mathematics

MDPI

*Article*

# Structure-Aware Low-Rank Adaptation for Parameter-Efficient Fine-Tuning

**Yahao Hu, Yifei Xie, Tianfeng Wang, Man Chen and Zhisong Pan ***

Command and Control Engineering College, Army Engineering University of PLA, Nanjing 210007, China; yahao_hu@163.com (Y.H.); xieyifei10@163.com (Y.X.); tianfengw97@163.com (T.W.); 19205060770@stu.csust.edu.cn (M.C.)
* Correspondence: panzhisong@aeu.edu.cn

**Abstract:** With the growing scale of pre-trained language models (PLMs), full parameter fine-tuning becomes prohibitively expensive and practically infeasible. Therefore, parameter-efficient adaptation techniques for PLMs have been proposed to learn through incremental updates of pre-trained weights, such as in low-rank adaptation (LoRA). However, LoRA relies on heuristics to select the modules and layers to which it is applied, and assigns them the same rank. As a consequence, any fine-tuning that ignores the structural information between modules and layers is suboptimal. In this work, we propose structure-aware low-rank adaptation (SaLoRA), which adaptively learns the intrinsic rank of each incremental matrix by removing rank-0 components during training. We conduct comprehensive experiments using pre-trained models of different scales in both task-oriented (GLUE) and task-agnostic (Yelp and GYAFC) settings. The experimental results show that SaLoRA effectively captures the structure-aware intrinsic rank. Moreover, our method consistently outperforms LoRA without significantly compromising training efficiency.

**Keywords:** pre-trained language models; parameter-efficient fine-tuning; low-rank adaptation; intrinsic rank; training efficiency

**MSC:** 68T50

check for updates

**Citation:** Hu, Y.; Xie, Y.; Wang, T.; Chen, M.; Pan, Z. Structure-Aware Low-Rank Adaptation for Parameter-Efficient Fine-Tuning. *Mathematics* **2023**, *11*, 4317. https://doi.org/10.3390/math11204317

Academic Editors: Florin Leon, Mircea Hulea and Marius Gavrilescu

Received: 3 September 2023
Revised: 8 October 2023
Accepted: 10 October 2023
Published: 17 October 2023

## 1. Introduction

With the scaling of model and corpus size [1–5], large language models (LLMs) have demonstrated an ability for in-context learning [1,6,7] in various natural language processing (NLP) tasks, that is, learning from a few examples within context. Although in-context learning is now the prevalent paradigm for using LLMs, fine-tuning still outperforms it in task-specific settings. In such scenarios, a task-specific model is exclusively trained on a dataset comprising input–output examples specific to the target task. However, full parameter fine-tuning, which updates and stores all the parameters for different tasks, becomes impractical when dealing with large-scale models.

In fact, LLMs with billions of parameters can be effectively fine-tuned by optimizing only a few parameters [8–10]. This has given rise to a branch of parameter-efficient fine-tuning (PEFT) techniques [11–16] for model tuning. These techniques optimize a small fraction of the model parameters while keeping the rest fixed, thereby significantly reducing computational and storage costs. For example, LoRA [15] introduces trainable low-rank decomposition matrices into LLMs, enabling the model to adapt to a new task while preserving the integrity of the original LLMs and retaining the acquired knowledge. Fundamentally, this approach is built upon the assumption that updates to the weights of the pre-trained language model have a lower rank during adaptation to specific downstream tasks [8,9]. Thus, by reducing the rank of the incremental matrices, LoRA optimizes less than 0.5% of the additional trainable parameters. Remarkably, this optimization achieves comparable or even superior performance to that of full parameter fine-tuning.

However, despite its advantages, LoRA also comes with certain limitations that warrant consideration. One limitation lies in LoRA's reliance on heuristics to select the modules and layers to which it is applied. Though heuristics can be effective under specific circumstances, their lack of generalizability is a concern. This lack of generalizability can result in suboptimal performance, or even complete failure, when applied to new data. Another limitation is the assignment of the same rank to incremental matrices across different modules and layers. This tends to oversimplify the complex structural relationships and important disparities that exist within neural networks. This phenomenon is illustrated in Figure 1.

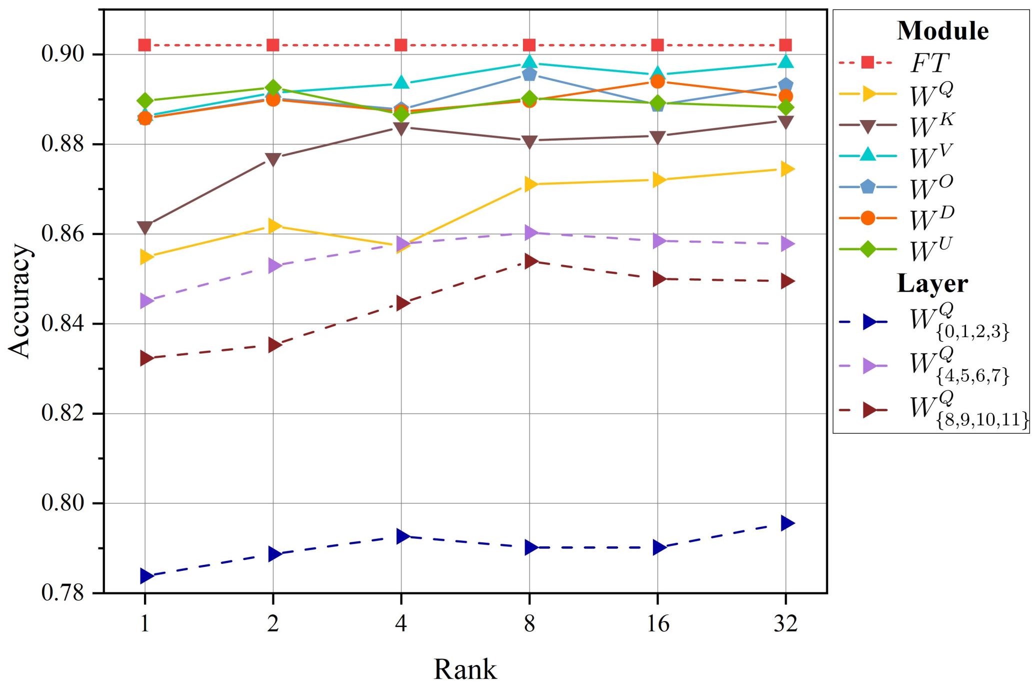

**Figure 1.** Fine-tuning performance of LoRA across different modules and layers with varying ranks on MRPC.

In this paper, we propose a novel approach called structure-aware low-rank adaptation (SaLoRA), which adaptively learns the intrinsic rank of each incremental matrix by removing rank-0 components. As shown in Figure 2, we introduce a diagonal gate matrix $\boldsymbol{G} = \mathrm{diag}(g_1, \ldots, g_r)$ for each incremental matrix. The modified incremental matrix can be represented as $\Delta \boldsymbol{W} = \boldsymbol{BGA}$. The incremental matrix is divided into triplets, where each triplet $\mathcal{T}_i$ contains the $i$-th column of $\boldsymbol{B}$, the $i$-th gate mask of $\boldsymbol{G}$ and the $i$-th row of $\boldsymbol{A}$. Here, $g_i$ represents the binary "gate" that indicates the presence or absence of the $i$-th triplet. Although incorporating the active triplet count as a penalty term in the learning objective is unfeasible, we employ a differentiable relaxation method to selectively remove non-critical triplets by considering the $L_0$ norm [17,18]. The $L_0$ norm is equal to the number of non-zero triplets and encourages the model to deactivate less essential triplets. This strategy assigns a higher rank to crucial incremental matrices to capture task-specific information. Conversely, less significant matrices are pruned to possess a lower ranks preventing overfitting. However, $\boldsymbol{A}$ and $\boldsymbol{B}$ are not orthogonal, implying potential dependence among the triplets. Removing these triplets can result in a more significant deviation from the original matrix. To enhance training stability and generalization, we introduce orthogonality regularization for $\boldsymbol{B}$ and $\boldsymbol{A}$. Furthermore, we integrate a density constraint and leverage Lagrangian relaxation [19] to control the number of valid parameters.

We conduct extensive experiments on a wide range of tasks and models to evaluate the effectiveness of SaLoRA. Specifically, we conduct experiments on the General Language Understanding Evaluation [20] benchmark in a task-oriented setting to assess the model's performance. In addition, we evaluate the model's performance in a task-agnostic setting by fine-tuning LLaMA-7B with a 50K cleaned instruction-following dataset [21], and then perform zero-shot task inference on two text style transfer tasks: sentiment transfer [22] and formality transfer [23]. The experimental results demonstrate that SaLoRA consistently outperforms LoRA without significantly compromising training efficiency.

