Improving language models fine-tuning with representation consistency targets

Anonymous ACL submission

Abstract

Fine-tuning contextualized representations learned by pre-trained language models has become a standard practice in the NLP field. However, pre-trained representations are prone to degradation (also known as representation collapse) during fine-tuning, which leads to instability, sub-optimal performance, and weak generalization. In this paper, we propose a novel fine-tuning method that avoids representation collapse during fine-tuning by discouraging undesirable changes of the representations. We show that our approach matches or exceeds the performance of the existing regularization-based fine-tuning methods across 13 language understanding tasks (GLUE benchmark and six additional datasets). We also demonstrate its effectiveness in low-data settings and robustness to label perturbation. Furthermore, we extend previous studies of representation collapse and propose several metrics to quantify it. Using these metrics and previously proposed experiments, we show that our approach obtains significant improvements in retaining the expressive power of representations.

1 Introduction

Using pre-trained representations for subsequent fine-tuning has been shown to significantly improve performance on a variety of natural language processing (NLP) tasks (Peters et al., 2018; Radford et al., 2019; Devlin et al., 2018; Dai et al., 2019; Yang et al., 2019; Liu et al., 2019b). Pre-trained representations are learned by language models on a large text corpus and capture a wide range of semantic information and general knowledge (Qiu et al., 2020). However, fine-tuning can lead to degradation of the generalizable representations, which subsequently results in overfitting and performance variability. Aghajanyan et al. (2020a) study the degradation of representations, or representation collapse, and define it as the failure of representations to generalize to out-of-distribution tasks (tasks not used during fine-tuning). They show that fine-tuned representations which can fit auxiliary tasks well are more likely to have better generalization performance i.e. better performance on unseen test data of the original task. Hence, reducing representation collapse is a strong constraint that helps to improve performance on the current task, in addition to ensuring that representations can fit auxiliary tasks.

Multi-task learning studies this phenomenon from an optimization perspective and keeps the representations generic by utilizing the auxiliary tasks during fine-tuning. In particular, representations are fine-tuned simultaneously for multiple tasks (task $A$ and auxiliary tasks $B_1, \ldots, B_k$, different tunable linear heads for each task). Multi-task learning helps to ensure that sentence representations do not overfit to a specific target task but instead contain rich information from diverse set of tasks. This leads to improvement in model generalization (performance on unseen test data) for each task. Although multi-task learning is known to be an effective strategy for improving generalization, it suffers from the dependence on auxiliary tasks data which is difficult to obtain.

In this paper, we propose a pseudo multi-task learning method that prevents representation collapse and improves generalization performance during fine-tuning, without actually requiring any auxiliary data. We implement our method as a loss function-based regularization. Furthermore, we introduce a knob which can be controlled to weaken or strengthen regularization compared to multi-task learning.

We compare our method against established regularization approaches which implicitly address the issue of representation degradation. We show that our approach consistently outperforms major fine-tuning methods on seven GLUE classification tasks and six additional tasks (a total of 13 tasks, see Figure 1 for a summary). We observe that...
on auxiliary tasks indicates less degree of representation collapse, and, interestingly, better generalization on the original fine-tuning task. Thus, representation collapse can be viewed as a form of overfitting and controlling it is a sensible method of regularizing the training procedure.

A natural direction to reduce representation collapse would be to simultaneously fine-tune the representations for task \( A \) and auxiliary tasks \( B \)'s. This is a well-known multi-task learning objective:

\[
\min_{\text{lin}} \sum_{(x,y) \in I_A} l_A(\text{lin}(f_{\text{fin}}(x)), y) + \lambda \sum_{i=1}^t \sum_{(x,y) \in I_{B_i}} l_{B_i}(\text{lin}(f_{\text{fin}}(x)), y) \tag{1}
\]

where \( \text{lin} \) is any linear function, \( l_T \) is a loss function for task \( T \), \( I_T \) is the set of labeled samples for task \( T \) and \( f_{\text{fin}} \) is the embedding function of the model being finetuned\(^1\). It prevents overfitting of representations to a single task and improves generalization performance of the finetuned model. However, a drawback of multi-task learning is that it is dependent on the auxiliary tasks data (\( I_{B_i} \)) which is often not available.

