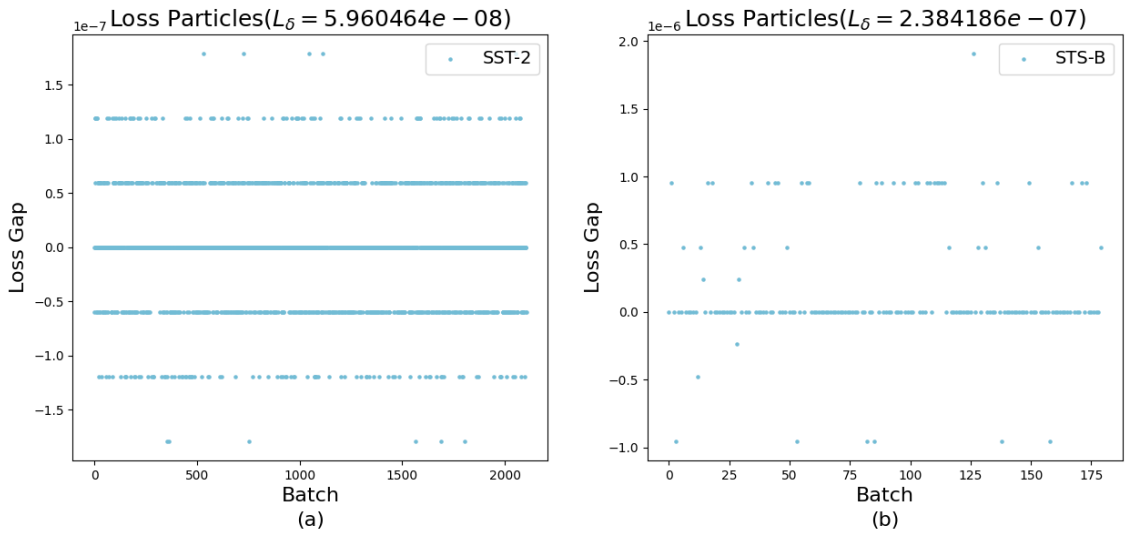

Figure 8: Loss particles across different tasks. The model used is BERT, with SST-2 (a) and STS-B (b) as the datasets.

Epoch(E_{pre})	lr	batch size	weight decay	Epoch(E_{post})	lr scheduler	optimizer
MRPC, RTE, CoLA, STS-B, SST-2						
10	2e-5	32	0.002	5	linear	Adam
QNLI, MNLI, QQP						
5	3e-5	64	0.002	1	linear	Adam

Table 4: The hyperparameters we used in the GLUE tasks.

same time, based on (Fayyaz et al., 2022), we extended the application of EL2N and GraNd methods to other GLUE tasks and found that they are not suitable for tasks such as MRPC, RTE, and CoLA. We found that for the BERT-base model, these three tasks are different from others, and it is more effective to select the 50% of data with lower scores for subsequent training. The performance of CoLA is as high as 60.33%, exceeding that of the full dataset by 2.72%. For the larger MNLI and QQP datasets, the result of $\text{PSP}_{\text{one-shot}}$ is at most 0.54% lower than that of the full dataset, but the amount of training data is reduced by 50%. Moreover, $\text{PSP}_{\text{one-shot}}$ outperforms the results of random pruning, which proves that the pre-trained model can improve its generalization ability by selecting a small amount of data based on its own sensitivity to the data, even without knowing the data labels.

D General Characteristics of Datasets

In Table 6, we provide an overall description of the GLUE datasets. In general, the GLUE datasets cover a large number of training set sizes and different types of language understanding tasks to measure the performance of the model in a wide range of tasks. The data we use does not contain any personally identifying info or offensive content.

