Bi-level Finetuning with Task-dependent Similarity Structure for Low-resource Training

Sai Ashish Somayajula[♠,*] **Lifeng Jin**[♣,*] **Linfeng Song**[♣] **Haitao Mi**[♣] **Dong Yu**[♣]
[♠]UC San Diego, USA
ssomayaj@ucsd.edu
[♣]Tencent AI Lab, USA
{lifengjin, lfsong, haitaomi, dyu}@tencent.com

Abstract

Training a large language model in low-resource settings is challenging since they are susceptible to overfitting with limited generalization abilities. Previous work addresses this issue by approaches such as tunable parameters reduction or data augmentation. However, they either limit the trained models' expressiveness or rely on task-independent knowledge. In this paper, we propose the Bi-level Finetuning with Task-dependent Similarity Structure framework where all parameters, including the embeddings for unseen tokens, are finetuned with task-dependent information from the training data only. In this framework, a task-dependent similarity structure is learned in a data-driven fashion, which in turn is used to compose soft embeddings from conventional embeddings to be used in training to update all parameters. In order to learn the similarity structure and model parameters, we propose a bi-level optimization algorithm with two stages—search and finetune—to ensure successful learning. Results of experiments on several classification datasets in low-resource scenarios demonstrate that models trained with our method outperform strong baselines. Ablation experiments further support the effectiveness of different components in our framework. Code is available at https://github.com/Sai-Ashish/BFTSS.

1 Introduction

Finetuning pretrained large models in low-resource scenarios [1] face many challenges (Kann et al., 2020; Hedderich et al., 2021; Şahin, 2022). One of the challenges is the overfitting of the model when finetuned on the small training data. Different approaches have been proposed to tackle this problem and achieved great results. Some approaches restrict the number of parameters to be updated in the finetuning process to avoid overfitting to small amounts of data (Xu et al., 2021). However, parameter restriction while model finetuning may impact the model's expressiveness.

Other methods such as data augmentation (Wei and Zou, 2019; Hoang et al., 2018) aim at increasing the training data size through synthesizing new training data examples to boost generalization ability. These methods rely on either external lexical resources and heuristics, which are limited in domain and language; or pretrained language models, the semantic similarity space of which is not task-dependent. For example, Apple may be replaced by Microsoft in synonym replacement based on some lexical resource, but the "replace-ability" really depends on whether the task is "separate tech companies from gas companies" or "find companies with their headquarters in Washington state", the information of which pretrained language models or static lexical resources are not able to provide.

Our motivation is to combine strengths of both approaches. For generalization ability to unseen data, all parameters, especially embeddings of words not in the training data, should participate in the finetuning. At the same time, no external knowledge source should be required for such finetuning to ensure the method being scalable to different tasks. The ideal training framework for these goals should allow training signals to flow from tokens in training data to unseen tokens in a task-dependent way. Such a framework will ensure that the generalization ability of the trained model is strengthened through finetuning without the risk of overfitting quickly to a small amount of training data.

Our approach proposed in this paper, Bi-level Finetuning with Task-dependent Similarity Structure (BFTSS), aims to meet these goals. First, we propose a low-resource finetuning method where all parameters of the model, including the embeddings of unseen tokens, can be tuned directly through soft embeddings. The soft embeddings

[*]Equal Contribution
[1]Our work is conducted in low resource scenarios that comprise a few hundred instances of data.

Findings of the Association for Computational Linguistics: ACL 2023, pages 8569–8588

are constructed through the use of a similarity matrix with pairwise similarity scores between words, termed a similarity structure[2] in this paper. Second, we propose a bi-level optimization algorithm to learn a task-dependent similarity structure with low-resource task data, where no extra data, knowledge source or task-dependent prior knowledge is required. Since the similarity structure is usually very large, two different methods are proposed to reduce its size and make the learning tractable. Finally, we conduct extensive experiments on different datasets and with different model sizes. Comparison to baseline models and ablated models shows the effectiveness of our approach, where the performance of the models trained in the proposed method surpasses all baselines by large margins.

2 Related work

Low-resource training has been a challenging but important task in natural language processing (Hedderich et al., 2021). Approaches have been proposed to tackle the issues encountered in low-resource training. Robust finetuning methods are applied in such training scenarios, such as approaches restricting tunable parameters (Houlsby et al., 2019; Lee et al., 2019; Chen et al., 2020; Xu et al., 2021) and noise-robust training methods (Jia et al., 2019; Onoe and Durrett, 2019; Jin et al., 2021). They alleviate the overfitting problem but introduce no new information beyond the training data into the model. Data augmentation on the token level (Wei and Zou, 2019; Raiman and Miller, 2017; Vania et al., 2019) as well as on the sentence level using syntax (Şahin and Steedman, 2018; Şahin, 2022), back-translation (Hoang et al., 2018; Xie et al., 2020) or generation models (Ding et al., 2020; Lowell et al., 2021; Liu et al., 2022; Zhou et al., 2022; Wang et al., 2022; Somayajula et al., 2022) are approaches which aims at introducing extra information into model training. Other similar methods rely on pseudo-labeling extra data (Mintz et al., 2009; Le and Titov, 2019; Lison et al., 2020) using task insights and heuristics. Most of them are designed for a specific task, or require external knowledge sources.

Bi-level optimization (BLO) has wide applications in machine learning. The neural architecture search proposed by Liu et al. (2018) uses BLO. It is also used in data selection (Shu et al., 2019; Wang et al., 2020; Ren et al., 2020) and meta-learning (Finn et al., 2017). Feurer et al. (2015) proposed a BLO-based optimization framework for hyperparameter tuning. Baydin et al. (2017) proposed BLO-based learning rate adaptation. Noisy label correction using BLO is proposed by Baydin et al. (2017). Among the papers mentioned above, the lower parameters are the model weights, and the upper parameters are the meta-variables, such as hyperparameters, architecture, training example weights, etc., to be optimized using BLO.

3 Bi-level Finetuning with Task-dependent Similarity Structure

The Bi-level Finetuning with Task-dependent Similarity Structure framework, as shown in Figure 1, centers around how to learn and utilize a task-dependent similarity structure S. The structure is first initialized and learned along with model parameters with bi-level optimization on task data, where soft embeddings of words are derived from the structure to propagate training signals into unseen words. After this first phase of training, named Search phase, the Finetune phase follows in which only model parameters are updated with the similarity structure fixed.

3.1 Motivation and Overview of BFTSS

In BFTSS, a task-dependent similarity structure is first learned with a small training data and then used to improve the performance of the language models. The motivation for the task-dependent similarity structure comes from the observation that only a few words appear in the training data in a low-resource scenario with their word embeddings updated in training. However, we want to pass more information about the unseen words to the model to train it. One way to do that is to identify a word's similar words in the vocabulary and estimate gradients of them from the seen word's gradient. This similarity structure is encoded as a similarity matrix S, with each row corresponding to a word[3] in the vocabulary with size V. The row entries represent the task-dependent semantic proximity of a word with all other words in the vocabulary. Previous methods for data augmentation implicitly

[2]By using "similarity structure" instead of "similarity matrix", we would like to emphasize that 1) the similarity scores are interrelated among words with underlying task-dependent structures, 2) a matrix is just one way of expressing the relations among words.

[3]The smallest string units for a pretrained language model may be words, subwords or other symbols. We use *word* to refer to all items in the tokenization vocabulary.

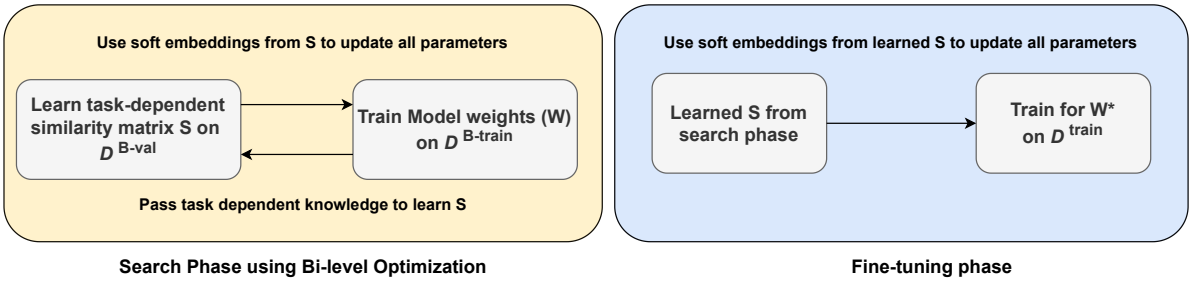

Figure 1: The high level overview of Bi-level Finetuning with Task-dependent Similarity Structure.

assume that the similarity structure for a task is identical to a general similarity structure, which can be derived from pretrained models or gathered from external lexical semantic resources. However, similarity structures are task-specific with varying degrees of closeness to the general similarity structure, as shown in the Apple-Microsoft example. In our framework, they are trained with task data to ensure they are task-specific.

With a task-specific similarity structure, we are able to train all parameters through soft embeddings. Soft embeddings are defined as a linear combination of the embedding vectors weighted by the entries of the S matrix. Intuitively, it means that when a model with parameters W sees a word in the text input, it also sees all the related words weighted by the entries of the corresponding row in S. Thus the optimal model weights learned this way would be dependent on S, i.e., $W^*(S)$.