In this paper, we present a method inspired by pseudo multi-task learning, which targets representation collapse based on pseudo auxiliary tasks. That is, we want to optimize loss function in equation 1 and benefit from multi-task learning, yet without depending on real auxiliary task data (\( I_{B_i} \)).

Our choice of auxiliary tasks is based on the following observation: pre-trained representations can fit most of the available natural language tasks using a linear head. Suppose the pre-trained representation function (defined by the language model) \( f_{\text{pre}} : \mathcal{I} \rightarrow \mathbb{R}^d \) maps the inputs in \( \mathcal{I} \) to a d-dimensional space. For a regression task \( T^2 \) with labeled samples \( I_T = \{(x_1, y_1), \ldots, (x_N, y_N)\} \), fitting a linear model on the pre-trained representations for a task \( T \) is equivalent to finding optimal linear head weights \( w \in \mathbb{R}^d, b \in \mathbb{R} \) s.t. \( w^T f_{\text{pre}}(x) + b \) approximates the labeling function \( y \) i.e. \( y_i \approx w^T f_{\text{pre}}(x_i) + b \).

**Pseudo auxiliary task:** Each pseudo-task \( B_i \) is defined by a linear function \( (w_i \in \mathbb{R}^d, b_i \in \mathbb{R}) \) on pre-trained embeddings, it has the labeling function

\[y_i \approx w_i^T f_{\text{pre}}(x_i) + b_i\]

\(^1\)In standard multi-task learning, \( \lambda = 1 \) but we can easily extend it to use multi-task learning as a regularizer.

\(^2\)For classification problem with \( k \) classes, this corresponds to existence of \( w_1, \ldots, w_k \in \mathbb{R}^d, b_1, \ldots, b_k \) s.t. \( \text{softmax}(w_i^T f_{\text{pre}}(x) + b_1, \ldots, w_k^T f_{\text{pre}}(x) + b_k) \) approximates the labeling function.
x → w^T f_{\text{pre}}(x) + b_i for an input x and pre-trained representation function f_{\text{pre}}(\cdot).

Since pseudo auxiliary tasks are regression tasks, a natural choice for the loss function is the \( \ell_2 \) loss function i.e. \( l_{B_i^t}(a, b) = ||a - b||_2^2 \). Let \( \mathcal{I} \) be the set of unlabeled inputs for pseudo tasks, then the second term in equation 1 is equal to the following:

\[
\lambda \sum_{i=1}^{I} \min_{\psi} \sum_{x \in \mathcal{I}} ||l_{B_i^t}(f_{\text{fin}}(x)) - w_i^T f_{\text{pre}}(x) - b_i||_2^2
\]

We ignore the first term (loss for task A, \( \mathcal{L}_A \)) as this does not change in our reduction.

**Knob to control the strength of regularization:** equation 2 biases the fine-tuned representations towards a linear transformation of pre-trained representations. For instance, if fine-tuned representations are a linear shift of the pre-trained representations, then the regularization loss is zero (see Section A for further discussion). In order to allow some freedom to the fine-tuning process, we introduce a knob in form of a function family \( \Psi \). Rather than using a linear head, we allow a potentially non-linear head \( \psi \in \Psi \) on fine-tuned embeddings to fit the auxiliary tasks defined above. Thus, the generalized pseudo multi-task loss becomes \( \hat{L}_{\text{aux}} = \lambda \cdot \sum_{i=1}^{T} \min_{\psi \in \Psi} \sum_{x \in \mathcal{I}} (\psi(f_{\text{fin}}(x)) - w_i^T f_{\text{pre}}(x) - b_i)^2 \)

The main challenge in using the above regularization objective is in selecting the suitable distribution for pseudo auxiliary tasks and the resulting intractable loss function with infinite number of terms. We show (in Appendix A) that under some mild assumptions, this can be reduced to a tractable problem with closed form solution.