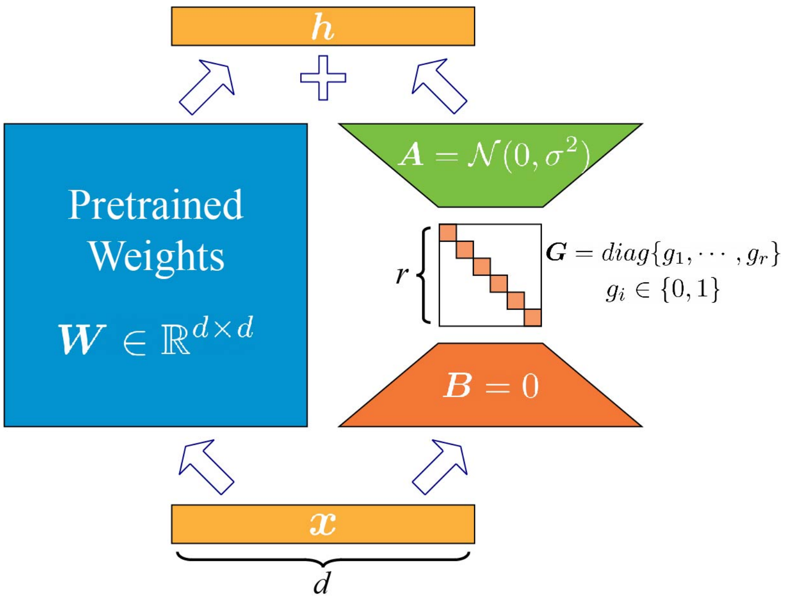
$h$
Pretrained Weights
$W \in \mathbb{R}^{d\times d}$
$A = \mathcal{N}(0, \sigma^2)$
$r$
$G = diag\{g_1, \cdots, g_r\}$
$g_i \in \{0, 1\}$
$B = 0$
$x$
$d$

**Figure 2.** Structure-aware low-rank adaptation.

## 2. Backgound

**Transformer Architecture.** The Transformer [24] is primarily constructed using two key submodules: a multi-head self-attention (MHA) layer and a fully connected feed-forward (FFN) layer. The MHA is defined as follows:

$$\begin{aligned} MHA(Q, K, V) &= Concat(head_1, \ldots, head_h)W^O, \\ head_i &= Atention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{1}$$

where $Q, K, V \in \mathbb{R}^{n\times d}$ are input-embedding matrices; $W^O \in \mathbb{R}^{d\times d}$ is an output projection; $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d\times d_k}$ are query, key and value projections of head $i$, respectively; $n$ is sequence length; $d$ is the embedding dimension; $h$ is the number of heads and $d_k = d/h$ is the hidden dimension of the projection subspaces. The FFN consists of two linear transformations separated by a ReLU activation:

$$FFN(x) = ReLU(xW^U + b^U)W^D + b^D \tag{2}$$

where $W^U \in \mathbb{R}^{d\times d_m}$ and $W^D \in \mathbb{R}^{d_m\times d}$.

**Parameter-Efficient Fine-Tuning.** With the growing size of models, recent works have developed three main categories of parameter-efficient fine-tuning (PEFT) techniques. These techniques optimize a small fraction of model parameters while keeping the rest fixed, thereby significantly reducing computational and storage costs [10]. For example, addition-based methods [11–13,25,26] introduce additional trainable modules or parameters that are not part of the original model or process. Specifcation-based methods [14,27,28] specify certain parameters within the original model or process as trainable, whereas the others remain frozen. Reparameterization-based methods [15,16,29], including LoRA, reparameterize existing parameters into a parameter-efficient form by transformation. In this study, we focus on reparameterization-based methods, with particular emphasis on LoRA.

**Low-Rank Adaptation.** LoRA, as introduced in the work of Hu et al. [15], represents a typical example of a reparameterization-based method. In LoRA, some pre-trained weights of LLMs' dense layers are reparameterized by injecting trainable low-rank incremental matrices. This reparameterization only allows low-rank matrices to be updated, while keeping the original pre-trained weights frozen. By reducing the rank of these matrices, LoRA effectively reduces the number of parameters during the fine-tuning process of LLMs. Consider a pre-trained weight matrix $\boldsymbol{W} \in \mathbb{R}^{d \times k}$, accompanied by a low-rank incremental matrix $\Delta\ \boldsymbol{W} = \boldsymbol{BA}$. For $\boldsymbol{h} = \boldsymbol{Wx}$, the modified forward pass is as follows:

$$\boldsymbol{h} = \boldsymbol{Wx} + \frac{\alpha}{r}\Delta\boldsymbol{W}\boldsymbol{x} = \boldsymbol{Wx} + \frac{\alpha}{r}\boldsymbol{BAx} \tag{3}$$

where $\boldsymbol{B} \in \mathbb{R}^{d \times r}$, $\boldsymbol{A} \in \mathbb{R}^{r \times k}$, with the rank $r \ll min(d,k)$, and $\alpha$ is a constant scale hyperparameter. The matrix $\boldsymbol{A}$ adopts a random zero-mean Gaussian initialization, while the matrix $\boldsymbol{B}$ is initialized as a zero matrix. Consequently, the product $\Delta\boldsymbol{W} = \boldsymbol{BA}$ is initially set to zero at the beginning of training. Let $\boldsymbol{B}_{*j}$ and $\boldsymbol{A}_{j*}$ denote the $j$-th column of $\boldsymbol{B}$ and the $j$-th row of $\boldsymbol{A}$, respectively. Using this notation, $\Delta\boldsymbol{W}$ can be expressed as $\Delta\boldsymbol{W} = \sum_{j=1}^{r} \boldsymbol{B}_{*j}\boldsymbol{A}_{j*}$.

## 3. Method

In this section, we will first give a brief introduction to parameter-efficient fine-tuning, and then discuss our proposed model based on the problem definition.

### *3.1. Problem Formalization*

We consider the general problem of efficiently fine-tuning LLMs for specific downstream tasks. Firstly, let us introduce some notations. Consider a training corpus $\mathcal{D} = (x_i, y_i)_{i=1}^{N}$, where $N$ represents the number of samples. Each sample consists an input, $x_i$, and its corresponding output, $y_i$. We use the index $i$ to refer to the incremental matrix, i.e., $\Delta\boldsymbol{W}_i = \boldsymbol{B}_i\boldsymbol{A}_i$ for $i = 1, \ldots, K$, where $K$ is the number of incremental matrices. However, LoRA's assumption of identical ranks for each incremental matrix overlooks structural relationships and the varying importance of weight matrices across different modules and layers during fine-tuning. This oversight can potentially impact overall model performance. Our objective is to determine the optimal $\{rank^*(\Delta\boldsymbol{W}_i)\}_{i=1}^{K}$ on the fly. The optimization objective can be formulated as follows:

$$\begin{aligned} &\min_{\boldsymbol{\mathcal{W}}} \mathcal{R}(\boldsymbol{\mathcal{W}}) \triangleq \frac{1}{N}\left(\sum_{i=1}^{N} \mathcal{L}(f(x_i;\boldsymbol{\mathcal{W}}), y_i)\right) \\ &s.t. \quad rank(\Delta\boldsymbol{W}_i) \leq r, k = 1, \ldots, K. \end{aligned} \tag{4}$$

where $\boldsymbol{\mathcal{W}} = \{\Delta\boldsymbol{W}_i, \ldots, \Delta\boldsymbol{W}_K\}$ represents the sets of trainable parameters and $\mathcal{L}$ corresponds to a loss function, such as cross-entropy for classification. Note that $rank(\Delta\boldsymbol{W}_i) \in \{0, 1, \ldots, r\}$ is an unknown parameter that needs to be optimized.

### *3.2. Structure-Aware Intrinsic Rank Using $L_0$ Norm*

To find the optimal $\{rank^*(\Delta\boldsymbol{W}_i)\}_{i=1}^{K}$ on the fly, with minimal computational overhead during training, we introduce a gate matrix $\boldsymbol{G}$ to define the structure-aware intrinsic rank:

$$\Delta\boldsymbol{W} = \boldsymbol{BGA} = \sum_{j=1}^{r} g_j \boldsymbol{B}_{*j}\boldsymbol{A}_{j*} \tag{5}$$

where the $g_j \in \{0, 1\}$ serves as a binary "gate", indicating the presence or absence of the $j$-th rank. The gate matrix $\boldsymbol{G}$ = diag$(g_1, \ldots, g_r)$ is a diagonal matrix consisting of the pruning variables. By learning the variable $g_j$, we can control the rank of each incremental matrix individually, rather than applying the same rank to all matrices. To deactivate non-critical rank-0 components, the ideal approach would be to apply $L_0$ norm regularization to the gate matrix $\boldsymbol{G}$:

$$||\boldsymbol{G}||_0 = \sum_{j=1}^{r} g_j \tag{6}$$

where $r$ is the rank of incremental matrices. The $L_0$ norm measures the number of non-zero triplets; thus, optimizing $L_0$ would encourage the model to deactivate less important incremental matrices.