For training, the task-dependent similarity matrix S should not be learned by reducing the training loss on the model parameters $W^*(S)$, because this is similar to adding more parameters to an already huge language model. Therefore, the task-dependent similarity matrix S is learned using a bi-level optimization approach. Bi-level optimization is used because of the inter-dependency between the optimal model weights W^* and the optimal similarity matrix S^*. The learned optimal model weights $W^*(S)$ depend on S and S is learned in a way that further improves the performance of $W^*(S)$. This shows that both parameters influence and benefit from each other. With bi-level optimization, we are able to first estimate the W parameters by one-step gradient descent with S fixed on one portion of the training data, and learn the S parameter by using the learned W parameter on a different portion. This bi-level optimization phase of W and S is called the Search Phase.

Finally, with the learned S and W from the Search Phase, normal finetuning is conducted in the Finetuning Phase for further tuning the W parameters on the entire training data. The learned S parameters is fixed throughout the phase.

3.2 Similarity structure initialization

The similarity structure is encoded in this work as a similarity matrix $S \in \mathbb{R}^{V \times V}$, where V is the vocabulary size. Each row of the S matrix represents a word in the vocabulary. The row entries represent a word's semantic proximity with all the other words in the vocabulary. One way to initialize this S matrix is to add the inner product of the pretrained language model's embedding matrix $E \in \mathbb{R}^{H \times V}$ with itself (where H is the hidden dimension of the embedding layer of the language model) to the identity matrix:

$$S = \alpha I + (1 - \alpha) f(\{\hat{E}^T \hat{E}\}^d),$$

where $\hat{E}$ is the normalized E matrix where each column vector is a unit norm vector, d is the inverse temperature, $I \in \mathbb{R}^{V \times V}$ is the identity matrix, α is trade-off parameter, and f is the normalizing function that normalizes each row to sum to 1. The identity matrix is added to make the initialization a spiky distribution with the highest weight for the diagonal elements. The pretrained model's embedding layer decides the weights of the off-diagonal elements. These language models are pretrained on a huge corpus of texts and the cosine distance between the embedding vectors depict the semantic proximity between the words. The inner-product matrix is raised to the power of inverse temperature and normalized to make it a smooth and light-tailed distribution. α controls how strong the pretrained embeddings influence the similarity values. By setting α to 1, we have Vanilla finetuning where the similarity terms are not considered. In this paper, we set both terms to have equal weights and omit it in the following sections.

3.3 Soft Embeddings

Soft embeddings are defined as the linear combination of all the related embedding vectors whose weights are determined by the similarity structure S. Formally, we define the soft embedding vector of a word as follows,

$$\hat{e}_i^{(t)} = \sum_{j=0}^{K} s_{i,j}^{(t)} E_j^{(t)} = e_i^{(t)} S^{(t)} \{E^{(t)}\}^T$$

where K is the number of related words used to calculate soft embeddings, $E^{(t)} \in \mathbb{R}^{H \times V}$ and $S^{(t)} \in \mathbb{R}^{V \times V}$ are the embedding matrix and similarity matrix at t^{th} iteration. $E_j^{(t)}$ is the embedding vector of j^{th} word. $s_{i,j}^{(t)}$ is the $\{i,j\}^{th}$ element of the $S^{(t)}$ matrix which describes how similar the j-th word is to the i-th word. $e_i^{(t)}$ is the one-hot representation of the i-th word and $\hat{e}_i^{(t)}$ is the soft embedding of it.

When the model weights are updated with back-propagation, the embeddings of all the similar words (determined by the entries of the similarity matrix S) are updated, propagating task knowledge into all parts of the model:

$$\nabla e_i = \sum_{j=0}^{K} s_{ij}^{(t)} \nabla E_j^{(t)}.$$

3.4 Bi-level learning of a task-dependent similarity structure

Because the similarity structure needs to be trained with task data, a bi-level optimization-based approach is proposed in this work for learning such a task-dependent similarity structure. There are two stages in the bi-level learning process. In the first stage, the model weights W is updated to minimize the loss on one dataset, searching for the optimal model weights $W^*(S)$ on that dataset. In the second stage, the task-dependent similarity matrix S is updated searching for S^* that attains the minimum loss on a different dataset.[4]

[4]The two datasets used in the bi-level optimization are usually the training set and the validation set. However, because we are operating in low-resource scenarios, we split the small training dataset $\mathcal{D}^{\text{train}}$ into two halves to act as these two datasets. The first half $\mathcal{D}^{\text{B-train}}$ is used to train the model in the first stage. The second half $\mathcal{D}^{\text{B-val}}$ is used to train the optimal S parameter in the second stage. This procedure keeps the training data size the same among all baselines for fair comparison.

3.4.1 Training W

In the first stage, model parameters W are trained on BFTSS training set $\mathcal{D}^{\text{B-train}}$ with the similarity matrix S fixed:

$$W^*(S) = \min_W L(W, S, \mathcal{D}^{\text{B-train}}), \quad (1)$$

where W is model parameters, S is the similarity matrix, and L is the task loss. The optimal model weights are learned on $\mathcal{D}^{\text{B-train}}$ given a similarity matrix S. Hence we learn $W^*(S)$, which is dependent on S since W^* depends on the loss function $L(\cdot)$ which is a function of S. S is not updated in this stage because this would overfit the BFTSS training set; instead it will be updated in the second stage.

3.4.2 Training S

In the second stage, the optimal similarity matrix S is learned on BFTSS validation set $\mathcal{D}^{\text{B-val}}$ given the optimal model weights $W^*(S)$ learned in the first stage on $\mathcal{D}^{\text{B-train}}$. The model trained in the first stage $W^*(S)$ is evaluated on $\mathcal{D}^{\text{B-val}}$ and S is updated by minimizing the validation loss. The following optimization problem is solved at this stage:

$$\min_S \quad L(W^*(S), S, \mathcal{D}^{\text{B-val}}). \quad (2)$$

By performing both the stages iteratively with different parameters being fixed at each stage, we do not overfit on the any of the two dataset $\mathcal{D}^{\text{B-train}}$ and $\mathcal{D}^{\text{B-val}}$.

3.4.3 A bi-level optimization framework

Combining both the stages, we have the following bi-level optimization framework:

$$\begin{array}{ll} \min_S & L(W^*(S), S, \mathcal{D}^{\text{B-val}}) \\ s.t. & W^*(S) = \min_W L(W, S, \mathcal{D}^{\text{B-train}}) \end{array} \quad (3)$$

The algorithm consists of two learning stages. From the bottom, the optimization problem corresponds to the learning stage 3.4.1 and then 3.4.2. These two stages are conducted end-to-end. The solution obtained in the first stage $W^*(S)$ is a function of S. We solve for S by minimizing the validation loss in the second stage. The S learned in the second stage changes the training loss in the first stage, which changes the solution $W^*(S)$.

3.5 The full optimization algorithm

The full optimization algorithm, shown in Algo.1 includes two distinct phases. 1) Search phase: In this phase, optimal similarity matrix S^* is estimated by an iterative algorithm to solve the Bi-level optimization problem in Equation 3. The algorithm learns $S' \approx S^*$. 2) Finetune Phase: In this phase, we finetune the language model on the whole $\mathcal{D}^{\text{train}}$ for optimal model weights W^* with a fixed S'.

For optimization in the search phase, we develop a gradient-based optimization algorithm to solve the problem defined in Equation 3 (Liu et al., 2018). W is approximated using the one-step gradient descent.

$$W^*(S) \approx W' = W - \eta_w \nabla_W L(W, S, \mathcal{D}^{\text{B-train}}) \quad (4)$$

W' is plugged into the second-level objective function. The gradient with respect to the S matrix is calculated to update S:

$$S^* \approx S' = S - \eta_s \nabla_S L(W', S, \mathcal{D}^{\text{B-val}}). \quad (5)$$

Gradient of the loss function with respect to S is calculated using the chain rule. W' is an implicit function of S.

$$\begin{aligned}
&\nabla_S L(W', S, \mathcal{D}^{\text{B-val}}) = \\
&\quad \nabla_S L(W - \eta_w \nabla_W L(W, S, \mathcal{D}^{\text{B-train}}), S, \mathcal{D}^{\text{B-val}}) \\
&\quad = \nabla_S L(W', S, \mathcal{D}^{\text{B-val}}) - \eta_w \times \\
&\quad \nabla^2_{S,W} L(W, S, \mathcal{D}^{\text{B-train}}) \nabla_{W'} L(W', S, \mathcal{D}^{\text{B-val}})
\end{aligned}$$

Solving for S' involves an expensive matrix-vector product, whose computational complexity can be reduced by a finite difference approximation:

$$\begin{aligned}
&\nabla^2_{S,W} L(W, S, \mathcal{D}^{\text{B-train}}) \nabla_{W'} L(W', S, \mathcal{D}^{\text{B-val}}) = \\
&\frac{\nabla_S L(W^+, S, \mathcal{D}^{\text{B-train}}) - \nabla_S L(W^-, S, \mathcal{D}^{\text{B-train}})}{2\epsilon},
\end{aligned} \quad (6)$$

where

$$\begin{aligned}
W^{\pm} &= W \pm \epsilon \nabla_{W'} L(W', S, \mathcal{D}^{\text{B-val}}), \\
\epsilon &= \frac{0.01}{\|\nabla_{W'} L(W', S, \mathcal{D}^{\text{B-val}})\|_2}.
\end{aligned}$$

This procedure is carried out iteratively until convergence, at which time the Finetune phase starts. With the trained S' from the concluded first phase, the whole model is further finetuned for optimal weights $W(S')$ on the entire training data in $\mathcal{D}^{\text{train}}$ with S' fixed. This allows the model parameters to be tuned on the unseen $\mathcal{D}^{\text{B-val}}$ as well as $\mathcal{D}^{\text{B-train}}$.