**Theorem 1.** If for each pseudo auxiliary task \( B_i^t \), the linear weight \( w_i \in \mathbb{N}(0, I_d) \), the bias \( b_i \in \mathbb{N}(0, 1) \) are standard Gaussian random variables, and \( \Psi \) is of the form \( \{w^T \phi(\cdot) + b \ | \ \phi \in \Phi\} \), then

\[
\hat{L}_{\text{aux}} = \lambda \cdot T \cdot \min_{\phi \in \Phi} \sum_{x \in \mathcal{I}} ||f_{\text{pre}}(x) - \phi(f_{\text{fin}}(x))||_2^2
\]

**Regularization Loss:** Based on Theorem 1, we use the following regularization loss in our finetuning for task A

\[
\hat{L}_{\text{reg}} = \lambda \cdot \min_{\phi \in \Phi} \sum_{x \in \mathcal{I}} ||f_{\text{pre}}(x) - \phi(f_{\text{fin}}(x))||_2^2
\]

where \( \mathcal{I} \) are the inputs to pseudo-tasks, \( f_{\text{pre}} \) and \( f_{\text{fin}} \) are embedding functions and \( \Phi \) is a class of functions (selected before finetuning) defining the strength of regularization. Our proposed method capacity-based consolidation of representations (CAPCORT) has the following objective:

\[
\mathcal{L}_{\text{total}} = \mathcal{L}_A + \hat{L}_{\text{reg}}
\]

where \( \mathcal{L}_A \) standard loss (e.g. cross entropy loss for classification) for task A and \( \hat{L}_{\text{reg}} \) is the regularization loss defined above.

**Choosing the right representation** Normally, sentence-level representations are obtained from the final encoder blocks of a transformer model. However, it may be more beneficial to use representations from the lower layers. In fact, Zhang et al. (2020) show that re-initializing weights of the top layers of encoder improves fine-tuning performance, suggesting that representation consistency may not be desirable for the top layers. Thus, we consider a variant of regularization that uses representations from the intermediate layers of encoder.

![Figure 2: Illustration of CAPCORT. Pre-trained model parameters are non-trainable (frozen), function \( \phi \) controls the amount of structural changes in representations.](image)

**3 Related Work**

**Optimization Methods:** Optimizer, learning rate, and use of bias correction are known to play an important role in stabilizing fine-tuning (Zhang et al., 2020; Mosbach et al., 2020a). We use the corrected fine-tuning strategy as our baseline and follow the improved optimization practices for all methods. Training for a large number of iterations with a lower learning rate has also been shown to enhance performance. Substantially longer training is out of the scope of this paper due to resources constraints but can be viewed as a complementary technique to the proposed approaches. Re-initialization of the top few layers improves performance in a few-shot learning setting (Zhang et al., 2020).

**Methods anchored at pre-trained model:** Weight Consolidation and Elastic Weight Consolidation regularize the fine-tuning process by encouraging fine-tuned weights to be close to pre-trained
weights (Chen et al., 2020; Kirkpatrick et al., 2017). Mixout is a variant of Dropout regularization that replaces dropped neurons with the pre-trained model neurons, thereby mixing pre-trained and fine-tuned parameters (Lee et al., 2019).

Weight consolidation is the closest alternative to our method. But in contrast to weight consolidation, CAPCORT does not put direct constraints on weight updates, but rather tries to control the structural changes in representations. Specifically, it prevents the loss of information content of the representations, yet does not necessarily require fine-tuned representations to stay close to the pre-trained representations.

Other loss function based methods: FreeLB uses adversarial training to improve fine-tuning (Zhu et al., 2019). SMART (Jiang et al., 2020) is a trust region-based method that avoids aggressive updates during fine-tuning. R3F (Aghajanyan et al., 2020a) induces bias towards a solution with a locally smooth prediction function and has been shown to outperform FreeLB and SMART. R4F extends R3F by adding a Lipschitz constraint on the top classification layer. Supervised Contrastive Learning (SCL) induces representations of examples with the same label to be close to each other and far from the examples of other classes (Gunel et al., 2020). A major disadvantage of SCL is a requirement for large mini-batch size and, hence, heavy memory consumption. We implement a memory-efficient SCL version but exclude the original implementation from the baselines due to computational cost (see Appendix I).

Additional fine tuning techniques: Data augmentation via back translation, synonym replacement, random deletion, and synthetic noise can improve fine-tuning performance. (Feng et al., 2021; Wei and Zou, 2019). For related work on multi-task learning, parameter-efficient fine-tuning, domain adaptation, and text-to-text fine-tuning and measures of representation quality see Appendix B.