Unfortunately, the optimization objective involving $||\boldsymbol{G}||_0$ is computationally intractable due to its non-differentiability, making it impossible to directly incorporate it as a regularization term in the objective function. Instead, we use a stochastic relaxation approach, where the gate variables $g$ are treated as continuous variables distributed within the interval $[0,1]$. We leverage the reparameterization trick [30,31] to ensure that $g$ remains differentiable. Following prior studies [17,19], we adopt the Hard-Concrete (HC) distribution as a continuous surrogate for random variables $g$, illustrated in Figure 3. The HC distribution applies a hard-sigmoid rectification to $s$, which can easily be sampled by first sampling $u \in U(0,1)$ and then computing as follows:

$$\begin{aligned} s &= Sigmod\left(\frac{\log\frac{u}{1-u} + \log\theta}{\tau}\right) \times (\zeta - \gamma) + \gamma \\ g &= \min(1, \max(0, s)) \end{aligned} \tag{7}$$

where $\theta$ is the trainable parameter of the distribution and $\tau$ is the temperature. The interval $(\gamma, \zeta)$, with $\gamma < 0$ and $\zeta > 1$, enables the distribution to concentrate probability mass at the edge of the support. The final outputs $g$ are rectified into $[0,1]$. By summing up the probabilities of the gates being non-zero, the $L_0$ norm regularization can be computed via a closed form, as follows:

$$\begin{aligned} \mathbb{E}[||\boldsymbol{G}||_0] &= \sum_{j=1}^{r} \mathbb{E}[g_j > 0] \\ &= \sum_{j=1}^{r} Sigmod\left(\log\theta_j - \tau\log\frac{-\gamma}{\zeta}\right) \end{aligned} \tag{8}$$

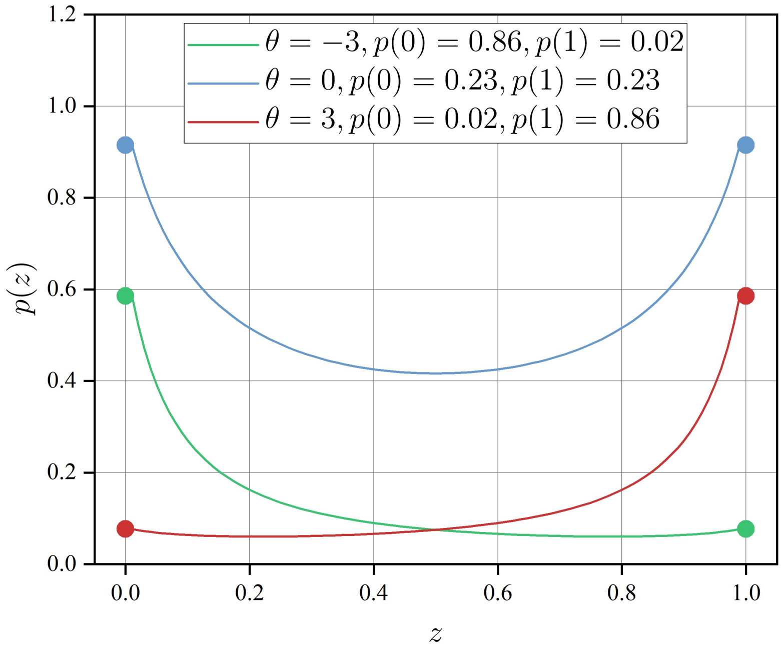

**Figure 3.** Hard-Concrete distribution with different parameters.

As $g$ now represents the output of the parameterized HC distribution function and serves as an intermediate representation for the neural network, gradient-based optimization methods can perform gradient updates for $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_r\}$. For each training batch, we sample the gate mask and then share it across the training examples within the batch to enhance sampling efficiency.

### 3.3. Enhanced Stability Using Orthogonal Regularization

In deep networks, orthogonality plays a crucial role in preserving the norm of the original matrix during multiplication, preventing signal vanishing or exploding [32]. However, in LoRA, where $\boldsymbol{B}$ and $\boldsymbol{A}$ are not orthogonal, the dependence can lead to larger variations when removing certain columns or rows through $L_0$ regularization. This, in turn, leads to training instability and the potential for negative effects on generalization [16]. For this, we turn to orthogonal regularization, which enforces the orthogonality condition:

$$\mathcal{R}_{orth}(\boldsymbol{B}, \boldsymbol{A}) = ||\boldsymbol{B}^T\boldsymbol{B} - \boldsymbol{I}||_F^2 + ||\boldsymbol{A}\boldsymbol{A}^T - \boldsymbol{I}||_F^2 \tag{9}$$

where $\boldsymbol{I}$ is the identity matrix.

Now, let us substitute Equations (8) and (9) into Equation (4) to derive the new training objective:

$$\min_{\boldsymbol{\mathcal{W}},\boldsymbol{\Theta}} \mathcal{R}(\boldsymbol{\mathcal{W}}, \boldsymbol{\Theta}) \triangleq \frac{1}{N}(\sum_{i=1}^{N} \mathcal{L}(f(x_i; \boldsymbol{\mathcal{W}}), y_i)) + \lambda \sum_{i=1}^{K} \mathbb{E}[||\boldsymbol{G}_i||_0] + \beta \sum_{i=1}^{K} \mathcal{R}_{orth}(\boldsymbol{B}_i, \boldsymbol{A}_i) \tag{10}$$

where $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i, \ldots, \boldsymbol{\theta}_K\}$ represents the sets of trainable parameters, and $\lambda$ and $\beta$ are two constant hyperparameters.

### 3.4. Controlled Budget Using Lagrangian Relaxation

If we only rely on Equation (10) to learn the intrinsic rank for each incremental matrix, the resulting parameter budget cannot be directly controlled. This limitation becomes problematic in many real-world applications that require a specific model size or parameter budget. To address this issue, we further introduce an additional density constraint on $\mathcal{R}(\boldsymbol{\mathcal{W}}, \boldsymbol{\Theta})$ to guide the network towards achieving a specific desired budget.

$$\begin{aligned} &\min_{\boldsymbol{\mathcal{W}}} \mathcal{R}(\boldsymbol{\mathcal{W}}) \triangleq \frac{1}{N}(\sum_{i=1}^{N} \mathcal{L}(f(x_i; \boldsymbol{\mathcal{W}}), y_i)) + \beta \sum_{i=1}^{K} \mathcal{R}_{orth}(\boldsymbol{B}_i, \boldsymbol{A}_i) \\ &s.t. \quad C(\boldsymbol{\Theta}) \triangleq \sum_{i=1}^{K} \frac{\mathbb{E}[||\boldsymbol{G}_i||_0] \times (d_i + k_i)}{\#(\boldsymbol{B}_i) + \#(\boldsymbol{A}_i)} = b \end{aligned} \tag{11}$$

where $b$ represents the target density and $\#(x)$ counts the total number of parameters in matrix $x$. $\Delta \boldsymbol{W}_i = \boldsymbol{B}_i \boldsymbol{A}_i$, where $\boldsymbol{B}_i$ is of $d_i \times r_i$, and $\boldsymbol{A}_i$ is of $r_i \times k_i$. However, lowering the density constraint poses a challenging and (not necessarily strictly) constrained optimization problem. To tackle this challenge, we leverage Lagrangian relaxation as an alternative approach, along with the corresponding min-max game:

$$\max_{\lambda} \min_{\boldsymbol{\mathcal{W}},\boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\mathcal{W}}, \boldsymbol{\Theta}, \lambda) \triangleq \mathcal{R}(\boldsymbol{\mathcal{W}}, \boldsymbol{\Theta}) + \lambda(C(\boldsymbol{\Theta}) - b)^2 \tag{12}$$

where $\lambda \in \mathbb{R}$ is the Lagrangian multiplier, which is jointly updated during training. The updates to $\lambda$ would increase the training loss unless the equality constraint is satisfied, resulting in the desired parameter budget. We optimize the Lagrangian relaxation by simultaneously performing gradient descent on $(\boldsymbol{\mathcal{W}}, \boldsymbol{\Theta})$ and projected gradient ascent (to $\mathbb{R}^+$) on $\lambda$, as demonstrated in previous works [19,33]. During the experiments, we observed that the term $\lambda(C(\boldsymbol{\Theta}) - b)^2$ converged quickly. To enhance training efficiency, we only optimize $(\boldsymbol{\Theta}, \lambda)$ between $T_{start}$ and $T_{end}$ time steps. We provide a summarized algorithm in Algorithm 1.