Algorithm 1 Optimization algorithm

Split training dataset $\mathcal{D}^{\text{train}}$ into two halves, $\{\mathcal{D}^{\text{B-train}}, \mathcal{D}^{\text{B-val}}\}$.

# Search phase
while not converged **do**
  Update model weights W using Eq.(4) on $\mathcal{D}^{\text{B-train}}$
  Update similarity matrix S using Eq.(5) on $\mathcal{D}^{\text{B-val}}$
end while

Finetune phase
With the learned S', learn for optimal W on $\mathcal{D}^{\text{train}}$ until convergence.

3.6 Dimensionality reduction of S

The dimension of S is $V \times V$ which is generally difficult to optimize when V is very large. In this section, we discuss ways to reduce the dimensionality of S, making it substantially more convenient to optimize. We propose two different ways to reduce the dimension of S from $V \times V$ to $V \times K$ where $K \ll V$.

BFTSS Top-K : After the S matrix initialization as stated in Section 3.2, we choose the K words with the highest similarity scores and their corresponding indices from each row in the S matrix. The entries corresponding to the top-k words in each row of the S matrix are updated, thus reducing the dimension from $V \times V$ to $V \times K_{\text{top-K}}$ where $K_{\text{top-K}} \ll V$.

BFTSS U-V : From Section 3.2, we initialize S as follows,

$$S = I + f(\{\hat{E}^T \hat{E}\}^d) = I + \hat{S}$$

where $\hat{S} = f(\{\hat{E}^T \hat{E}\}^d)$. S is a full rank matrix because of the added identity matrix, but $\hat{S}$ may not be a full-rank matrix.[5] $\hat{S}$ can be decomposed into a product of two lower-rank matrices. Thus to efficiently reduce the dimension of the S matrix, we apply rank reduction on $\hat{S}$. We use PARAFAC (Bro, 1997) to decompose $\hat{S}$ into two factors U and $V \in \mathbb{R}^{V \times K_{\text{U-V}}}$ of rank $K_{\text{U-V}} \ll$ V such that,

$$S = I + f(\{\hat{E}^T \hat{E}\}^d) = I + \hat{S} \approx I + U \times V^T$$

[5] There is an interdependency between similar words making their corresponding rows dependent.

Reconstruction of S matrix is not needed to perform soft embedding operations. The following operation is performed instead,

$$\begin{aligned}\hat{e}_i^{(t)} &= e_i^{(t)} \times S^{(t)} \times \{E^{(t)}\}^T \\ &= \{E_i^{(t)}\}^T + h(((e_i^{(t)} \times U) \times V^T) \times \{E^{(t)}\}^T),\end{aligned} \tag{7}$$

where h is the Top-K operation which selects similar words dynamically contrary to the static selection in BFTSS Top-K. Multiplication is performed in a specific order to avoid reconstruction of S matrix everytime for soft embeddings. $(e_i^{(t)} \times U) \in \mathcal{R}^{K_{\text{U-V}}}$ is a $K_{\text{U-V}}$-dimensional vector. $\text{V} \in \mathcal{R}^{V \times K_{\text{U-V}}}$. Thus, it boils down to product of a $K_{\text{U-V}}$-dimensional vector with $K_{\text{U-V}} \times V$ dimensional matrix (V^T). It is then multiplied with the embedding matrix to get a H dimensional soft embedding vector. Thus the computational complexity is of the same order as BFTSS Top-K approach.

4 Experiments

4.1 Datasets

We perform experiments on several datasets from GLUE (Warstadt et al., 2018; Wang et al., 2019). The GLUE datasets span a wide range of tasks such as linguistic acceptability (CoLA), semantic textual similarity (STS-B), paraphrase (QQP), natural language inference (RTE, QNLI, MNLI), and sentiment classification (SST-2). To simulate a low-resource finetuning scenario, 100, 300, and 1k examples are sampled from the original training dataset for training. The models are evaluated on the original development set following Xu et al. (2021).

4.2 Baselines

Many different baselines are compared with the proposed method in this paper. Vanilla finetuning is the classic finetuning method where the whole training dataset is used for training. RecAdam (Chen et al., 2020) is an advanced version of weight decay with time-varying coefficients for the cross-entropy loss term and the regularization loss term. Child-D and Child-F (Xu et al., 2021) are methods where a mask is applied to the gradients to restrict the number of parameters being updated to avoid overfitting. Top-K-layer finetuning (Houlsby et al., 2019) only updates the top K layers, and Mixout (Lee et al., 2019) randomly replaced the updated parameters with the pretrained parameters. Finally EDA (Wei and Zou, 2019) is a popular data augmentation method with a dependency on the knowledge resource WordNet (Fellbaum, 1998).

All baselines are evaluated with the pretrained BERT-base and BERT-large models. They are finetuned on the subsampled training datasets. The averaged results on the original development set over ten random seeds are reported following Xu et al. (2021). For BFTSS Top-K and BFTSS U-V, top 50 words and U-V dimension of 100 worked the best among other choices. More information about the hyperparameter tuning process can be found in the appendix.

4.3 Main Results

Table 1 shows the results of the average scores[6] for the proposed and baseline methods. Models trained with our methods are most accurate compared to the baselines over all the sampled data sizes for both BERT-base and BERT-large models, often by large margins. This improvement indicates that our approach to using bi-level optimization to learn a task-dependent similarity structure for finetuning without any external knowledge is very effective in boosting the model's performance over baselines. Among baseline methods, Mixout and Top-K-layer Tuning perform better than other baselines. However, there is still a substantial performance gap between these methods and our proposed methods. For example, BFTSS Top-K method achieves an average gain of 10.58%, 4.73%, and 1.50% over mixout in 100, 300, and 1K training examples scenario, respectively, on the BERT-base model . Our BFTSS U-V method achieves an average gain of 10.75%, 4.77%, and 1.39% over mixout in 100, 300, and 1K training examples scenario, respectively, on the BERT-base model . The trend is similar for BERT-large models and also when compared to Top-K-layer Tuning.

Because Mixout proposes to replace randomly sampled model parameters with pretrained model parameters while finetuning (Lee et al., 2019), and Top-K-layer Tuning only tunes the Top-K layers while freezing the remaining bottom weights, they both can be considered as putting restrictions on model capacity to avoid overfitting. Different from these methods, the proposed methods utilize information about unseen words in the form of a task-dependent similarity structure to serve as an informative prior for the model in finetuning. The model, especially the embeddings of the unseen

[6]Task performance values are in the appendix.

Method	100	300	1000
Vanilla	33.11	46.17	65.28
RecAdam	36.65	44.46	68.28
Child-D	38.38	52.65	66.88
Child-F	38.09	50.89	66.52
Top-K-layer	39.91	58.01	68.47
Mixout	43.97	58.28	68.80
EDA	52.95	56.95	62.92
BFTSS Top-K	***54.55***	***63.01***	**70.30**
BFTSS U-V	**54.72**	**63.05**	***70.19***

(a) Test Results (%) on all datasets with a BERT-base model.

Method	100	300	1000
Vanilla	38.70	56.80	69.31
RecAdam	36.53	56.92	70.16
Child-D	48.05	64.14	71.37
Child-F	47.51	63.05	70.18
Top-K-layer	51.86	64.94	72.05
Mixout	52.98	64.22	72.32
EDA	52.75	60.14	65.04
BFTSS Top-K	***58.00***	**66.53**	***72.86***
BFTSS U-V	**58.10**	***66.50***	**73.11**

(b) Test Results (%) on all datasets with a BERT-large model.

Table 1: Experiment results of the baseline methods and proposed methods. The reported numbers are the averaged evaluation metrics across all tasks with different number of sampled training data examples. **Bold** indicates the highest number, and ***italic*** indicates the second highest.

words, receives informative updates from limited training data. Results here show that by providing more information about unseen words in the vocabulary instead of restricting the tunable parameters, models can be trained to have better generalization without overfitting.

We also compare to the popular data augmentation method, EDA (Wei and Zou, 2019). Different from the baselines above, an external lexical resource, WordNet, is used in EDA for synonym replacement. Our method outperforms EDA in all datasplits despite having no access to any additional data. On the BERT-base model , our BFTSS Top-K method outperforms EDA by an average performance gain of 1.6%, 6.06%, and 7.38% in 100, 300, and 1000 training examples scenarios. Similarly, on the BERT-base model, our BFTSS U-V method outperforms EDA by an average performance gain of 1.77%, 6.1%, and 7.27% in 100, 300, and 1000 training examples scenarios. The trend is similar for the BERT-large models.