4 Experiments

4.1 Set Up

Datasets We evaluate methods on GLUE benchmark (Wang et al., 2019) and six extra datasets for language model assessment (see Appendix D). For all the datasets, we perform experiments on the full dataset size and few-sample training sets of 250, 500, 1000 data points. To avoid the excessively high computational cost of fine-tuning on large-scale datasets, we limited their full training sets to 10,000 data points (marked with a suffix -10k in Table 7). For few-sample experiments, we fixed the same data subset across all models to avoid performance changes related to data variability. Since test set labels are unavailable, we use development set to report the performance.

Fine-tuning settings: Following Zhang et al. (2020) and Mosbach et al. (2020b), we use BERT-large model with optimization settings that enhance the stability of fine-tuning, i.e. applying bias correction in AdamW optimizer. We fine-tune all models for 5 epochs (unless otherwise specified) at a learning rate of 2e-5, and run each experiment with 5 different seeds. Our implementation is based on the HuggingFace library. For each task we use a batch size of 4 or smaller in case it causes memory issues (see Appendix D). In contrast to prior works (Devlin et al., 2018; Aghajanyan et al., 2020a; Chen et al., 2020), we do not search for the optimal learning rate for our method across different tasks but instead fix the optimization settings across all the experiments. Thus, we evaluate our method under more stringent settings and require it to be robust to the choice of the learning rate. For each method, we search its most optimal hyperparameters by performing evaluation on the unused fraction of the training set (see Appendix D). We also discuss results for a subset of datasets in Appendix L when we search over optimal learning rate and epochs for each method and task. We observe that CAPCORT outperforms all the baseline methods in this case as well. However, due to resource constraint we do not run these experiments for all datasets and dataset size settings.

Performance metrics: For each task, we use a recommended performance metric (Table 7). To compare methods across multiple tasks, we use average rank and mean performance. Method’s rank corresponds to the position of the method in a list of all methods sorted by performance. The minimal and best possible rank is 1. The average rank of a method is obtained by averaging ranks across all tasks. Since standard fine-tuning is susceptible to failed runs that substantially lowers the average resulting performance (Mosbach et al., 2020b), we investigate performance in both cases when failed runs are filtered out or preserved. Failed runs are those with performance close to the majority classifier on the unused part of the training set (Dodge et al., 2020) (see Appendix D for thresholds).
4.2 Choices of Φ

Φ defines the strength of the regularizer. A singleton Φ containing an identity function is the strongest regularizer which keeps fine-tuned representations close to the pre-trained representations. Conversely, for a rich Φ, e.g., deep/wide neural networks, φ can be chosen to reconstruct the lost information in representation, thereby providing a weak regularization. In this paper, we experiment with a singleton Φ containing identity function (CAPCORT-I ) and a shallow multi-layer perceptrons (CAPCORT-MLP ). Our experiments show that CAPCORT-I based on layers 5, 10, and 20 (from the input) exceeds the performance of top layer regularizing objective (see Appendix H). Thus, we regularize the intermediate layer in our experiments (Choice among 5, 10, or 20 is a hyperparameter). We observe that layer 5 is most effective for small training datasets and layer 20 is most effective for large training datasets (see Appendix H). In CAPCORT-MLP , we set the depth of MLP to be 2, keeping the width equal to the representation dimension. By varying the depth from 1 to 5, we observe that lower depth performs better on smaller training sets. Training with large datasets is robust to choice of depth (see Appendix G).

4.3 Baselines

STD++ : standard++, an improved variant of the standard fine-tuning scheme that includes the use of bias correction in AdamW, following recently proposed practices that enhance the stability of fine-tuning (Zhang et al., 2020; Mosbach et al., 2020b).

Weight Consolidation (Kirkpatrick et al., 2017; Chen et al., 2020) an approach that encourages agreement between pre-trained θpre and fine-tuned θfin models weights, by adding a regularization term \( \sum ||θ_i^{fin} - θ_i^{pre}||_2^2 \) to the loss function.

Local smoothness inducing regularization R3F (Aghajanyan et al., 2020a) prevents aggressive model updates by restricting divergence of model outputs upon input perturbation. For model \( f(·) \) and input token embeddings \( x \), R3F adds a regularization term \( KL_{\Sigma}(f(x)||f(x+z)) \) to the loss function, where noise \( z \) is sampled from the Gaussian distribution and \( KL_{\Sigma} \) is the symmetric Kullback–Leibler divergence.

Reinit-top-k (Zhang et al., 2020) improves fine-tuning performance by re-initializing the top-k layers of the encoder (closer to the output) with \( N(0, 0.02^2) \). Following the original study, we perform hyperparameter search for \( k = 2, 4 \) or 6.