**Algorithm 1** SaLoRA

```
Input: Dataset $\mathcal{D}$; total iterations $T$; target density $b$; hyperparameters $\tau, \gamma, \zeta, \beta, \eta_p, \eta_c$.
Output: The fine-tuned parameters $\{\boldsymbol{\mathcal{W}}, \boldsymbol{\Theta}\}$.
  for $t = 1, \ldots, T$ do
    Sample a mini-batch from $\mathcal{D}$
    if $T_{start} \leq t < T_{end}$ then
      Sample a gate mask set $\boldsymbol{\mathcal{G}}$ from HC distribution and share it across the mini-batch
      Compute the gradient $\mathcal{L}(\boldsymbol{\mathcal{W}}, \boldsymbol{\Theta}, \lambda)$
      Update $\boldsymbol{\mathcal{W}}^{(t+1)} = \boldsymbol{\mathcal{W}}^{(t)} - \eta_p \nabla_{\boldsymbol{\mathcal{W}}} \mathcal{L}(\boldsymbol{\mathcal{W}}^{(t)}, \boldsymbol{\Theta}^{(t)}, \lambda^{(t)})$
      Update $\boldsymbol{\Theta}^{(t+1)} = \boldsymbol{\Theta}^{(t)} - \eta_c \nabla_{\boldsymbol{\Theta}} \mathcal{L}(\boldsymbol{\mathcal{W}}^{(t)}, \boldsymbol{\Theta}^{(t)}, \lambda^{(t)})$
      Update $\lambda^{(t+1)} = \lambda^{(t)} + \eta_c \nabla_{\lambda^{(t)}} \mathcal{L}(\boldsymbol{\mathcal{W}}^{(t)}, \boldsymbol{\Theta}^{(t)}, \lambda^{(t)})$
    else
      Compute the gradient $\mathcal{L}(\boldsymbol{\mathcal{W}})$
      Update $\boldsymbol{\mathcal{W}}^{(t+1)} = \boldsymbol{\mathcal{W}}^{(t)} - \eta_p \nabla_{\boldsymbol{\mathcal{W}}} \mathcal{L}(\boldsymbol{\mathcal{W}}^{(t)})$
    end if
  end for
  return The fine-tuned parameters $\{\boldsymbol{\mathcal{W}}^{(T)}, \boldsymbol{\Theta}^{(T)}, \lambda^{(T)}\}$.
```

### *3.5. Inference*

During training, the gate mask $g_i$ is a random variable drawn from the HC distribution. At inference time, we first calculate the expected value of each $g_i$ in $\boldsymbol{G}$. If the value of $g_i$ is greater than 0, we retain the corresponding $i$-th low-rank triplet. This procedure enables us to obtain the deterministic matrices $\boldsymbol{B}$ and $\boldsymbol{A}$.

## 4. Experiments

We evaluated the effectiveness of the proposed SaLoRA on RoBERTa [34] and LLaMA-7B in both task-oriented and task-agnostic settings.

**Baselines.** We compared SaLoRA with the following methods:

- **Fine-tuning (FT)** is the most common approach for adaptation. To establish an upper bound for the performance of our proposed method, we fine-tuned all parameters within the model.
- **Adapting tuning**, as proposed by Houlsby et al. [25], incorporates adapter layers between the self-attention module (and the MLP module) and the subsequent residual connection. Each adapter module consists of two fully connected layers with biases and a nonlinearity in between. This original design is referred to as **Adapter**$^H$. Recently, Pfeiffer et al. [11] introduced a more efficient approach, applying the adapter layer only after the MLP module and following a LayerNorm. We call it **Adapter**$^P$.
- **Prefix-tuning (Prefix)** [12] prepends a sequence of continuous task-specific activations to the input. During tuning, prefix-tuning freezes the model parameters and only backpropagates the gradient to the prefix activations.
- **Prompt-tuning (Prompt)** [13] is a simplified version of prefix-tuning, allowing the additional $k$ tunable tokens per downstream task to be prepended to the input text.
- **LoRA**, introduced by Hu et al. [15], is a state-of-the-art method for parameter-efficient fine-tuning. The original implementation of LoRA applied the method solely to query and value projections. However, empirical studies [16,35] have shown that extending LoRA to all matrices, including $W^Q$, $W^K$, $W^V$, $W^O$, $W^U$ and $W^D$, can further improve its performance. Therefore, we compare our approach with this generalized LoRA configuration to maximize its effectiveness.
- **AdaLoRA**, proposed by Zhang et al. [16], utilizes singular value decomposition (SVD) to adaptively allocate the parameter budget among weight matrices based on their respective importance scores. However, this baseline involves computationally intensive operations, especially for large matrices. The training cost can be significant, making it less efficient for resource-constrained scenarios.

### *4.1. Task-Oriented Performance*

**Models and Datasets.** We evaluated the performance of different adaptive methods on the GLUE benchmark [20] using pre-trained RoBERTa-base (125M) and RoBERTa-large (355 M) [34] models from the HuggingFace Transformers library [36]. See Appendix A for additional details on the datasets we used.

**Implementation Details.** For running all the baselines, we utilized a publicly available implementation [37]. We evaluated the performance of LoRA, AdaLoRA and SaLoRA at $r = 8$. To maintain a controlled parameter budget, we set the desired budget ratio (b) to 0.50 for both SaLoRA and AdaLoRA. During training, we used the AdamW optimizer [38], along with the linear learning rate scheduler. During our experiments, we observed that using a larger learning rate ($\eta_c$) significantly improved the learning process for both the gate matrices and Lagrange multiplier. Therefore, we set $\eta_c$ to 0.01 for all conducted experiments. We fine-tuned all models using an NVIDIA A100 (40 GB) GPU. Additional details can be found in Appendix B.

**Main Results.** We compared SaLoRA with the baseline methods under different model scale settings, and the experimental results on the GLUE development set are presented in Table 1. We can see that SaLoRA consistently achieved better or comparable performance compared with existing approaches for all datasets. Moreover, it even outperformed the FT method. SaLoRA's superiority was particularly striking when compared with LoRA, despite both models having a similar parameter count of 1.33 M/3.54 M for base/large model scales. After training, SaLoRA effectively utilized only $0.5 \times 1.33$ M/$0.5 \times 3.54$ parameters, yet still attained superior performance. This observation emphasizes the effectiveness of our method in learning the intrinsic rank for incremental matrices.

**Table 1.** Results with RoBERTa-base and RoBERTa-large on GLUE development set. We report the Pearson correlation for STS-B, Matthew's correlation for CoLA, and accuracy for other tasks. We report the mean and maximum deviation of 5 runs using different random seeds. The best results are shown in bold. † indicates numbers published in prior works.

| Model and Method | # Trainable Parameters | MNLI ACC | SST-2 ACC | CoLA Mathew | QQP ACC | QNLI ACC | RTE ACC | MRPC ACC | STS-B Pearson | ALL Avg |
|---|---|---|---|---|---|---|---|---|---|---|
| $\text{RoB}_{\text{base}}$(FT) † | 125.00 M | 87.6 | 94.8 | **63.6** | **91.9** | 92.8 | 78.7 | **90.2** | **91.2** | 86.4 |
| $\text{RoB}_{\text{base}}$(Prefix) | 1.33 M | $82.58_{\pm 0.24}$ | $91.65_{\pm 0.44}$ | $47.45_{\pm 4.47}$ | $85.98_{\pm 0.11}$ | $85.91_{\pm 0.44}$ | $60.47_{\pm 0.54}$ | $88.09_{\pm 1.37}$ | $87.49_{\pm 0.63}$ | 78.70 |
| $\text{RoB}_{\text{base}}$(Prompt) | 0.62 M | $79.14_{\pm 0.99}$ | $88.33_{\pm 1.44}$ | $42.35_{\pm 5.12}$ | $73.30_{\pm 5.84}$ | $80.51_{\pm 1.64}$ | $55.81_{\pm 2.67}$ | $69.75_{\pm 0.59}$ | $76.94_{\pm 3.59}$ | 70.77 |
| $\text{RoB}_{\text{base}}$(LoRA) | 1.33 M | $87.49_{\pm 0.44}$ | $94.77_{\pm 0.53}$ | $61.22_{\pm 2.09}$ | $91.39_{\pm 0.18}$ | $92.85_{\pm 0.28}$ | $79.24_{\pm .21}$ | $89.46_{\pm 1.23}$ | $90.89_{\pm 0.18}$ | 85.91 |
| $\text{RoB}_{\text{base}}$(AdaLoRA) | 1.33 M | $\mathbf{87.93}_{\pm \mathbf{0.20}}$ | $94.59_{\pm 0.21}$ | $59.29_{\pm 1.06}$ | $90.94_{\pm 0.13}$ | $92.61_{\pm 0.10}$ | $76.39_{\pm 1.30}$ | $87.35_{\pm 0.39}$ | $90.87_{\pm 0.15}$ | 85.00 |
| $\text{RoB}_{\text{base}}$(SaLoRA) | 1.33 M | $87.83_{\pm 0.04}$ | $\mathbf{95.14}_{\pm \mathbf{0.73}}$ | $63.39_{\pm 1.79}$ | $91.46_{\pm 0.09}$ | $\mathbf{92.99}_{\pm \mathbf{0.21}}$ | $\mathbf{81.01}_{\pm \mathbf{0.87}}$ | $\mathbf{90.20}_{\pm \mathbf{0.74}}$ | $91.13_{\pm 0.17}$ | **86.64** |
| $\text{RoB}_{\text{large}}$(FT) † | 356.05 M | 90.2 | 96.4 | 68.0 | **92.2** | 94.7 | 86.6 | 90.9 | 92.4 | 88.9 |
| $\text{RoB}_{\text{large}}$($\text{Adapt}^{\text{P}}$) † | 4.05 M | $90.2_{\pm 0.3}$ | $96.1_{\pm 0.3}$ | $68.3_{\pm 1.0}$ | $91.9_{\pm 0.1}$ | $94.8_{\pm 0.2}$ | $83.8_{\pm 2.9}$ | $90.2_{\pm 0.7}$ | $92.1_{\pm 0.7}$ | 88.4 |
| $\text{RoB}_{\text{large}}$($\text{Adapt}^{\text{H}}$) † | 7.05 M | $89.5_{\pm 0.5}$ | $96.2_{\pm 0.3}$ | $66.5_{\pm 4.4}$ | $92.1_{\pm 0.1}$ | $94.7_{\pm .2}$ | $83.4_{\pm 1.1}$ | $88.7_{\pm 2.9}$ | $91.0_{\pm 1.7}$ | 87.8 |
| $\text{RoB}_{\text{large}}$(Prefix) | 3.02 M | $88.61_{\pm 0.12}$ | $94.70_{\pm 0.44}$ | $60.06_{\pm 1.44}$ | $87.57_{\pm 0.25}$ | $89.60_{\pm 0.37}$ | $77.33_{\pm 1.37}$ | $89.85_{\pm .88}$ | $89.97_{\pm 3.37}$ | 84.71 |
| $\text{RoB}_{\text{large}}$(Prompt) | 1.09 M | $85.65_{\pm 2.54}$ | $93.95_{\pm 0.83}$ | $58.34_{\pm 2.63}$ | $83.98_{\pm 1.45}$ | $84.92_{\pm 3.28}$ | $58.70_{\pm 4.83}$ | $74.22_{\pm 2.65}$ | $80.47_{\pm 0.71}$ | 77.53 |
| $\text{RoB}_{\text{large}}$(LoRA) | 3.41 M | $89.96_{\pm 0.12}$ | $96.10_{\pm 0.11}$ | $\mathbf{68.76}_{\pm \mathbf{1.75}}$ | $88.67_{\pm 0.88}$ | $94.86_{\pm 0.07}$ | $85.49_{\pm 1.51}$ | $90.93_{\pm 0.74}$ | $92.25_{\pm 0.17}$ | 88.38 |
| $\text{RoB}_{\text{large}}$(AdaLoRA) | 3.54 M | $\mathbf{90.84}_{\pm \mathbf{0.03}}$ | $96.29_{\pm 0.19}$ | $67.61_{\pm 0.12}$ | $91.12_{\pm 0.26}$ | $94.82_{\pm 0.11}$ | $86.28_{\pm 0.36}$ | $89.89_{\pm 0.31}$ | $92.27_{\pm 0.16}$ | 88.64 |
| $\text{RoB}_{\text{large}}$(SaLoRA) | 3.54 M | $90.67_{\pm 0.07}$ | $\mathbf{96.63}_{\pm \mathbf{0.28}}$ | $68.37_{\pm 0.34}$ | $91.95_{\pm 0.08}$ | $\mathbf{94.98}_{\pm \mathbf{0.09}}$ | $\mathbf{87.80}_{\pm \mathbf{1.29}}$ | $\mathbf{91.81}_{\pm 1.57}$ | $\mathbf{92.43}_{\pm \mathbf{0.18}}$ | **89.33** |