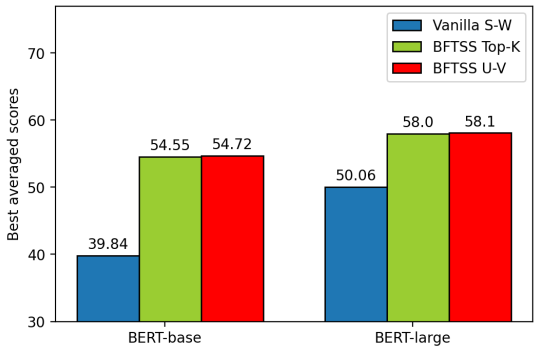

Figure 2: Averaged scores of Vanilla S-W tuning and our approaches with BERT-base and BERT-large model, on 100 data split settings.

EDA can be seen as creating symbolic data augmentations by replacing words according to a general similarity structure coupled with other operations such as random swap, insertion and deletion. With increasing training examples, the accuracy improvement of our method over EDA increases. This result indicates that initially, the general similarity structure is helpful due to low amount of training data compared to no augmentation at all. However, as the training data increases, the general similarity structure along with other heuristics brings more noise than information, resulting in smaller gains and even performance loss compared to Vanilla finetuning. The task-specific similarity structure from our method can benefit the models in all cases, because it is close to the general similarity structure when the training data is small, and moves to a similarity structure tailored for the task when training data increases.

Finally, the two dimensionality reduction methods, Top-K and U-V, perform quite similarly under different conditions, which indicates that both dimensionality reduction methods provides similar benefits to model training.

4.4 Ablation experiments

4.4.1 Vanilla S-W

The effectiveness of the bi-level optimization training is examined in the Vanilla S-W experiments, where Vanilla S-W method represents training the similarity structure S and the model W on the whole training dataset, $\mathcal{D}^{\text{B-train}}$, without the BFTSS framework. We use the U-V procedure for the dimensionality reduction of S. Fig. 2 compares our method with Vanilla S-W on BERT-base and

Method(Top-K)	BERT-base	BERT-large
Vanilla	33.11	38.70
Random S	40.04	43.78
Initial S	42.33	51.01
BFTSS Random	50.68	51.13
BFTSS	**54.55**	**58.00**

(a) Impact of initialization of S and using BFTSS Top-K at 100 data split settings. We report the average scores on all datasets

Method (U-V)	BERT-base	BERT-large
Vanilla	33.11	38.70
Random S	29.30	28.45
Initial S	42.51	52.46
BFTSS Random	37.01	33.56
BFTSS	**54.72**	**58.10**

(b) Impact of initialization of S and using BFTSS U-V at 100 data split settings. We report the average scores on all datasets.

Table 2: Results of experiments showing the impact of initialization of S. The Top-K and U-V indicate if Top-K or U-V, respectively, was used for dimensionality reduction of S.

BERT-large in 100 training examples scenario. Results show our method outperforms Vanilla S-W by a large margin. In Vanilla S-W, both the S and W parameters are learned on the training dataset without the bi-level optimization framework. Compared to Vanilla finetuning where S is not used, performance from Vanilla S-W models are much higher, indicating that the initial similarity structure is already helpful for updating all parts of the model. However, the bi-level learning of S and W is able to provide further benefit for model learning.

4.4.2 Initialization of S

Initial values in the S matrix are important to the success of the BFTSS. Experiments are conducted using the following initialization methods:

- Random S: Use a randomly initialized S for Finetune phase of the optimization algorithm without learning for a task-dependent similarity matrix.
- Initial S: Use the initial S for Finetune phase of the optimization algorithm without learning for a task-dependent similarity matrix.
- BFTSS Random: Perform the optimization algorithm on a randomly initialized S.

Table 2 compares our method with the three initialization methods of S described above. The first two initialization methods do not learn S with task data. Results show that our method outperforms both initialization methods emphasizing the need to learn for a task-dependent S. Interestingly, Random S Top-K outperforms Vanilla, indicating that when S is initialized randomly, static similar word selection strategy (BFTSS Top-K) regularizes the model's performance where as dynamic similar word selection strategy (BFTSS U-V) is injecting noise into the model that is making the performance worse. Furthermore, Initial S performs better than Vanilla and Random S for both the rank reduction methods, indicating the initialization's importance.

We further inspect the role of learning of S in our algorithm. By comparing BFTSS , BFTSS Random, and Random S, we can observe that BFTSS Random performs better than Random S. This observation indicates that a learned S is more helpful than a random initialization of S. Furthermore, BFTSS performs better than BFTSS Random, indicating the initialization of S from pretrained language models provides a more informative prior to the similarity structure than a random initialization. The proposed initialization of S is derived from the pretrained model's embedding layer. Such a model is pretrained on a huge corpus of texts in an unsupervised fashion. The model would have learned a latent representation of the words in the vocabulary that captures semantic proximity between them. Thus an Initialized S is better than a Random S. However, since a pretrained model is pretrained on a huge corpus of texts, there is no guarantee that the proposed initialization of S is task-dependent. Hence, we need to further learn for a task-dependent S.

5 Conclusion

To mitigate the impact of overfitting leading to low generalization ability of large language models, especially in low-resource scenarios, we propose Bi-level Finetuning with Task-dependent Similarity Structure-BFTSS, a bi-level optimization framework to learn a task-specfic similarity structure and further enable the generalization ability of updating 'unseen' or 'similar' words of language models on a given task. We also introduce two variants, BFTSS Top-K and BFTSS U-V, to reduce the dimensionality and make computation efficient. Extensive experimental results on various datasets show the

effectiveness of our approaches in low-resource scenarios when comparing to different baseline methods.

6 Limitation

The way the method applies to larger datasets needs further exploration. As the number of training examples increases, the accuracy gain over vanilla finetuning reduces, indicating that our method best works in low-resource scenarios. Another limitation is that we performed experiments only in one language. It will be interesting to apply our method to tasks in other languages and understand the impact of task-dependent similarity structure on the model's performance in those scenarios. BFTSS Top-K and BFTSS U-V methods perform similarly. Scenarios where BFTSS Top-K and BFTSS U-V differ in performance, should be further explored. We plan to address them in our future works.

References

Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. 2017. Online learning rate adaptation with hypergradient descent. *arXiv preprint arXiv:1703.04782*.

Rasmus Bro. 1997. Parafac. tutorial and applications. *Chemometrics and intelligent laboratory systems*, 38(2):149–171.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. *arXiv preprint arXiv:2004.12651*.

Sang Keun Choe, Willie Neiswanger, Pengtao Xie, and Eric Xing. 2022. Betty: An automatic differentiation library for multilevel optimization. *arXiv preprint arXiv:2207.02849*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bosheng Ding, Linlin Liu, Lidong Bing, Canasai Kruengkrai, Thien Hai Nguyen, Shafiq Joty, Luo Si, and Chunyan Miao. 2020. DAGA: Data augmentation with a generation approach for low-resource tagging tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6045–6057, Online. Association for Computational Linguistics.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Matthias Feurer, Jost Springenberg, and Frank Hutter. 2015. Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.

Michael A. Hedderich, Lukas Lange, Heike Adel, Jannik Strötgen, and Dietrich Klakow. 2021. A survey on recent approaches for natural language processing in low-resource scenarios. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2545–2568, Online. Association for Computational Linguistics.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 18–24, Melbourne, Australia. Association for Computational Linguistics.

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

Wei Jia, Dai Dai, Xinyan Xiao, and Hua Wu. 2019. ARNOR: Attention regularization based noise reduction for distant supervision relation classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1399–1408, Florence, Italy. Association for Computational Linguistics.

Lifeng Jin, Linfeng Song, Kun Xu, and Dong Yu. 2021. Instance-adaptive training with noise-robust losses against noisy labels. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5647–5663, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Katharina Kann, Ophélie Lacroix, and Anders Søgaard. 2020. Weakly supervised pos taggers perform poorly on truly low-resource languages. volume 34, pages 8066–8073.

Phong Le and Ivan Titov. 2019. Boosting entity linking performance by leveraging unlabeled documents. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1935–1945, Florence, Italy. Association for Computational Linguistics.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2019. Mixout: Effective regularization to finetune large-scale pretrained language models. *arXiv preprint arXiv:1909.11299*.

Pierre Lison, Jeremy Barnes, Aliaksandr Hubin, and Samia Touileb. 2020. Named entity recognition without labelled data: A weak supervision approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1518–1533, Online. Association for Computational Linguistics.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Yongtai Liu, Joshua Maynez, Gonçalo Simões, and Shashi Narayan. 2022. Data augmentation for low-resource dialogue summarization. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 703–710, Seattle, United States. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization (2017). *arXiv preprint arXiv:1711.05101*.

David Lowell, Brian Howard, Zachary C. Lipton, and Byron Wallace. 2021. Unsupervised data augmentation with naive augmentation and without unlabeled data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4992–5001, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. *arXiv preprint arXiv:2006.04884*.

Yasumasa Onoe and Greg Durrett. 2019. Learning to denoise distantly-labeled data for entity typing. 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)*, pages 2407–2417, Minneapolis, Minnesota. Association for Computational Linguistics.