Data Augmentation generates augmented samples by adding noise to the training data (keeping the label intact). In our implementation, we add noise \( \epsilon \sim N(0, δ) \) to the token embeddings.

For a detailed description and regularization coefficients of each baseline method, see Appendix E.

4.4 Results

Performance on GLUE and extra tasks: Table 1 shows a comparison of CAPCORT-I and CAPCORT-MLP with major fine-tuning approaches. Our methods outperform other approaches for the majority of tasks - 5/7 GLUE tasks and 5/6 non-GLUE tasks (more details in Table 24, Appendix J). CAPCORT-I and CAPCORT-MLP outperform baseline methods in terms of mean performance and average rank, with improvements in the mean performance over the corrected fine-tuning strategy by 1.65 and 2.06 points respectively for GLUE benchmark, and 4.26 and 4.52 points for non-GLUE benchmark.

Stabilizing Fine-tuning: Similarly to Dodge et al. (2020); Mosbach et al. (2020b); Wang et al. (2019), we observe that the standard fine-tuning procedure is prone to instability and suboptimal convergence (failed run). Our methods demonstrate higher stability and less fraction of failed runs than other approaches (Table 2). In particular, CAPCORT-I incurs the least number of failed runs (<1%) and in case of not filtering out failed runs its performance drops only by 0.02%. In contrast, performance of baseline methods is reduced more significantly when failed runs are not filtered (STD++ : -16.45, DA : -5.16, WC : -2.14, ReInit : -3.33, R3F : -9.58). CAPCORT-I and CAPCORT-MLP achieve lower average standard deviation than other baseline methods in both cases when failed runs are discarded or preserved.

Fine-tuning in few-sample setting To investigate our methods’ robustness to small dataset sizes, we study CAPCORT-MLP and CAPCORT-I performance in limited data settings (Table 3). Notably, CAPCORT-I outperforms the strongest few-shot fine-tuning baseline, ReInit, for the majority of tasks: 8/13 with 250 training data points, 9/13 with 500 training data points, 8/13 with 1000 training data points (see Appendix K).

Overall, we find that CAPCORT-MLP yields performance gain on large-scale datasets, whereas CAPCORT-I is effective for few-sample fine-tuning.
Table 1: Performance for our methods as well as baseline methods on different datasets. Rank refers to the average rank of the method. Low performing fine-tuning runs are filtered. Detailed non-GLUE results in Appendix F. Appendix I discusses results on a few tasks when we search over optimal learning rate/epochs for each task/method.

| Tasks | STD++ | DA | WC | RelInit | R3F  | CAPCORT-I  | CAPCORT-MLP |
|-------|-------|----|----|--------|-----|-------------|-------------|
| GLUE datasets |       |    |    |        |     |             |             |
| RTE   | 70.76 | 73.65 | 72.2 | 70.90 | 70.40 | 74.37       |             |
| MNLI-10k | 65.56 | 65.50 | **66.69** | 65.15 | 65.05 | 65.74 | 65.22 |
| SST   | 92.08 | 92.00 | 92.73 | 92.04 | 92.09 | 93.19       |             |
| MRPC  | 86.84 | 90.67 | 88.6 | 90.98 | 89.9  | **91.49**  | 91.12       |
| QNLI-10k | 87.23 | 87.44 | 87.19 | 87.28 | 86.97 | 87.46       | **87.61**  |
| QQP-10k | 76.74 | 76.44 | 76.25 | 77.22 | 74.90 | 79.03       | **79.30**  |
| COLA  | 59.69 | **63.45** | 61.50 | 61.25 | 62.04 | 62.34       | 62.47       |
| Mean  | 76.98 | 78.45 | 77.88 | 77.83 | 77.33 | 78.63       | **79.04**  |
| Rank  | 6.57  | 4.14  | 4.29 | 3.71  | 5.43  | 1.86        | 2.00        |

| Non-GLUE datasets |       |    |    |        |     |             |             |
|-------------------|-------|----|----|--------|-----|-------------|-------------|
| Mean              | 86.04 | 89.92 | 90.23 | 89.67 | 88.64 | 90.30       | **90.56**  |
| Rank              | 6.50  | 4.33  | 2.50 | 5.67  | 5.33  | 2.17        | **1.50**   |