### *4.2. Task-Agnostic Performance*

**Models and Datasets.** We present the experiments conducted to evaluate the performance of the self-instruct tuned LLaMA-7B models on instruction-following data [21]. Our objective was to assess their capability in comprehending and executing instructions for arbitrary tasks. We evaluated model performance on two text style transfer datasets: Yelp [22] and GYAFC [23]). Text style transfer refers to the task of changing the style of a sentence to the desired style while preserving the style-independent content. The prompts used in these experiments can be found in Appendix C.

**Implementation Details.** We tuned the learning rate $\eta_p$ from $\{8 \times 10^{-5}, 3 \times 10^{-5}, 1 \times 10^{-4}, 3 \times 10^{-4}, 8 \times 10^{-4}, 1 \times 10^{-3}\}$ and kept the following hyperparameters fixed: $r = 8, b = 0.5,$

$\eta_c = 0.01$, $\beta = 0.1$, $\tau = 1.0$, $\gamma = -0.1$ and $\zeta = 1.1$. All models were fine-tuned on an NVIDIA A800 (80 G) GPU.

Furthermore, we compared the performance of SaLoRA with dataset-specific style transfer models, including StyTrans [15], StyIns [16] and TSST [17]. In contrast to SaLoRA, these models were trained on a specific dataset. To evaluate the performance of style transfer models, we used the following metrics: (1) Transfer accuracy (ACC) using a fine-tuned BERT-base [39] classifier with each dataset. (2) Semantic similarity to human references via BLEU [40] score. (3) Sentence fluency (PPL) via perplexity, as measured by KenLM [41].

**Main Results.** Table 2 presents our experimental results on the Yelp and GYAFC datasets. Compared with LoRA, our method SaloRA achieved better or comparable performance across all directions on both datasets. This demonstrates the effectiveness of our method. In the negative-to-positive transfer direction, though SaloRA's transfer accuracy was lower than the dataset-specific models (e.g., StyIns achieved 92.40 compared with SaloRA's 71), it still aligned with the human reference accuracy of 64.60. Furthermore, SaloRA exhibited a lower perplexity (PPL) compared with dataset-specific models. These results show that SaLoRA (including LoRA) aligns more closely with human writing tendencies. In the formal-to-informal transfer direction, we also observed that our transfer accuracy was lower than dataset-specific models. This disparity may be attributed to the inherent bias of a large model for generating more formal outputs. This can be verified from the fact that SaLoRA exhibited a significant improvement in the transfer accuracy compared with dataset-specific models.

**Table 2.** Automatic evaluation results on Yelp and GYAFC datasets. ↑ indicates that higher values mean better performance, and vice versa.

| | **Yelp** | | | | | | **GYAFC** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model and Method** | **Negative to Positive** | | | **Positive to Negative** | | | **Informal to Formal** | | | **Formal to Informal** | | |
| | **ACC ↑** | **BLEU ↑** | **PPL ↓** | **ACC ↑** | **BLEU ↑** | **PPL ↓** | **ACC ↑** | **BLEU ↑** | **PPL ↓** | **ACC ↑** | **BLEU ↑** | **PPL ↓** |
| Reference | 64.60 | 100 | 102.62 | 93.80 | 100 | 77.53 | 88.44 | 100 | 66.86 | 87.63 | 100 | 105.28 |
| StyTrans | 88.40 | 25.85 | 173.35 | 94.20 | 24.90 | 141.88 | 32.81 | 54.91 | 144.15 | 80.86 | 27.69 | 201.78 |
| StyIns | **92.40** | 25.98 | 116.01 | 89.60 | 26.08 | 105.79 | 54.73 | 60.87 | 96.53 | 80.57 | 30.25 | 132.54 |
| TSST | 91.20 | 28.95 | 112.86 | 94.40 | 28.83 | 101.92 | 65.62 | **61.83** | 87.04 | **85.87** | 33.54 | 128.78 |
| LaA$_{7B}$ | 2.20 | **33.58** | 208.69 | 0.80 | 31.12 | 156.14 | 12.01 | 60.18 | 189.78 | 7.75 | 34.61 | 145.43 |
| LaA$_{7B}$(LoRA) | 71.00 | 25.96 | 82.20 | 92.80 | 31.83 | **83.03** | 89.34 | 61.06 | 68.52 | 34.45 | **41.59** | 82.96 |
| LaA$_{7B}$(SaLoRA) | 73.20 | 24.76 | **76.49** | **94.60** | **31.96** | 87.41 | **89.63** | 61.76 | **67.53** | 39.04 | 40.91 | **79.54** |

### *4.3. Analysis*

**The Effect of Rank** $r$**.** Figure 4 illustrates the experimental results of fine-tuning RoBERTa-large across different ranks. We see that the rank $r$ significantly influenced the model's performance. Both large and small values of $r$ led to suboptimal results. This observation emphasizes that selecting the optimal value for $r$ through heuristic approaches is not always feasible. Notably, SaLoRA consistently improved performance across all ranks when compared with the baseline LoRA. This suggests that SaLoRA effectively captured the "intrinsic rank" of the incremental matrix.

**The Effect of Sparsity** $b$**.** Figure 5 shows the experimental results of fine-tuning RoBERTa-large across various levels of sparsity. Remarkably, SaLoRA consistently exhibited enhanced performance across all sparsity levels compared with the baseline. This result suggests that SaLoRA's modifications facilitated the acquisition of the "intrinsic rank" of the incremental matrix under different sparsities. It is noteworthy that SaLoRA's performance even surpassed the results of LoRA under low sparsity conditions (0.125). The fact that SaLoRA can outperform LoRA even under low sparsity conditions highlights its capacity to capture and leverage parameters with a constrained budget. Consequently, SaLoRA's efficacy could be expanded on a limited budget, making it a versatile method with a broader range of applications.