Jonathan Raiman and John Miller. 2017. Globally normalized reader. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1059–1069, Copenhagen, Denmark. Association for Computational Linguistics.

Zhongzheng Ren, Raymond Yeh, and Alexander Schwing. 2020. Not all unlabeled data are equal: Learning to weight data in semi-supervised learning. *Advances in Neural Information Processing Systems*, 33:21786–21797.

Gözde Gül Şahin. 2022. To augment or not to augment? a comparative study on text augmentation techniques for low-resource NLP. *Computational Linguistics*, 48(1):5–42.

Gözde Gül Şahin and Mark Steedman. 2018. Data augmentation via dependency tree morphing for low-resource languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5004–5009, Brussels, Belgium. Association for Computational Linguistics.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. 2019. Meta-weight-net: Learning an explicit mapping for sample weighting. *Advances in neural information processing systems*, 32.

Sai Ashish Somayajula, Linfeng Song, and Pengtao Xie. 2022. A multi-level optimization framework for end-to-end text augmentation. *Transactions of the Association for Computational Linguistics*, 10:343–358.

Clara Vania, Yova Kementchedjhieva, Anders Søgaard, and Adam Lopez. 2019. A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1105–1116, Hong Kong, China. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Yufei Wang, Can Xu, Qingfeng Sun, Huang Hu, Chongyang Tao, Xiubo Geng, and Daxin Jiang. 2022. PromDA: Prompt-based data augmentation for low-resource NLU tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4242–4255, Dublin, Ireland. Association for Computational Linguistics.

Yulin Wang, Jiayi Guo, Shiji Song, and Gao Huang. 2020. Meta-semi: A meta-learning approach for semi-supervised learning. *arXiv preprint arXiv:2007.02394*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*

(EMNLP-IJCNLP), pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, volume 33, pages 6256–6268. Curran Associates, Inc.

Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. 2021. Raise a child in large language model: Towards effective and generalizable fine-tuning. *arXiv preprint arXiv:2109.05687*.

Ran Zhou, Xin Li, Ruidan He, Lidong Bing, Erik Cambria, Luo Si, and Chunyan Miao. 2022. MELM: Data augmentation with masked entity language modeling for low-resource NER. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2251–2262, Dublin, Ireland. Association for Computational Linguistics.

A Baselines

We include the following baselines in our experiments:

- Vanilla: Vanilla finetuning of the language model on the training dataset.
- RecAdam: Chen et al. (2020) proposed RecAdam optimizer to mitigate the effect of catastrophic forgetting while finetuning a large language model. RecAdam is an advanced version of weight decay with time-varying coefficients for the cross-entropy loss term and the regularization loss term.
- Child-D: Xu et al. (2021) proposed to update the weights of a sub-network within a large language model to tackle the fine-tuning instability and catastrophic forgetting issue. Child-D estimates a static mask with probability p_D from the fisher information matrix at the beginning of the training. p_D = {0.1, 0.2, 0.3}.
- Child-F: Also proposed in (Xu et al., 2021). Unlike Child-D, Child-F utilizes a dynamic mask during gradient computation. At every iteration, a mask is sampled from a Bernoulli distribution parameterized by p_F = {0.2, 0.3, 0.4}.
- Top-K-layer Finetuning: Only the top-K layers are finetuned with the remaining bottom layers frozen (Houlsby et al., 2019). K = {0, 3, 6, 12}.
- Mixout: Lee et al. (2019) proposed to replace the language model parameters with their corresponding pretrained weights while finetuning with probability p = {0.1, 0.2, . . . , 0.8}. This way, the authors propose reducing the model's deviation from the pretrained weights, thereby tackling the catastrophic forgetting issue.

B Hyperparameter Settings

For all methods, we finetune the pretrained BERT-base[7] and BERT-large[8] models provided by Huggingface (Wolf et al., 2020). We follow the same settings for the models as Devlin et al. (2018). All models are trained using a batch size of 16, a warm-up ratio of 10%, and AdamW optimizer (Loshchilov and Hutter, 2019) with β_1 = 0.9, β_2 = 0.999, ϵ = 1e-6.

For one dataset, the averaged task performance is calculated using model performance evaluated from ten runs with different random seeds. The average scores, reported in the result tables in the main paper, are the averaged value of all task performance numbers. The task performance numbers are reported in the tables below in the appendix.

For our method the training can be decomposed into two phases, 1) Search phase and 2) Finetune phase. We report the hyperparameter settings for each phase and grid searched for the best hyperparameter setting for a task following Xu et al. (2021).

- Search phase: We grid search for $K_{\text{top-K}}$ parameter in {50, 100} and $K_{\text{U-V}}$ decomposition dimension in {100, 300}. We split the training dataset $\mathcal{D}^{\text{train}}$ into two halves, $\{\mathcal{D}^{\text{B-train}}, \mathcal{D}^{\text{B-val}}\}$ to be used for stage 3.4.1 and then 3.4.2 of search phase respectively. We use the Adam optimizer for S with β_1 = 0.9, β_2 = 0.999, ϵ = 1e-8. We grid search for the learning rate of S in {5e-06, 4e-05}. We follow the same settings for W as Devlin et al. (2018). We grid search for the learning rate of W in {4e-05, 2e-05, 5e-06} and number of epochs in {3, 6, 9}. Since it is an optimization problem, we search for the hyperparameters from the provided choices that lead to a smooth reduction in the loss curves. Grid search is performed in a similar fashion as Xu et al. (2021).
- Finetune phase: We use the default hyperparameter settings for learning W^* following Devlin et al. (2018). We train for optimal W on full training dataset $\mathcal{D}^{\text{train}}$ in this phase. We use the S^* obtained in the search phase.

We follow the settings reported in the paper and use the code provided for all the baselines. For a fair comparison (Mosbach et al., 2020), we run the baselines for the same number of training steps as our entire algorithm (search phase and finetuning phase). We grid-searched over all choices of hyperparameters reported in their respective papers and report results on the best set of hyperparameters found. We used Betty library (Choe et al., 2022) for the MLO implementation. We use the V100 GPU

[7] https://huggingface.co/bert-base-cased/tree/main
[8] https://huggingface.co/bert-large-cased/tree/main

Dataset	Dev	Metrics
CoLA	1.0k	Matthews Corr
STS-B	1.5k	Spearman Corr
SST-2	872	Accuracy
QQP	40k	F1
QNLI	5.5k	Accuracy
MNLI	9.8k	Accuracy
RTE	277	Accuracy

Table 3: Statistics and metrics of the classification datasets.

machine for all our experiments. Our algorithm executed in a couple of minutes which is comparable to the running time of vanilla finetuning.

C Tables

Table 3 describes the datasets used for the training and evaluation of the models. It reports the number of examples in the dev set and the evaluation metrices for each task. Tables 4-9, shows the comparison of our method with all the baselines mentioned above for each dataset and split settings. Tables 10-13 shows the results of our ablation studies with different initializations of S matrix to understand the impact of initialization. Last column (**Avg**) is the average performance metric of the model for a given approach across all the tasks. Table 14 shows the comparison of our method with Vanilla S-W on BERT-base and BERT-large for each task to understand the impact of BFTSS.

Method	CoLA	STSB	SST-2	QQP	QNLI	MNLI	MNLI-m	RTE	Avg
Vanilla	1.32, 2.9 (4.64)	22.49, 23.77 (50.44)	55.44, 4.98 (64.33)	13.0, 18.13 (46.46)	52.47, 11.88 (69.49)	33.71, 3.35 (39.35)	33.93, 3.6 (39.49)	52.56, 3.32 (55.96)	33.11
RecAdam	0.98, 3.19 (8.02)	33.44, 44.13 (63.09)	53.37, 5.46 (63.07)	29.49, 29.59 (59.48)	54.93, 9.12 (64.51)	35.08, 3.83 (38.7)	34.95, 3.8 (39.23)	50.97, 3.91 (55.96)	36.65
Child-D	1.62, 4.04 (10.16)	41.08, 3.91 (47.82)	63.02, 5.98 (71.1)	16.72, 24.87 (60.34)	62.93, 9.01 (70.16)	34.89, 3.5 (40.15)	35.61, 4.22 (41.99)	51.16, 3.49 (55.6)	38.38
Child-F	2.12, 2.77 (6.02)	37.4, 5.45 (44.04)	62.31, 5.98 (72.36)	21.36, 28.21 (64.09)	60.52, 11.41 (69.63)	34.73, 3.39 (40.03)	35.27, 4.02 (41.56)	51.05, 3.05 (54.51)	38.09
Top-K-layer	2.63, 3.71 (7.38)	45.57, 12.88 (61.98)	65.61, 7.24 (75.92)	17.2, 27.71 (59.5)	63.08, 8.59 (69.23)	36.03, 2.85 (39.83)	36.47, 3.35 (40.76)	52.67, 3.0 (56.32)	39.91
Mixout	6.82, 6.98 (17.63)	58.02, 8.65 (72.54)	64.32, 8.12 (77.52)	28.02, 25.66 (63.41)	65.9, 5.38 (70.91)	37.31, 3.23 (41.13)	38.36, 4.15 (42.96)	52.96, 2.98 (57.4)	43.97
EDA	5.72, 7.34 (14.98)	71.3, 4.39 (76.2)	78.26, 5.46 (83.14)	60.06, 2.31 (63.73)	68.94, 3.94 (71.99)	42.28, 3.42 (47.79)	42.4, 4.04 (48.17)	54.66, 3.2 (57.76)	52.95
BFTSSTop-K	16.27, 8.19 (26.51)	78.29, 1.37 (80.47)	74.71, 6.52 (83.03)	57.75, 3.6 (63.5)	71.7, 2.13 (74.81)	39.98, 2.26 (43.1)	41.66, 2.61 (45.3)	56.03, 1.9 (58.48)	***54.55***
BFTSSU-V	15.84, 8.86 (26.58)	78.56, 1.32 (80.93)	75.54, 4.51 (81.65)	58.37, 3.8 (63.94)	71.84, 2.04 (73.93)	39.98, 2.17 (42.52)	41.72, 2.48 (44.84)	55.96, 2.6 (60.29)	**54.72**

Table 4: Test Results (%) on BERT-base model in 100 data split settings. For each task, average, standard deviation and maximum of the evaluation metric over ten random seeds have been reported in the tables. The format used is average, standard deviation (maximum).