| Tasks | STD++ | DA | WC | RelInit | R3F  | CAPCORT-I  | CAPCORT-MLP |
|-------|-------|----|----|--------|-----|-------------|-------------|
| 250 Training datapoints |       |    |    |        |     |             |             |
| Mean  | 71.98 | 73.16 | 72.86 | **74.37** | 73.27 | 74.23 | 73.94       |
| Rank  | 5.62  | 4.92  | 4.62 | 3.00  | 4.04  | 2.50        | 3.31        |
| 500 Training datapoints |       |    |    |        |     |             |             |
| Mean  | 73.81 | 76.72 | 76.63 | **77.24** | 75.26 | 77.23 | 76.84       |
| Rank  | 6.08  | 4.38  | 3.69 | 3.31  | 4.85  | 2.77        | 2.92        |
| 1000 Training Datapoints |       |    |    |        |     |             |             |
| Mean  | 77.34 | 79.20 | 79.04 | 79.37 | 78.98 | **79.65**  | 79.28       |
| Rank  | 5.69  | 4.00  | 3.62 | 3.54  | 4.62  | 2.69        | **3.85**   |

Table 2: Stability of fine-tuning results. CAPI : CAPCORT-I , CAPM : CAPCORT-MLP . Frac: fraction of fine-tuning runs filtered due to low performance. Mean, std: mean and standard deviation in performance across all datasets. See Appendix M for details.

| Tasks | STD++ | DA | WC | RelInit | R3F  | CAPCORT-I  | CAPCORT-MLP |
|-------|-------|----|----|--------|-----|-------------|-------------|
| 0%    | 64.71 | 78.58 | 81.4 | 79.97 | **83.99** | 83.04       |             |
| 5%    | 64.14 | 6.14  | 2.89 | 5.89  | 8.02  | **0.60**   | 1.28        |
| 10%   | 64.71 | 78.58 | 81.4 | 79.97 | **83.99** | 83.04       |             |
| 20%   | 64.71 | 78.58 | 81.4 | 79.97 | **83.99** | 83.04       |             |
| 30%   | 64.71 | 78.58 | 81.4 | 79.97 | **83.99** | 83.04       |             |

Table 4: Mean performance over 13 datasets when training with noisy data. CAPI : CAPCORT-I , CAPM : CAPCORT-MLP . See Appendix N for detailed results.

4.5 Robustness to label perturbation

Real-world data can often contain mislabeled samples, which can hinder the training process. Hence, robustness to label noise is a desirable quality of the fine-tuning approaches. Here, we study the performance of the fine-tuning methods under label perturbation. We introduce label noise as follows: let $C = \{1, \ldots, c\}$ be a class of labels and $X = \{(x, y)\}$ be the true dataset for the fine-tuning task. Our fine-tuning method has access to a noisy dataset $X' = \{(x, y')\}$ where $y' = y$ with probability $1 - p$ and sampled uniformly from $\{1, \ldots, c\} \setminus \{y\}$ with probability $p$.

CAPCORT-I and CAPCORT-MLP show the highest resistance to label perturbation, retaining closest to the original performance upon introducing 5-10% noise to the labels (Table 4). The second most mislabel-resistant approach, WC, is also close to our method conceptually, as it discourages the fine-tuned model to deviate from pre-trained weights. Other methods exhibit substantially lower performance when trained on noisy data.

...
5 Degradation of representations during fine-tuning: analytical perspective

A common problem associated with fine-tuning is that representations lose the ample information content of pre-trained models. It reduces the information content to a small subset needed solely for the current task, a phenomenon described as **representation collapse**. It is closely related to **catastrophic forgetting**, a phenomenon observed during sequential training when the model forgets information from earlier tasks after being trained on a new task. In contrast to catastrophic forgetting, representation collapse is measured as the loss in the expressive power of the representations irrespective of the performance on the pre-training task. In this section, we discuss both existing metrics as well as propose new metrics to quantify representation collapse and compare the effect of different methods in mitigating representation collapse.

5.1 Continual Learning perspective

Aghajanyan et al. (2020a) study representation collapse with **probing experiments** as follows: (i) fine-tune a model on a downstream task A, (ii) freeze the encoding layers and train the top linear layer for a different task B. Low performance in the second step implies representation collapse in the first step. To assess robustness of the proposed approach to representation collapse, we perform a series of probing experiments. In our experiments, we use four GLUE and four non-GLUE datasets in the first step and all datasets in the second step except the one used in the first step (Table 5).