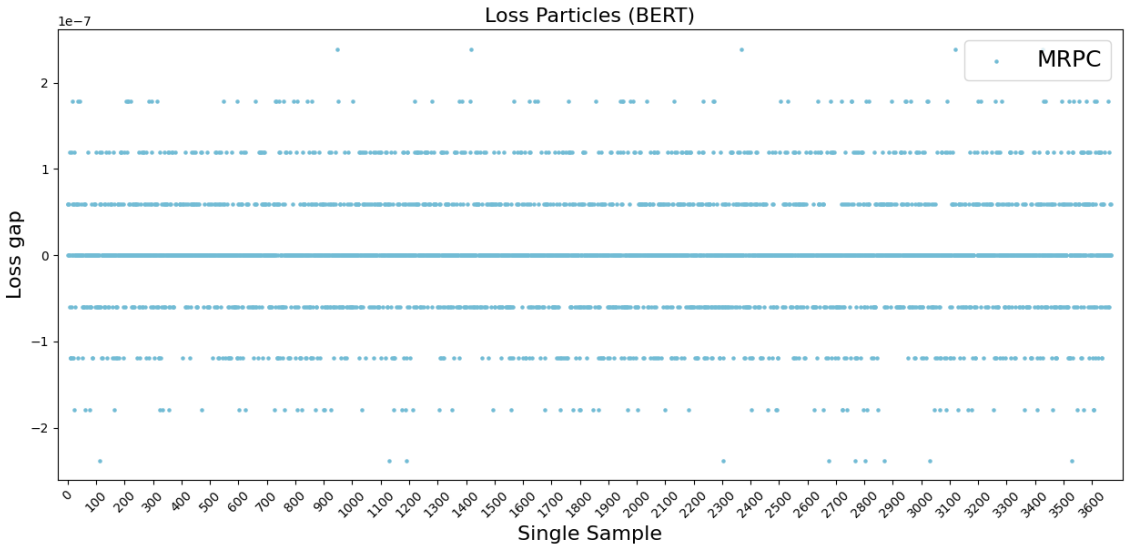

Figure 9: Loss particles for each sample in the MRPC dataset using the BERT model.

Dataset	MRPC	RTE	CoLA	STS-B	SST-2	QNLI	MNLI	QQP	AVG
FULL	86.24	67.08	57.61	88.76	92.78	91.45	84.37	91.10	82.42
Random	81.49	63.72	54.26	87.96	92.77	90.37	83.14	89.75	80.43
GraNd	65.69	52.71	53.23	87.46	92.83	90.03	82.24	89.89	76.76
EL2N	61.02	52.70	50.37	–	92.78	90.68	82.16	89.95	74.24
EL2N*	85.66	65.34	59.99	–	92.55	**91.41**	83.45	**91.02**	81.35
$\text{PSP}_{\text{one-shot}}$	**87.25**$^{\downarrow}$	**68.23**$^{\downarrow}$	**60.33**$^{\downarrow}$	**89.28**$^{\uparrow}$	**93.01**$^{\uparrow}$	91.26$^{\uparrow}$	**83.83**$^{\uparrow}$	90.86$^{\uparrow}$	**83.01**

Table 5: Comparison results of $\text{PSP}_{\text{one-shot}}$ and other methods on GLUE tasks using the BERT-base model. The retention ratio ρ of all data subsets is 50%. The superscript "↓" indicates selecting the 50% of data points with smaller scores, and the superscript "↑" indicates selecting the 50% of data points with larger scores. The GraNd and EL2N scores are the average scores of five independently trained proxy models. Since STS-B is a regression task, we did not calculate its EL2N and Single EL2N scores. To exclude the influence of random factors, all experimental results are the average of five independent runs.