Method	CoLA	STSB	SST-2	QQP	QNLI	MNLI	MNLI-m	RTE	Avg
Vanilla	3.57, 5.81 (15.07)	67.85, 9.14 (74.6)	73.77, 9.14 (81.88)	28.37, 23.51 (61.32)	66.35, 8.74 (74.19)	37.33, 4.55 (41.77)	38.11, 5.39 (43.49)	54.04, 4.19 (58.84)	46.17
RecAdam	6.25, 6.97 (17.02)	68.17, 3.77 (73.69)	68.85, 9.13 (78.67)	11.34, 23.67 (58.51)	69.22, 4.14 (72.58)	38.32, 3.46 (41.4)	39.16, 3.88 (43.05)	54.4, 2.52 (58.48)	44.46
Child-D	4.61, 6.19 (17.58)	78.6, 4.51 (84.31)	79.77, 4.29 (84.86)	49.2, 21.38 (65.58)	71.97, 4.91 (76.95)	39.74, 3.28 (44.45)	40.98, 3.76 (47.06)	56.32, 2.56 (59.57)	52.65
Child-F	3.84, 5.04 (14.6)	79.18, 1.94 (83.25)	79.6, 4.61 (82.22)	39.87, 29.02 (66.05)	70.34, 7.04 (76.19)	38.42, 3.47 (43.4)	39.62, 3.82 (45.61)	56.25, 2.71 (61.37)	50.89
Top-K-layer	22.81, 11.79 (34.59)	80.74, 2.31 (84.25)	84.31, 1.14 (85.89)	62.52, 2.63 (67.2)	73.47, 1.84 (75.36)	40.91, 3.23 (45.67)	42.69, 4.05 (48.54)	56.61, 1.78 (58.84)	58.01
Mixout	21.87, 10.88 (37.67)	81.83, 1.56 (83.81)	83.39, 2.09 (85.78)	57.68, 7.06 (68.63)	74.94, 2.53 (78.24)	43.02, 2.36 (46.82)	45.48, 2.42 (49.42)	58.05, 2.5 (62.82)	58.28
EDA	9.45, 2.88 (15.29)	77.69, 3.69 (82.02)	84.47, 1.64 (86.7)	62.29, 1.25 (63.87)	72.17, 1.39 (74.21)	46.23, 3.95 (52.07)	47.69, 4.07 (51.6)	55.56, 1.79 (59.57)	56.95
BFTSSTop-K	33.18, 4.71 (38.29)	83.77, 0.81 (85.37)	84.68, 1.79 (86.47)	66.89, 1.19 (68.14)	76.59, 1.67 (78.82)	48.4, 2.9 (52.37)	51.18, 3.35 (56.05)	59.35, 1.63 (61.37)	***63.01***
BFTSSU-V	32.86, 5.13 (42.06)	83.8, 0.71 (84.65)	84.87, 1.94 (87.27)	66.25, 1.69 (69.32)	76.97, 0.99 (78.36)	48.97, 2.37 (52.24)	51.64, 2.91 (55.52)	59.03, 2.26 (62.45)	**63.05**

Table 5: Test Results (%) on BERT-base model in 300 data split settings. For each task, average, standard deviation and maximum of the evaluation metric over ten random seeds have been reported in the tables. The format used is average, standard deviation (maximum).

Method	CoLA	STSB	SST-2	QQP	QNLI	MNLI	MNLI-m	RTE	Avg
Vanilla	40.90, 6.09 (45.63)	84.05, 1.54 (85.99)	86.37, 0.7 (87.61)	69.21, 1.31 (70.97)	79.64, 0.95 (80.98)	49.78, 3.45 (54.57)	52.63, 4.11 (57.94)	59.71, 2.55 (63.18)	65.28
RecAdam	43.81, 3.78 (49.95)	86.41, 0.72 (86.87)	87.47, 1.16 (89.56)	70.59, 0.74 (72.22)	80.72, 0.98 (82.06)	55.77, 3.47 (59.7)	58.63, 3.12 (62.36)	62.82, 2.94 (66.43)	68.28
Child-D	42.18, 3.46 (44.96)	84.95, 1.43 (86.39)	86.56, 0.96 (87.61)	69.27, 1.82 (71.87)	79.77, 1.22 (81.29)	53.64, 1.72 (57.52)	56.58, 1.87 (61.1)	62.09, 2.48 (66.79)	66.88
Child-F	41.02, 8.80 (45.0)	84.55, 1.43 (86.31)	86.92, 0.96 (87.84)	69.49, 1.82 (71.52)	79.55, 1.22 (81.05)	53.79, 1.72 (57.17)	56.64, 1.87 (59.12)	60.22, 2.48 (64.98)	66.52
Top-K-layer	43.93, 2.0 (47.6)	85.44, 1.0 (86.69)	86.94, 1.24 (89.11)	70.69, 1.08 (72.09)	80.33, 0.67 (81.44)	57.5, 3.17 (60.44)	60.55, 3.05 (63.11)	62.35, 2.61 (65.7)	68.47
Mixout	45.87, 1.75 (48.49)	86.32, 1.17 (87.61)	86.88, 0.87 (88.42)	68.16, 3.15 (71.61)	80.22, 0.89 (81.27)	58.31, 1.88 (61.88)	60.88, 1.72 (64.27)	63.72, 2.0 (67.15)	68.80
EDA	18.75, 5.36 (27.09)	80.95, 1.62 (83.1)	87.27, 0.69 (88.3)	66.14, 1.05 (68.07)	76.95, 1.2 (78.69)	56.96, 2.14 (60.09)	58.64, 2.03 (61.44)	57.69, 1.52 (60.29)	62.92
BFTSSTop-K	44.33, 4.03 (49.66)	87.05, 0.37 (87.35)	87.9, 1.15 (89.68)	71.33, 0.51 (72.05)	81.08, 0.93 (81.97)	61.69, 1.79 (63.64)	63.99, 1.71 (66.02)	65.05, 2.23 (69.31)	**70.30**
BFTSSU-V	43.77, 2.35 (46.33)	87.18, 0.32 (87.57)	87.66, 0.99 (89.11)	71.15, 0.57 (72.08)	81.3, 0.41 (82.17)	61.46, 1.76 (63.84)	63.97, 1.6 (66.47)	65.05, 2.27 (67.87)	***70.19***

Table 6: Test Results (%) on BERT-base model in 1000 data split settings. For each task, average, standard deviation and maximum of the evaluation metric over ten random seeds have been reported in the tables. The format used is average, standard deviation (maximum).

Method	CoLA	STSB	SST-2	QQP	QNLI	MNLI	MNLI-m	RTE	Avg
Vanilla	8.72, 14.33 (37.75)	51.93, 13.06 (68.51)	58.42, 9.64 (78.33)	9.72, 18.35 (46.86)	61.83, 8.11 (70.05)	33.94, 1.96 (36.33)	33.82, 1.63 (35.81)	51.19, 3.63 (56.32)	38.70
RecAdam	1.48, 2.62 (4.64)	44.85, 13.61 (57.33)	53.56, 2.15 (57.34)	11.98, 23.16 (55.86)	57.02, 4.67 (64.23)	35.92, 2.84 (38.09)	35.71, 3.04 (38.91)	51.7, 2.33 (53.79)	36.53
Child-D	14.62, 11.18 (36.65)	70.62, 14.35 (80.45)	69.69, 8.1 (81.19)	29.64, 28.75 (66.6)	68.09, 5.95 (73.99)	37.91, 3.86 (42.93)	38.78, 3.89 (43.86)	55.09, 4.04 (61.73)	48.05
Child-F	14.42, 10.21 (27.59)	64.23, 16.7 (78.93)	68.58, 8.2 (81.08)	34.19, 25.27 (62.46)	67.8, 5.29 (74.08)	37.69, 2.52 (40.57)	38.41, 3.05 (42.39)	54.8, 4.15 (61.73)	47.51
Top-K-layer	24.67, 15.52 (40.53)	71.88, 6.17 (77.96)	77.13, 7.59 (88.65)	43.79, 25.06 (65.52)	65.09, 6.21 (72.74)	38.17, 3.0 (43.61)	39.28, 4.0 (47.08)	54.84, 4.92 (62.45)	51.86
Mixout	23.37, 13.66 (43.49)	74.03, 9.0 (82.85)	80.22, 8.16 (87.16)	45.21, 15.06 (62.38)	68.56, 5.83 (77.7)	38.2, 3.77 (44.78)	38.82, 3.81 (45.16)	55.42, 4.64 (63.9)	52.98
EDA	7.15, 11.22 (32.74)	71.64, 8.0 (81.33)	82.29, 5.72 (88.76)	53.92, 19.03 (62.52)	70.43, 3.14 (74.04)	41.01, 4.37 (47.35)	41.61, 5.03 (48.6)	53.9, 3.02 (59.21)	52.75
BFTSSTop-K	28.21, 8.83 (38.83)	76.19, 3.48 (81.34)	85.55, 1.81 (87.96)	60.0, 5.9 (68.37)	73.11, 3.91 (77.96)	41.64, 2.01 (44.83)	42.68, 2.42 (45.71)	56.57, 3.5 (63.18)	***58.00***
BFTSSU-V	30.48, 8.22 (44.52)	76.62, 2.16 (79.55)	87.31, 1.7 (89.68)	58.35, 4.78 (63.24)	71.98, 1.85 (74.35)	40.72, 2.14 (43.76)	41.88, 4.04 (50.16)	57.44, 3.93 (67.51)	**58.10**