![Figure 3: Top: illustration of three different data distributions, $V$ and $V(\lambda_i)$ correspond to $i$th eigenvalue and its associated eigenvector after eigendecomposition of Gram matrix. Data from the left distribution spans all three dimensions, indicating the highest degree of diversity. The intuition behind our metrics is the optimality of fine-tuned representations by capturing their geometric diversity. The intuition behind our metrics is the following: if all representations lie in a small dimensional space such as a straight line or a single point, then they contain little information and are not expressive. But if representations are well spread out and span the entire representation space, then they possess high information capacity.](image)

Table 5: Model is fine-tuned for task A with different methods, then top layer is trained for the remaining 12 tasks and the mean performance is presented. Aver. is average over the choice of task A. CAPI is CAPCORT-I, CAPM is CAPCORT-MLP. AG: AGNEWS-10k, SCIT: SCITAIL-10k, SCIC: SCICITE-10k, QNLI: QNLI-10k, QQP-QQP-10k, MNLI: MNLI-10k.

|       | A STD++ | DA | WC | RelInit | R3F | CAPI | CAPM |
|-------|---------|----|----|---------|----|------|------|
| QNLI  | 37.6    | 37.1 | 44.5 | 37.7 | 36.5 | 41.7 | **49.5** |
| QQP   | 39.8    | 42.6 | 44.5 | 41.2 | 36.3 | **52.4** | 44.1 |
| RTE   | 32.0    | 32.0 | 37.2 | 48.9 | 33.3 | **51.9** | 42.0 |
| MNLI  | 36.3    | 40.6 | 41.3 | **52.5** | 43.0 | 51.4 | 48.5 |
| AG    | 41.1    | 42.5 | 42.3 | 43.3 | 41.4 | 43.7 | **47.8** |
| IMDB  | 45.2    | 44.0 | 43.3 | 42.0 | 44.5 | 47.9 | **48.4** |
| SCIT  | 39.0    | **50.3** | 46.2 | 44.6 | 34.0 | 47.8 | 48.8 |
| SCIC  | 43.9    | 44.7 | 46.3 | 41.4 | 39.1 | **48.3** | 48.1 |

Aver. 39.4 41.7 43.2 44.0 38.5 **48.1** 47.1

We observe that CAPCORT-MLP and CAPCORT-I show high resistance to representation collapse, outperforming other approaches in 6/8 cases (Table 5). For instance, fine-tuning for QNLI-10k in the first step with CAPCORT-MLP results in a mean performance of 49.5 in the second step, whereas the next best baseline results in a mean performance of 44.5.

5.2 Diversity of fine-tuned representations

We illustrate representation collapse metrics from the geometric perspective in Figure 3. The top three plots show three different distributions of data points (representations). Assume that $\lambda_i$ and $V(\lambda_i)$ correspond to $i$th eigenvalue and its associated eigenvector after eigendecomposition of the data matrix. The left distribution spans all three dimensions, indicating the highest degree of
data diversity. Since data points equally lie in all dimensions, all three eigenvectors will be of equal importance and all three eigenvalues will be approximately equal. In contrast, the central distribution spans two axes, leading to a smaller third eigenvalue that corresponds to the “redundant” dimension. Right distribution has all the data points collapsed along one axis, resulting in one eigenvalue being substantially higher than the others. Overall, more uniform distribution of the eigenvalues corresponds to a better representation matrix diversity. In the bottom barplot we show distribution of the top-20 eigenvalues of the fine-tuned representations with CAPCORT-I and std++ after training on QQP dataset with 250 points (Figure 3). Interestingly, CAPCORT-I preserved a closer to uniform eigenvalue distribution, while std++ results in representations with much higher first eigenvalue, indicating representation collapse.