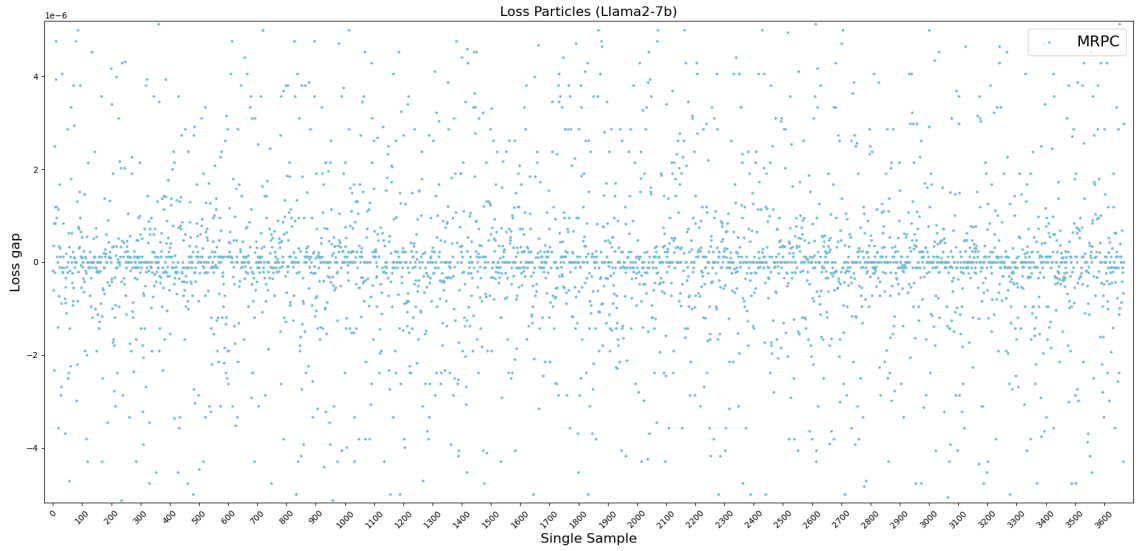

Figure 10: Loss particles for each sample in the MRPC dataset using the LLaMa2-7b model.

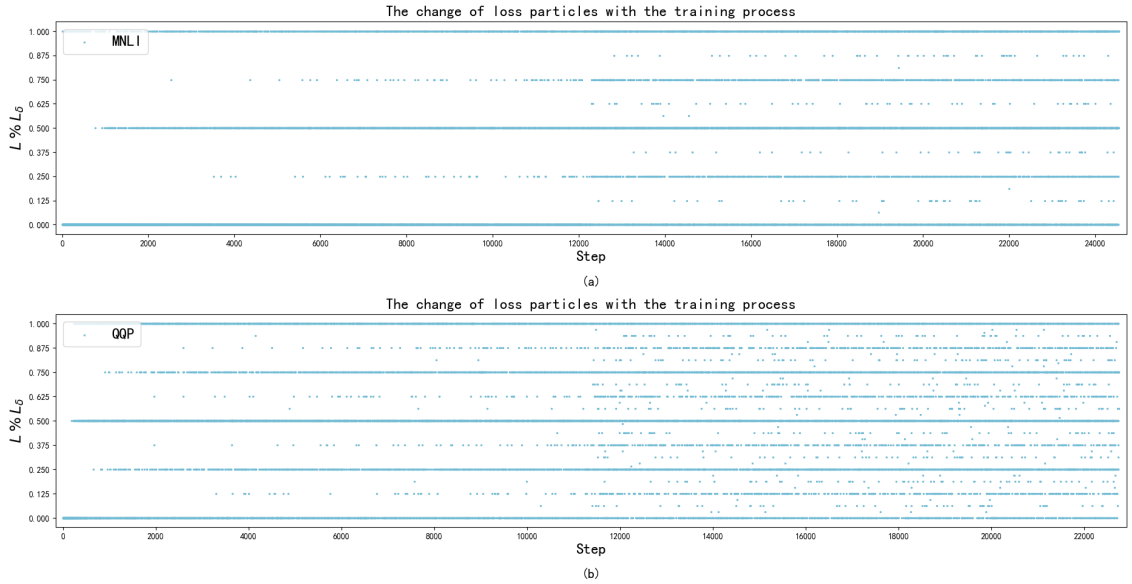

Figure 11: The change-diagram of loss particles of MNLI(a) and QQP(b) during the training process. Using the BERT model, the formula $L\%L_\delta$ is used to record the remainder of the loss L at each step divided by the initial loss particle L_δ. If the remainder is 0, it means that the loss particle remains unchanged; otherwise, the loss particle has undergone cleavage. Let l_{stepi} represent the loss particle at the i-th step, then $l_{stepi} = \frac{T}{2^\tau} L_\delta$, where $T \in [0, 1, 2, \ldots, 2^\tau]$.