Table 7: Test Results (%) on BERT-large model in 100 data split settings. For each task, average, standard deviation and maximum of the evaluation metric over ten random seeds have been reported in the tables. The format used is average, standard deviation (maximum).

Method	CoLA	STSB	SST-2	QQP	QNLI	MNLI	MNLI-m	RTE	Avg
Vanilla	41.57, 4.97 (48.7)	75.01, 14.67 (84.06)	83.88, 6.89 (90.37)	40.81, 27.44 (67.27)	74.98, 1.67 (77.54)	39.07, 5.19 (48.02)	40.51, 5.98 (50.34)	58.59, 4.16 (64.62)	56.80
RecAdam	35.42, 9.07 (46.58)	81.05, 1.08 (82.3)	86.03, 1.96 (89.33)	39.19, 29.21 (68.25)	74.93, 3.22 (79.97)	40.13, 2.8 (43.5)	41.27, 5.21 (48.68)	57.36, 4.02 (64.26)	56.92
Child-D	42.32, 3.87 (47.92)	83.19, 2.8 (86.73)	88.75, 1.37 (90.25)	64.39, 3.16 (69.27)	77.53, 4.41 (80.62)	47.27, 5.19 (54.06)	49.68, 6.17 (57.66)	59.96, 3.25 (63.9)	64.14
Child-F	38.7, 7.13 (46.09)	82.55, 2.53 (85.33)	88.76, 1.23 (90.48)	63.27, 6.95 (69.81)	76.0, 2.5 (79.22)	46.62, 5.26 (55.43)	49.08, 5.88 (57.83)	59.42, 3.37 (63.9)	63.05
Top-K-layer	44.64, 4.39 (54.52)	83.48, 2.12 (85.7)	89.24, 0.97 (90.6)	65.08, 5.03 (70.46)	79.11, 1.38 (81.05)	48.01, 4.45 (54.99)	50.24, 5.22 (58.28)	59.75, 3.97 (65.34)	64.94
Mixout	43.46, 5.15 (49.96)	84.75, 2.2 (86.98)	88.67, 0.65 (89.68)	62.97, 7.88 (70.48)	77.37, 2.24 (82.3)	47.33, 7.12 (57.65)	49.12, 8.25 (61.38)	60.11, 2.43 (63.18)	64.22
EDA	13.95, 3.94 (18.07)	81.24, 1.78 (84.65)	89.21, 0.95 (90.48)	63.08, 5.04 (67.34)	74.3, 2.58 (78.13)	51.06, 2.74 (55.08)	53.11, 2.94 (57.5)	55.16, 3.42 (61.01)	60.14
BFTSSTop-K	44.57, 3.28 (49.89)	84.71, 0.9 (86.18)	88.99, 1.0 (90.25)	66.01, 2.74 (70.19)	78.47, 1.57 (81.07)	53.17, 4.65 (57.18)	55.18, 4.86 (59.64)	61.16, 1.59 (63.54)	**66.53**
BFTSSU-V	44.4, 3.83 (51.31)	85.2, 1.07 (86.84)	89.23, 0.8 (90.83)	66.58, 1.92 (69.65)	78.48, 1.83 (80.62)	52.75, 5.8 (58.63)	55.09, 2.85 (58.93)	60.25, 2.16 (64.98)	***66.50***

Table 8: Test Results (%) on BERT-large model in 300 data split settings. For each task, average, standard deviation and maximum of the evaluation metric over ten random seeds have been reported in the tables. The format used is average, standard deviation (maximum).

Method	CoLA	STSB	SST-2	QQP	QNLI	MNLI	MNLI-m	RTE	Avg
Vanilla	47.48 (-)	81.86 (-)	90.25 (-)	71.30 (-)	81.68 (-)	55.72 (-)	61.09 (67.44)	65.09 (-)	69.31
RecAdam	50.66, 2.15 (53.18)	86.97, 1.3 (88.38)	90.28, 0.5 (90.94)	71.46, 1.59 (73.51)	82.74, 1.11 (84.7)	56.41, 4.25 (60.58)	58.72, 4.32 (62.85)	64.04, 2.25 (67.51)	70.16
Child-D	50.37 (-)	82.76 (-)	90.39 (-)	71.79 (-)	83.42 (-)	62.93 (-)	61.24 (67.04)	68.09 (-)	71.37
Child-F	48.44 (-)	82.25 (-)	90.34 (-)	72.15 (-)	83.09 (-)	62.47 (-)	57.19 (65.92)	65.52 (-)	70.18
Top-K-layer	51.73, 2.59 (55.58)	86.95, 1.67 (88.92)	90.56, 0.98 (91.74)	72.66, 1.07 (74.07)	83.4, 0.66 (84.5)	61.13, 3.54 (65.9)	63.55, 3.47 (68.28)	66.39, 3.92 (70.76)	72.05
Mixout	53.52, 2.06 (55.67)	87.91, 0.54 (88.77)	90.4, 0.64 (91.63)	70.75, 2.0 (73.49)	83.03, 1.59 (84.51)	62.06, 3.04 (65.86)	64.92, 3.1 (68.82)	65.92, 2.46 (69.68)	72.32
EDA	22.79, 13.75 (36.45)	85.61, 1.31 (87.14)	90.53, 0.78 (91.97)	68.6, 3.45 (72.24)	79.86, 2.9 (83.14)	56.41, 9.35 (64.02)	58.33, 10.1 (66.52)	58.23, 4.29 (62.45)	65.04
BFTSSTop-K	51.11, 3.67 (55.48)	88.98, 0.5 (89.56)	90.41, 0.3 (90.94)	72.62, 0.76 (73.68)	83.28, 0.68 (84.46)	63.65, 2.61 (67.52)	66.04, 1.73 (68.47)	66.79, 1.48 (68.95)	***72.86***
BFTSSU-V	51.45, 2.78 (54.95)	89.27, 0.33 (89.94)	90.76, 0.75 (91.63)	72.47, 0.98 (74.06)	83.23, 0.88 (84.39)	64.47, 3.67 (68.24)	66.37, 2.21 (68.39)	66.86, 2.65 (71.84)	**73.11**

Table 9: Test Results (%) on BERT-large model in 1000 data split settings. For each task, average, standard deviation and maximum of the evaluation metric over ten random seeds have been reported in the tables. The format used is average, standard deviation (maximum). The results for Vanilla, Child-D, and Child-F were taken from the original paper (Xu et al., 2021). The paper only reported the mean of ten seeds.

Method	CoLA	STSB	SST-2	QQP	QNLI	MNLI	MNLI-m	RTE	Avg
Vanilla	1.32, 2.9 (4.64)	22.49, 23.77 (50.44)	55.44, 4.98 (64.33)	13.0, 18.13 (46.46)	52.47, 11.88 (69.49)	33.71, 3.35 (39.35)	33.93, 3.6 (39.49)	52.56, 3.32 (55.96)	33.11
Random S Top-K	0.95, 2.0 (4.86)	43.12, 15.71 (69.8)	56.4, 5.5 (65.71)	31.29, 25.64 (63.43)	65.65, 5.08 (69.89)	35.01, 3.28 (40.45)	34.94, 3.65 (42.18)	52.96, 2.58 (57.04)	40.04
Initialized S Top-K	1.99, 3.78 (10.3)	52.45, 9.4 (69.0)	63.28, 7.11 (73.17)	30.54, 27.53 (59.87)	65.31, 9.74 (72.93)	35.76, 4.25 (40.57)	36.8, 5.39 (42.95)	52.49, 2.08 (54.51)	42.33
BFTSSRandomTop-K	4.65, 4.16 (14.97)	73.42, 3.85 (78.04)	65.68, 6.23 (75.8)	62.99, 1.12 (64.48)	69.25, 1.05 (70.88)	38.27, 1.45 (41.28)	38.25, 2.45 (40.97)	52.92, 3.72 (56.32)	50.68
BFTSSTop-K	16.27, 8.19 (26.51)	78.29, 1.37 (80.47)	74.71, 6.52 (83.03)	57.75, 3.6 (63.5)	71.7, 2.13 (74.81)	39.98, 2.26 (43.1)	41.66, 2.61 (45.3)	56.03, 1.9 (58.48)	**54.55**

Table 10: Impact of initialization of S and using BFTSS(Top-K) on BERT-base at 100 data split settings. For each task, average, standard deviation and maximum of the evaluation metric over ten random seeds have been reported in the tables. The format used is average, standard deviation (maximum).