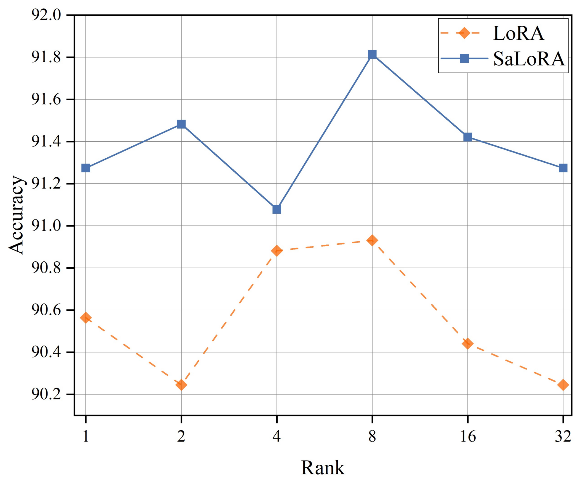

**Figure 4.** Fine-tuning performance under different ranks.

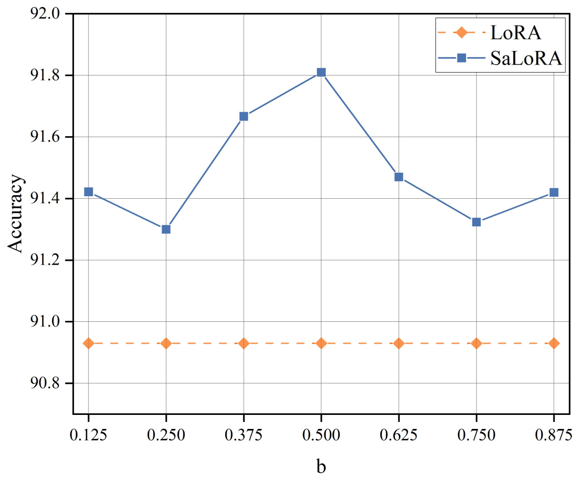

**Figure 5.** Fine-tuning performance under different sparsity levels.

**Ablation Study.** We investigated the impact of Lagrangian relaxation and orthogonal regularization in SaLoRA. Specifically, we compared SaLoRA with the following variants: (i) SaLoRA$_{\lambda=0}$: SaLoRA without Lagrangian relaxation; (ii) SaLoRA$_{\beta=0}$: SaLoRA without orthogonal regularization. These variations involved the fine-tuning of the RoBERTa-base model on the CoLA, STS-B, and MRPC datasets. The target sparsity was set to 0.5 by default. SPS represented the expected sparsity of the incremental matrix. From Table 3, we see that:

1. Without Lagrangian relaxation, the parameter budget was uncontrollable, being 0.37, 0.42 and 0.43 on the three datasets, respectively. Such results highlight the pivotal role that Lagrangian relaxation plays in controlling the allocation of the parameter budget. Nonetheless, it is worth noting that omitting Lagrange relaxation may lead to slight enhancements in performance. However, given the emphasis on control over the parameter budget, this incremental enhancement should be disregarded.
2. Without orthogonal regularization, the performance of SaLoRA degenerated. These results validate that incorporating orthogonal regularization into SaLoRA ensures the independence of doublets from one another, leading to a significant enhancement in its performance.

**Table 3.** Ablation studies on Lagrangian relaxation and orthogonal regularization.

| Method | MRPC | | STS-B | | CoLA | |
|---|---|---|---|---|---|---|
| | ACC | SPS | ACC | SPS | ACC | SPS |
| SaLoRA | $\mathbf{90.20}_{\pm\mathbf{0.74}}$ | $0.51_{\pm0.00}$ | $91.13_{\pm0.17}$ | $0.52_{\pm0.00}$ | $63.39_{\pm1.79}$ | $0.52_{\pm0.00}$ |
| SaLoRA$_{\lambda=0}$ | $89.95_{\pm0.49}$ | $0.37_{\pm0.02}$ | $\mathbf{91.20}_{\pm\mathbf{0.12}}$ | $0.42_{\pm0.03}$ | $\mathbf{63.65}_{\pm\mathbf{3.39}}$ | $0.43_{\pm0.02}$ |
| SaLoRA$_{\beta=0}$ | $90.00_{\pm0.94}$ | $0.51_{\pm0.00}$ | $90.78_{\pm0.27}$ | $0.52_{\pm0.00}$ | $62.89_{\pm2.16}$ | $0.52_{\pm0.00}$ |

**Visualization of Four Components.** We plotted the visualization of expected sparsity $\hat{b}$, the Lagrangian multiplier $\lambda$ and $||A^T A - I||_F^2$ and $||B^T B - I||_F^2$ to show whether these four components were regularized by Lagrangian relaxation and orthogonal regularization, respectively. Specifically, we fine-tuned the RoBERTa-base using SaLoRA on the CoLA, STS-B and MRPC datasets. The initial Lagrangian multiplier $\lambda$ was 0 and the target sparsity $b$ was 0.5. From Figure 6, we see that:

1. The expected sparsity $\hat{b}$ decreased from 0.92 to about 0.50, and the Lagrangian multiplier $\lambda$ kept increasing during training. The results indicate that the SaLoRA algorithm placed more emphasis on satisfying the constraints, eventually reaching a trade-off between satisfying the constraints and optimizing the objective function.
2. The values of $||A^T A - I||_F^2$ and $||B^T B - I||_F^2$ could be optimized to a highly negligible level (e.g., 0.001). Therefore, this optimization process enforced orthogonality upon both matrices $A$ and $B$, guaranteeing the independence of doublets from one another.

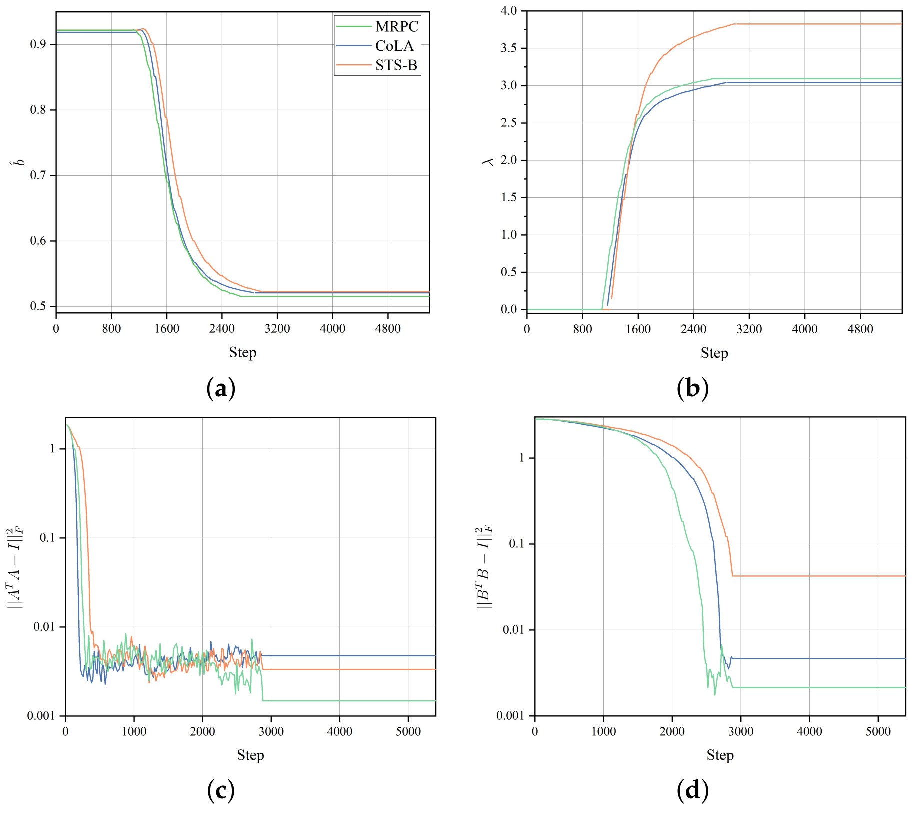

**Figure 6.** Visualization of expected sparsity $\hat{b}$ and the Lagrangian multiplier $\lambda$ under Lagrangian relaxation, and $||A^T A - I||_F^2$ and $||B^T B - I||_F^2$ under orthogonal regularization: (**a**) expected sparsity $\hat{b}$; (**b**) Lagrangian multiplier $\lambda$; (**c**) $A$ of $W^O$ at the first layer; and (**d**) $B$ of $W^O$ at the first layer.

**Comparison of Training Efficiency.** We analyzed the efficiency of SaLoRA in terms of memory and computational efficiency, as shown in Table 4. Specifically, we selected two scales of the RoBERTa model, that is, $RoB_{base}$ and $RoB_{large}$, and measured the peak GPU memory and training time under different batch sizes on an NVIDIA A100 (40 GB) GPU. From Table 4, we see that:

1. The GPU memory usages of both methods were remarkably similar. Such results demonstrate that SaLoRA does not impose significant memory overhead. The reason behind this is that SaLoRA only introduces gate matrices in contrast to LoRA. The total number of parameters was $r \times L \times M$. In this experiment, $r$ denotes the rank of the incremental matrix (set at 8), $L$ corresponds to the number of layers within the model (12 for $RoB_{base}$ and 24 for $RoB_{large}$) and $M$ stands for the number of modules in each layer (set at 6).