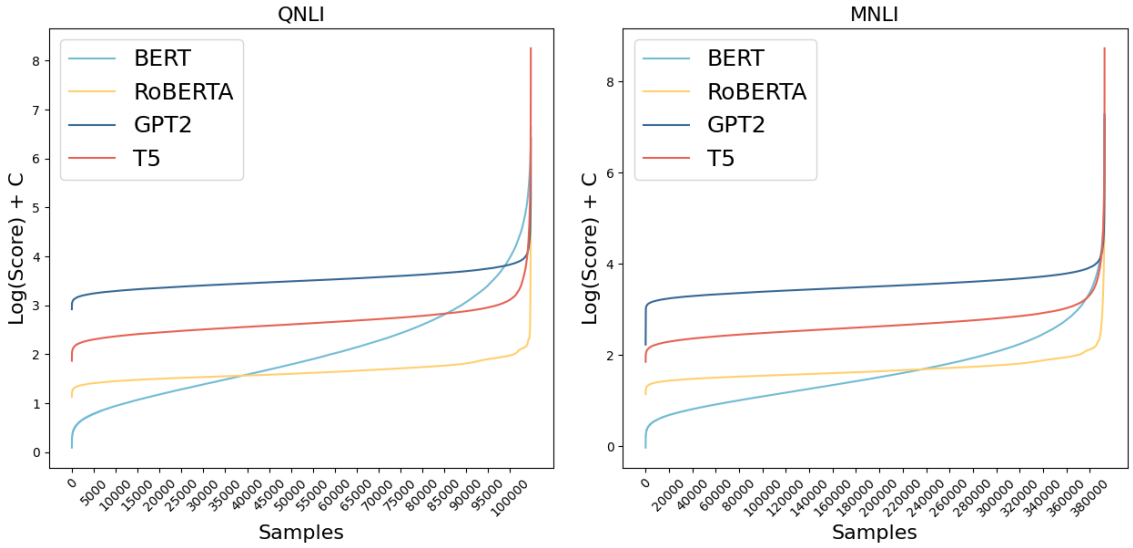

Figure 12: s_1 scores of output particles across different PLMs. The models used include BERT, RoBERTA, GPT-2, and T5. Comparisons are conducted across the QNLI, MNLI, MRPC and SST-2 tasks.