Method	CoLA	STSB	SST-2	QQP	QNLI	MNLI	MNLI-m	RTE	Avg
Vanilla	8.72, 14.33 (37.75)	51.93, 13.06 (68.51)	58.42, 9.64 (78.33)	9.72, 18.35 (46.86)	61.83, 8.11 (70.05)	33.94, 1.96 (36.33)	33.82, 1.63 (35.81)	51.19, 3.63 (56.32)	38.70
Random S Top-K	6.31, 5.3 (16.75)	62.4, 13.44 (75.49)	61.72, 9.73 (76.83)	36.71, 31.27 (64.6)	59.54, 7.91 (73.55)	35.5, 3.16 (39.87)	35.77, 3.5 (41.12)	52.27, 4.4 (58.12)	43.78
Initialized S Top-K	19.16, 12.01 (36.34)	68.71, 13.74 (78.67)	80.38, 7.73 (87.27)	39.57, 23.47 (60.9)	66.79, 7.0 (72.69)	37.9, 2.39 (40.64)	39.02, 4.14 (44.59)	56.53, 4.14 (63.54)	51.01
BFTSSRandomTop-K	9.68, 10.43 (27.52)	72.11, 4.28 (77.97)	67.63, 12.15 (83.26)	58.16, 5.0 (63.51)	67.98, 4.17 (72.8)	39.43, 3.14 (43.21)	40.2, 3.65 (44.28)	53.83, 4.53 (64.26)	51.13
BFTSSTop-K	28.21, 8.83 (38.83)	76.19, 3.48 (81.34)	85.55, 1.81 (87.96)	60.0, 5.9 (68.37)	73.11, 3.91 (77.96)	41.64, 2.01 (44.83)	42.68, 2.42 (45.71)	56.57, 3.5 (63.18)	**58.00**

Table 11: Impact of initialization of S and using BFTSS(Top-K) on BERT-large at 100 data split settings. For each task, average, standard deviation and maximum of the evaluation metric over ten random seeds have been reported in the tables. The format used is average, standard deviation (maximum).

Method	CoLA	STSB	SST-2	QQP	QNLI	MNLI	MNLI-m	RTE	Avg
Vanilla	1.32, 2.9 (4.64)	22.49, 23.77 (50.44)	55.44, 4.98 (64.33)	13.0, 18.13 (46.46)	52.47, 11.88 (69.49)	33.71, 3.35 (39.35)	33.93, 3.6 (39.49)	52.56, 3.32 (55.96)	33.11
Random S U-V	0.51, 2.52 (4.64)	-16.31, 53.34 (55.63)	51.23, 5.33 (62.39)	28.19, 29.72 (58.16)	51.36, 7.12 (67.05)	33.68, 3.08 (38.7)	33.59, 3.02 (38.74)	52.17, 2.71 (55.23)	29.3
Initialized S U-V	2.34, 4.73 (13.68)	52.45, 8.97 (68.62)	63.23, 7.0 (77.06)	31.63, 27.98 (61.44)	65.45, 9.16 (71.54)	36.01, 4.51 (40.87)	36.81, 5.47 (43.1)	52.13, 2.58 (55.23)	42.51
BFTSSRandom U-V	1.39, 2.02 (4.64)	37.29, 26.61 (59.0)	52.61, 5.94 (63.88)	33.26, 28.61 (56.49)	52.64, 4.56 (63.43)	34.27, 2.81 (39.29)	34.47, 3.14 (40.93)	50.18, 4.91 (55.96)	37.01
BFTSSU-V	15.84, 8.86 (26.58)	78.56, 1.32 (80.93)	75.54, 4.51 (81.65)	58.37, 3.8 (63.94)	71.84, 2.04 (73.93)	39.98, 2.17 (42.52)	41.72, 2.48 (44.84)	55.96, 2.6 (60.29)	**54.72**

Table 12: Impact of initialization of S and using BFTSS(U-V) on BERT-base at 100 data split settings. For each task, average, standard deviation and maximum of the evaluation metric over ten random seeds have been reported in the tables. The format used is average, standard deviation (maximum).

Method	CoLA	STSB	SST-2	QQP	QNLI	MNLI	MNLI-m	RTE	Avg
Vanilla	8.72, 14.33 (37.75)	51.93, 13.06 (68.51)	58.42, 9.64 (78.33)	9.72, 18.35 (46.86)	61.83, 8.11 (70.05)	33.94, 1.96 (36.33)	33.82, 1.63 (35.81)	51.19, 3.63 (56.32)	38.70
Random S U-V	1.14, 3.03 (6.56)	-14.66, 35.26 (29.0)	50.5, 1.35 (53.56)	22.42, 28.75 (57.26)	52.59, 3.82 (61.47)	32.99, 1.82 (36.94)	33.01, 1.49 (36.07)	49.57, 3.03 (53.07)	28.45
Initialized S U-V	18.82, 10.59 (31.91)	72.48, 7.79 (81.19)	82.49, 4.32 (87.61)	46.12, 18.53 (66.9)	66.71, 7.48 (73.57)	38.38, 2.79 (43.8)	39.18, 3.64 (43.5)	55.52, 3.92 (61.37)	52.46
BFTSSRandom U-V	0.71, 3.28 (5.42)	7.25, 38.19 (53.02)	50.88, 2.18 (55.62)	38.8, 26.77 (57.17)	51.4, 5.45 (65.07)	34.14, 2.38 (38.32)	34.22, 2.76 (40.5)	51.05, 3.38 (55.23)	33.56
BFTSSU-V	30.48, 8.22 (44.52)	76.62, 2.16 (79.55)	87.31, 1.7 (89.68)	58.35, 4.78 (63.24)	71.98, 1.85 (74.35)	40.72, 2.14 (43.76)	41.88, 4.04 (50.16)	57.44, 3.93 (67.51)	**58.10**

Table 13: Impact of initialization of S and using BFTSS(U-V) on BERT-large at 100 data split settings. For each task, average, standard deviation and maximum of the evaluation metric over ten random seeds have been reported in the tables. The format used is average, standard deviation (maximum).

Dataset	Vanilla S-W	Top-K	U-V
CoLA	2.05 (7.38)	16.27 (26.51)	15.84 (26.58)
STSB	55.12 (72.12)	78.29 (80.47)	78.56 (80.93)
SST-2	63.62 (72.36)	74.71 (83.03)	75.54 (81.65)
QQP	16.02 (58.16)	57.75 (63.5)	58.37 (63.94)
QNLI	59.29 (71.77)	71.70 (74.81)	71.84 (73.93)
MNLI	35.60 (40.60)	39.98 (43.1)	39.98 (42.52)
MNLI-m	35.46 (43.27)	41.66 (45.3)	41.72 (44.84)
RTE	51.59 (53.79)	56.03 (58.48)	55.96 (60.29)
Avg	39.84	***54.55***	**54.72**

(a) Comparison of Vanilla $S-W$ with our approaches, with BERT-base model, on 100 data split settings.

Dataset	Vanilla S-W	Top-K	U-V
CoLA	16.29 (35.16)	28.21 (38.83)	30.48 (44.52)
STSB	70.52 (80.68)	76.19 (81.34)	76.62 (79.55)
SST-2	79.02 (86.24)	85.55 (87.96)	87.31 (89.68)
QQP	34.58 (63.55)	60.00 (68.37)	58.35 (63.24)
QNLI	69.24 (75.65)	73.11 (77.96)	71.98 (74.35)
MNLI	38.15 (41.17)	41.64 (44.83)	40.72 (43.76)
MNLI-m	39.46 (43.64)	42.68 (45.71)	41.88 (50.16)
RTE	53.21 (64.26)	56.57 (63.18)	57.44 (67.51)
Avg	50.06	***58.00***	**58.10**

(b) Comparison of Vanilla $S-W$ with our approaches, with BERT-large model, on 100 data split settings.

Table 14: Results for the Vanilla S-W experiments. For each task, average, and maximum of the evaluation metric over ten random seeds have been reported in the tables.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
We discussed in Section 6

☐ A2. Did you discuss any potential risks of your work?
Not applicable. Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
We discussed in Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

We discussed in Section 4.1 and Section Appendix B

☑ B1. Did you cite the creators of artifacts you used?
We discussed in Section 4.1 and Section Appendix B

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Left blank.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Not applicable. Left blank.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
We discussed in Section 4.1 and Table 3.

C ☑ Did you run computational experiments?

We discussed in Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
We discussed in Appendix B.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
We discussed in Section 4.2 and Appendix B.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
We discussed in Appendix B.

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Not applicable. Left blank.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.