2. The training time of SaLoRA increased by 11% when using a batch size of 32 compared with LoRA. This suggests that the additional computational requirements introduced by SaLoRA are justified by its notable gains in performance. This is because SaLoRA is only utilized during a specific training phase ($T_{start}$ to $T_{end}$) comprising 30% of the overall training time. With the remaining 70% being equivalent to LoRA, the overall impact on training time remains manageable.

**Table 4.** Comparison of training efficiency between LoRA and SaLoRA on the MRPC dataset.

| Model | BS | Method | GPU Mem | Time |
|---|---|---|---|---|
| $\text{RoB}_{\text{base}}$ | 16 | LoRA | 3.54 GB | 15 min |
| | | SaLoRA | 3.54 GB | 20 min |
| | 32 | LoRA | 5.34 GB | 14 min |
| | | SaLoRA | 5.35 GB | 15 min |
| | 64 | LoRA | 9.00 GB | 13 min |
| | | SaLoRA | 9.00 GB | 14 min |
| $\text{RoB}_{\text{large}}$ | 16 | LoRA | 7.44 GB | 44 min |
| | | SaLoRA | 7.46 GB | 53 min |
| | 32 | LoRA | 12.16 GB | 40 min |
| | | SaLoRA | 12.18 GB | 44 min |
| | 64 | LoRA | 21.80 GB | 38 min |
| | | SaLoRA | 21.82 GB | 41 min |

**The Resulting Rank Distribution.** Figure 7 shows the resulting rank of each incremental matrix obtained from fine-tuning RoBERTa-base with SaLoRA. We observed that SaLoRA always assigned higher ranks to modules ($W^U$, $W^O$ and $W^V$) and layers (4, 5, 6 and 7). This aligns with the empirical results shown in Figure 1, indicating that modules ($W^U$, $W^O$ and $W^V$) and layers (4, 5, 6 and 7) play a more important role in model performance. Hence, these findings not only validate SaLoRA's effective prioritization of critical modules and layers, but also emphasizes its capacity to learn the structure-aware intrinsic rank of the incremental matrix.

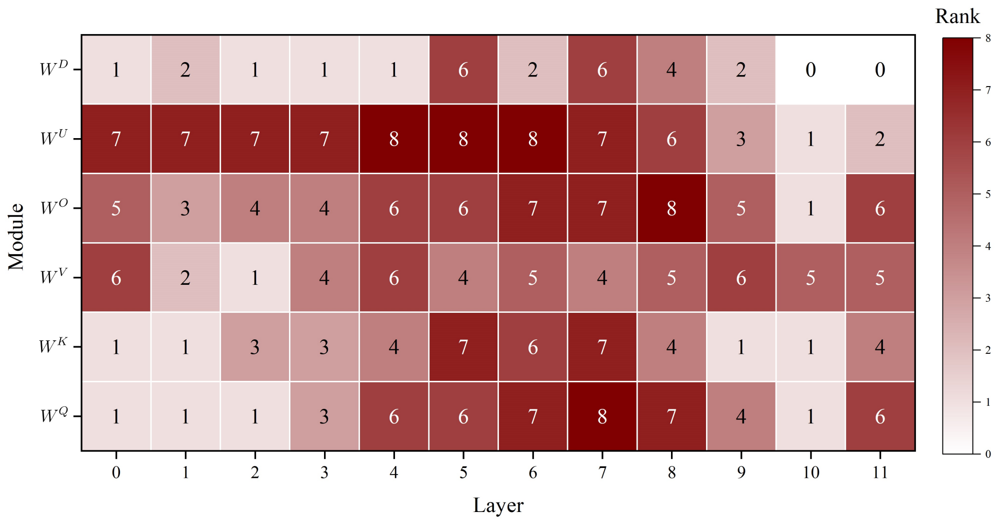

**Figure 7.** The resulting rank of each incremental matrix obtained from fine-tuning RoBERTa-base on MRPC with SaLoRA. The initial rank is set at 8, and the target sparsity is 0.5. The x-axis is the layer index and the y-axis represents different types of modules.

## 5. Conclusions

In this paper, we present SaLoRA, a structure-aware low-rank adaptation method that adaptively learns the intrinsic rank of each incremental matrix. In SaLoRA, we introduced a diagonal gate matrix to adjust the rank of the incremental matrix by penalizing the $L_0$ norm based on the count of activated gates. To enhance training stability and model generalization, we orthogonally regularized $\boldsymbol{B}$ and $\boldsymbol{A}$. Furthermore, we integrated a density constraint and employed Lagrangian relaxation to control the number of valid ranks. In our experiments, we demonstrated that SaLoRA effectively captures the structure-aware intrinsic rank and consistently outperforms LoRA without significantly compromising training efficiency.

**Author Contributions:** Conceptualization, Y.H. and Z.P.; methodology, Y.H. and M.C.; validation, Y.H. and M.C.; formal analysis, Y.H. and Y.X.; investigation, Y.X.; resources, Y.X.; data curation, Y.H.; writing—original draft preparation, Y.H.; visualization, Y.H. and T.W.; supervision, Z.P.; project administration, Z.P.; funding acquisition, Z.P. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the National Natural Science Foundation of China (No. 62076251).

**Data Availability Statement:** Data sharing is not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| PLMs | Pre-trained language models |
| LLMs | Large language models |
| NLP | Natural language process |
| LoRA | Low-rank adaptation |
| MHA | Multi-head self-attention |
| FFN | Feed-forward network |
| FT | Fine-tuning |
| PEFT | Parameter-efficient fine-tuning |
| HC | Hard-concrete distribution |

## Appendix A. Description of Datasets

**Table A1.** Description of datasets.

| Dataset | Description | Train | Valid | Test | Metrics |
|---|---|---|---|---|---|
| | | GLUE Benchmark | | | |
| MNLI | Inference | 393.0k | 20.0k | 20.0k | Accuracy |
| SST-2 | Sentiment analysis | 7.0k | 1.5k | 1.4k | Accuracy |
| MRPC | Paraphrase detection | 3.7k | 408 | 1.7k | Accuracy |
| CoLA | Linguistic acceptability | 8.5k | 1.0k | 1.0k | Matthews correlation |
| QNLI | Inference | 108.0k | 5.7k | 5.7k | Accuracy |
| QQP | Question answering | 364.0k | 40.0k | 391k | Accuracy |
| RTE | Inference | 2.5k | 276 | 3.0k | Accuracy |
| STS-B | Textual similarity | 7.0k | 1.5k | 1.4k | Pearson correlation |
| | | Text Style Transfer | | | |
| Yelp-Negative | Negative reviews of restaurants and businesses | 17.7k | 2.0k | 500 | Accuracy Similarity Fluency |
| Yelp-Positive | Positive reviews of restaurants and businesses | 26.6k | 2.0k | 500 | Accuracy Similarity Fluency |
| GYAFC-Informal | Informal sentences from the Family and Relationships domain | 5.2k | 2.2k | 1.3k | Accuracy Similarity Fluency |
| GYAFC-Formal | Formal sentences from the Family and Relationships domain | 5.2k | 2.8k | 1.0k | Accuracy Similarity Fluency |

## Appendix B. Training Details

We tuned the learning rate $\eta_p$ from $\{5 \times 10^{-5}, 7 \times 10^{-5}, 9 \times 10^{-5}, 2 \times 10^{-4}, 3 \times 10^{-4}, 4 \times 10^{-4}, 5 \times 10^{-4}, 7 \times 10^{-4}\}$ and selected the best learning rate.

**Table A2.** The hyperparameters we used for RoBERTa on the GLUE benchmark.

| | **Model** | **MNLI** | **SST-2** | **CoLA** | **QQP** | **QNLI** | **RTE** | **MRPC** | **STS-B** |
|---|---|---|---|---|---|---|---|---|---|
| # Epoch | $\text{RoB}_{\text{base}}$ | 15 | 20 | 20 | 20 | 15 | 40 | 40 | 30 |
| | $\text{RoB}_{\text{large}}$ | 15 | 20 | 20 | 20 | 15 | 40 | 40 | 30 |
| $\eta_p$ | $\text{RoB}_{\text{base}}$ | $9 \times 10^{-5}$ | $4 \times 10^{-4}$ | $5 \times 10^{-4}$ | $4 \times 10^{-4}$ | $5 \times 10^{-4}$ | $4 \times 10^{-4}$ | $4 \times 10^{-4}$ | $7 \times 10^{-4}$ |
| | $\text{RoB}_{\text{large}}$ | $9 \times 10^{-5}$ | $5 \times 10^{-4}$ | $4 \times 10^{-4}$ | $4 \times 10^{-4}$ | $4 \times 10^{-4}$ | $5 \times 10^{-4}$ | $2 \times 10^{-4}$ | $4 \times 10^{-4}$ |

$T_{start} = 0.2\times$ # Epochs, $T_{end} = 0.5\times$ # Epochs. $r = 8$, $b = 0.5$, $\alpha = 16$, $\eta_c = 0.01$, $\beta = 0.1$, $\tau = 1.0$, $\gamma = -0.1$, $\zeta = 1.1$.