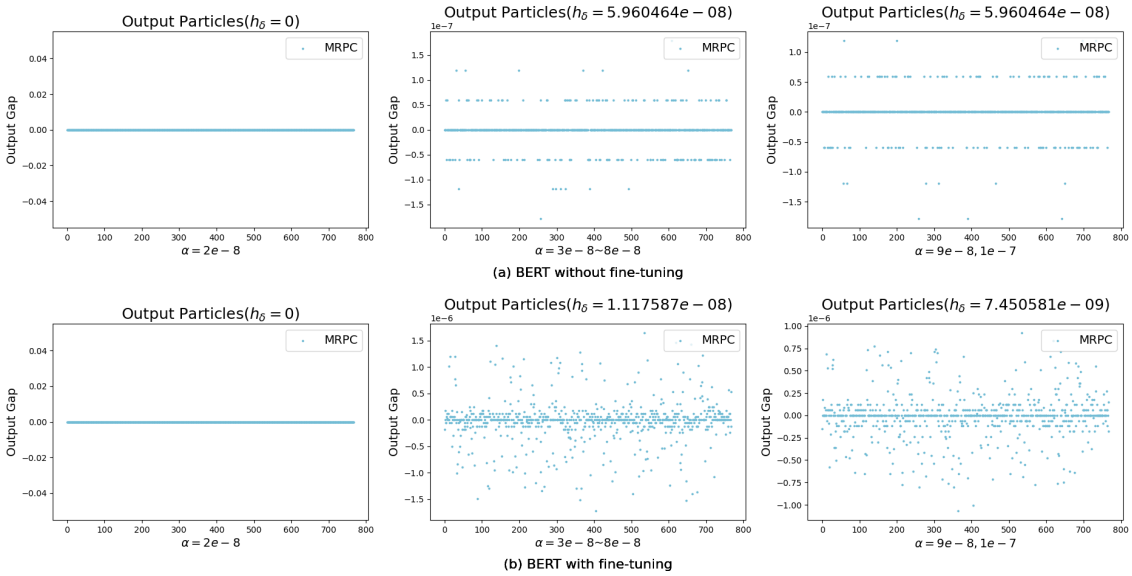

Figure 13: Output particles under different scaling factors α before and after fine-tuning. For the models before and after fine-tuning: 1) When $\alpha = 2e-8$, no output particles appear in either of them; 2) When $\alpha = 3e-8$, output particles appear in both of them, and the particles remain unchanged in the interval $[3e-8, 8e-8]$; 3) When $\alpha = 9e-8$ or $\alpha = 1e-7$, the output particles in each dimension change.

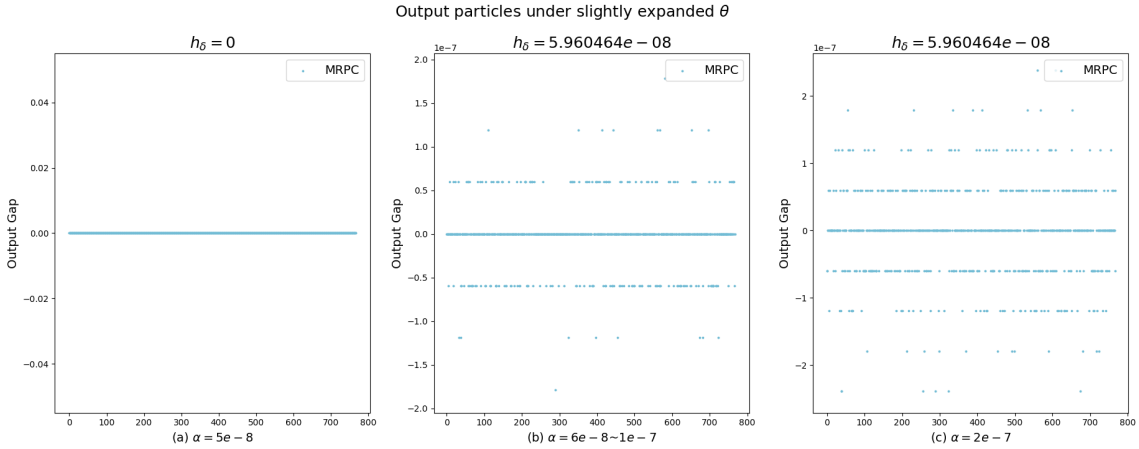

Figure 14: Output particles of the slight enlargement of the model θ using different scaling factors α. 1) When $\alpha = 5e-8$, no output particles appear; 2) When $\alpha = 6e-8$, output particles appear and remain unchanged in the interval $[6e-8, 1e-7]$; 3) When $\alpha = 2e-7$, the output particles in each dimension change. The model used is BERT and the dataset used is MRPC.

Name	|Train|	|Test|	Task	Eval Metric	Domain
			Single-Sentence Tasks		
CoLA	8.5k	1k	acceptability	Matthew's	misc.
SST-2	67k	1.8k	sentiment	Acc	movie reviews
			Similarity and Paraphrase Tasks		
MRPC	3.7k	1.7k	paraphrase	Acc	news
QQP	364k	391k	paraphrase	Acc	social QA questions
STS-B	7k	1.4k	sentence similarity	Pearson Cor.	misc.
			Inference Tasks		
MNLI	393k	20k	NLI	Acc	misc.
QNLI	105k	5.4k	QA/NLI	Matched Acc.	Wikipedia
RTE	2.5k	3k	NLI	Acc	news, Wikipedia

Table 6: Statistics of GLUE datasets.