## Appendix C. Prompts

**Table A3.** The prompts used in text style transfer.

**Yelp: Negative → Positive**

"Below is an instruction that describes a task. Write a response that appropriately completes the request.

### Instruction:
{Please change the sentiment of the following sentence to be more positive.}

### Input:
{$Sentence}

### Response:"

Yelp: Positive → Negative

"Below is an instruction that describes a task. Write a response that appropriately completes the request.

### Instruction:
{Please change the sentiment of the following sentence to be more negative.}

### Input:
{$Sentence}

### Response:"

GYAFC: Informal → Formal

"Below is an instruction that describes a task. Write a response that appropriately completes the request.

### Instruction:
{Please rewrite the following sentence to be more formal.}

### Input:
{$Sentence}

### Response:"

GYAFC: Formal → Informal

"Below is an instruction that describes a task. Write a response that appropriately completes the request.

### Instruction:
{Please rewrite the following sentence to be more informal.}

### Input:
{$Sentence}

### Response:"

## References

1. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. In *Proceedings of the Advances in Neural Information Processing Systems, Virtual, 6–12 December 2020*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 1877–1901.
2. Zeng, A.; Liu, X.; Du, Z.; Wang, Z.; Lai, H.; Ding, M.; Yang, Z.; Xu, Y.; Zheng, W.; Xia, X.; et al. Glm-130b: An open bilingual pre-trained model. *arXiv* **2022**, arXiv:2210.02414.
3. Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. LLaMA: Open and Efficient Foundation Language Models. *arXiv* **2023**, arXiv:2302.13971. [CrossRef]
4. OpenAI. GPT-4 Technical Report. *arXiv* **2023**, arXiv:2303.08774. [CrossRef]
5. Pavlyshenko, B.M. Financial News Analytics Using Fine-Tuned Llama 2 GPT Model. *arXiv* **2023**, arXiv:2308.13032. [CrossRef]
6. Kossen, J.; Rainforth, T.; Gal, Y. In-Context Learning in Large Language Models Learns Label Relationships but Is Not Conventional Learning. *arXiv* **2023**, arXiv:2307.12375. [CrossRef]
7. Dong, Q.; Li, L.; Dai, D.; Zheng, C.; Wu, Z.; Chang, B.; Sun, X.; Xu, J.; Li, L.; Sui, Z. A Survey on In-context Learning. *arXiv* **2022**, arXiv:2301.00234. [CrossRef]
8. Li, C.; Farkhoor, H.; Liu, R.; Yosinski, J. Measuring the Intrinsic Dimension of Objective Landscapes. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
9. Aghajanyan, A.; Gupta, S.; Zettlemoyer, L. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Online, 1–6 August 2021; pp. 7319–7328. [CrossRef]
10. Ding, N.; Qin, Y.; Yang, G.; Wei, F.; Yang, Z.; Su, Y.; Hu, S.; Chen, Y.; Chan, C.; Chen, W.; et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nat. Mac. Intell.* **2023**, *5*, 220–235. [CrossRef]
11. Pfeiffer, J.; Kamath, A.; Rücklé, A.; Cho, K.; Gurevych, I. AdapterFusion: Non-Destructive Task Composition for Transfer Learning. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, Online, 19–23 April 2021; pp. 487–503. [CrossRef]
12. Li, X.L.; Liang, P. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Online, 1–6 August 2021; pp. 4582–4597. [CrossRef]
13. Lester, B.; Al-Rfou, R.; Constant, N. The Power of Scale for Parameter-Efficient Prompt Tuning. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Online and Punta Cana, Dominican Republic, 7–11 November 2021; pp. 3045–3059. [CrossRef]
14. Ben Zaken, E.; Goldberg, Y.; Ravfogel, S. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Dublin, Ireland, 22–27 May 2022; pp. 1–9. [CrossRef]
15. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. In Proceedings of the International Conference on Learning Representations, Virtual Event, 25–29 April 2022.
16. Zhang, Q.; Chen, M.; Bukharin, A.; He, P.; Cheng, Y.; Chen, W.; Zhao, T. Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. In Proceedings of the the Eleventh International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
17. Louizos, C.; Welling, M.; Kingma, D.P. Learning Sparse Neural Networks through L_0 Regularization. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
18. Wang, Z.; Wohlwend, J.; Lei, T. Structured Pruning of Large Language Models. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 8–12 November 2020; pp. 6151–6162. [CrossRef]
19. Gallego-Posada, J.; Ramirez, J.; Erraqabi, A.; Bengio, Y.; Lacoste-Julien, S. Controlled Sparsity via Constrained Optimization or: How I Learned to Stop Tuning Penalties and Love Constraints. In *Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA, 29 November–1 December 2022*; Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2022; Volume 35, pp. 1253–1266.
20. Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S.R. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
21. Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; Hashimoto, T.B. Stanford Alpaca: An Instruction-Following LLaMA Model. 2023. Available online: https://github.com/tatsu-lab/stanford_alpaca (accessed on 14 March 2023).
22. Li, J.; Jia, R.; He, H.; Liang, P. Delete, Retrieve, Generate: A Simple Approach to Sentiment and Style Transfer. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), New Orleans, LA, USA, 1–6 June 2018; pp. 1865–1874. [CrossRef]

23. Rao, S.; Tetreault, J. Dear Sir or Madam, May I Introduce the GYAFC Dataset: Corpus, Benchmarks and Metrics for Formality Style Transfer. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), New Orleans, LA, USA, 1–6 June 2018; pp. 129–140. [CrossRef]
24. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.u.; Polosukhin, I. Attention is All you Need. In *Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
25. Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; Morrone, B.; De Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; Gelly, S. Parameter-Efficient Transfer Learning for NLP. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; Volume 97, pp. 2790–2799.
26. Mo, Y.; Yoo, J.; Kang, S. Parameter-Efficient Fine-Tuning Method for Task-Oriented Dialogue Systems. *Mathematics* **2023**, *11*, 3048. [CrossRef]
27. Lee, J.; Tang, R.; Lin, J. What Would Elsa Do? Freezing Layers during Transformer Fine-Tuning. *arXiv* **2019**, arXiv:1911.03090. [CrossRef]
28. Guo, D.; Rush, A.; Kim, Y. Parameter-Efficient Transfer Learning with Diff Pruning. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Online, 1–6 August 2021; pp. 4884–4896. [CrossRef]
29. Valipour, M.; Rezagholizadeh, M.; Kobyzev, I.; Ghodsi, A. DyLoRA: Parameter-Efficient Tuning of Pre-trained Models using Dynamic Search-Free Low-Rank Adaptation. In Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, Dubrovnik, Croatia, 2–6 May 2023; pp. 3274–3287.
30. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
31. Rezende, D.J.; Mohamed, S.; Wierstra, D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In Proceedings of the 1st International Conference on Machine Learning, Bejing, China, 22–24 June 2014; Xing, E.P., Jebara, T., Eds.; PMLR: Bejing, China, 2014; Volume 32, pp. 1278–1286.
32. Brock, A.; Lim, T.; Ritchie, J.; Weston, N. Neural Photo Editing with Introspective Adversarial Networks. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
33. Lin, T.; Jin, C.; Jordan, M. On Gradient Descent Ascent for Nonconvex-Concave Minimax Problems. In Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, 13–18 July 2020; Daumé, H., III, Singh, A., Eds.; PMLR: Vienna, Austria, 2020; Volume 119, pp. 6083–6093.
34. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv* **2019**, arXiv:1907.11692.
35. He, J.; Zhou, C.; Ma, X.; Berg-Kirkpatrick, T.; Neubig, G. Towards a Unified View of Parameter-Efficient Transfer Learning. In Proceedings of the International Conference on Learning Representations, Virtual Event, 25–29 April 2022.
36. Wolf, T.; Debut, L.; Sanh, V.; Chaumond, J.; Delangue, C.; Moi, A.; Cistac, P.; Rault, T.; Louf, R.; Funtowicz, M.; et al. Transformers: State-of-the-Art Natural Language Processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Online, 16–20 November 2020; pp. 38–45. [CrossRef]
37. Mangrulkar, S.; Gugger, S.; Debut, L.; Belkada, Y.; Paul, S. PEFT: State-of-the-Art Parameter-Efficient Fine-Tuning Methods. 2022. Available online: https://github.com/huggingface/peft (accessed on 6 July 2023).
38. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
39. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, MN, USA, 3–5 June 2019; pp. 4171–4186. [CrossRef]
40. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. Bleu: A Method for Automatic Evaluation of Machine Translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, USA, 6–12 July 2002; pp. 311–318. [CrossRef]
41. Heafield, K. KenLM: Faster and Smaller Language Model Queries. In Proceedings of the Sixth Workshop on Statistical Machine Translation, Edinburgh, UK, 30–31 July 2011; pp. 187–197.