Here, we formalize the intuitive explanation and define the representation collapse metrics based on geometric diversity of sentence-level representations. We compute the gram matrix for the representations $G$ where $G_{i,j} = \langle z_i, z_j \rangle$. From $G$ we obtain eigenvalues $\lambda_1 \geq \cdots \geq \lambda_d$. To measure diversity of representations, we use geometric mean (GM) and harmonic mean (HM) of the eigenvalues:

$$GM = \left( \prod_{i=1}^{d} \lambda_i \right)^{1/d} = \text{Determinant}^{1/d}(G),$$

$$HM = \left( \sum_{i=1}^{d} \frac{1}{\lambda_i} \right)^{-1} = \text{Trace} \left( G^{-1} \right)^{-1}$$

These metrics attain a high value if the representations are well spread out and are zero or close to zero if all/most of the representations lie in a smaller dimension subspace. In contrast to arithmetic mean, geometric and harmonic mean are not as sensitive to outliers. Our metrics are also connected to parameter estimation error for pseudo linear regression tasks when representations are used as features (see Appendix R).

We observe that representations typically lie in 20-dimensional space (Appendix P), meaning that most of the (normalized) eigenvalues except the first 20 are close to 0 even for the pre-trained model, leading to GM and HM being close to zero when computed on the entire set of eigenvalues. Hence, we chose top-$k$ $\lambda_i$ values for $k = 5, 10, 20$ where GM and HM are bounded away from 0.

$$GM-k = \left( \prod_{i=1}^{k} \lambda_i \right)^{1/k}, \quad HM-k = \left( \sum_{i=1}^{k} \frac{1}{\lambda_i} \right)^{-1}$$

We compare CAPCORT-I and CAPCORT-MLP to the existing baselines using GM and HM with $k = 5, 10, 20$ (Table 6). Low GM-$k$ and HM-$k$ indicates representation collapse, when data is spread around the first few eigenvectors. Table 6 shows that CAPCORT-I results in the most diverse representations among all the baseline methods (see Appendix P for detailed results).

### 6 Conclusion

In this paper, we propose novel regularization based finetuning method based on pseudo multi-task learning. We also introduce a knob in our regularization method which can be controlled to get stronger/weaker regularization than multi-task learning. We experiment with two choices of this knob, CAPCORT-I and CAPCORT-MLP and show that they both achieve significant performance gain on GLUE benchmark and six additional tasks, including few-shot learning settings and label noise conditions. We also study the degradation of representations during fine-tuning, representation collapse, and propose new metrics to quantify it. Our methods reduce representation collapse both in terms of newly proposed metrics as well as previously studied metrics using auxiliary task data.

**Guidelines on using the proposed methods:** A very restrictive $\phi$ in equation 3 yields a strong regularization. Between explored methods, CAPCORT-MLP yields better gain on large datasets (>5k training samples) and CAPCORT-I achieves superior performance for small datasets (<1k samples). For CAPCORT-I with small datasets, regularizing representations from a lower layer (5th or 10th) yield the best results. For CAPCORT-MLP , mlp of depth 2 yields good result across all dataset sizes. If training data is noisy, CAPCORT-I performs better for low noise setting (0-10%) and CAPCORT-MLP performs better for high noise setting (20-30%). For ranges not specified here, one can use either CAPCORT-I or CAPCORT-MLP as both improve performance across the spectrum.
References

Steven Abney. 2007. Semisupervised learning for computational linguistics. CRC Press.

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2020a. Better fine-tuning by reducing representational collapse. In International Conference on Learning Representations.

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2020b. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. arXiv preprint arXiv:2012.13255.

Eyal Ben-David, Carmel Rabinovitz, and Roi Reichart. 2020. Perl: Pivot-based domain adaptation for pre-trained deep contextualized embedding models. Transactions of the Association for Computational Linguistics, 8:504–521.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitlit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. arXiv e-prints, pages arXiv–2106.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In Proceedings of the 2006 conference on empirical methods in natural language processing, pages 120–128.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

Rich Caruana. 1997. Multitask learning. Machine learning, 28(1):41–75.

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.

Arman Cohan, Waleed Ammar, Madeleine Van Zuylen, and Field Cady. 2019. Structural scaffolds for citation intent classification in scientific publications. In NAACL.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860.

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.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. arXiv preprint arXiv:2002.06305.

Steven Y Feng, Varun Gangal, Jason Wei, Sarah Chandar, Soroush Vosoughi, Teruko Mitamura, and Edward Hovy. 2021. A survey of data augmentation approaches for nlp. arXiv preprint arXiv:2105.03075.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Francois Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. The journal of machine learning research, 17(1):2096–2030.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics, pages 249–256. JMLR Workshop and Conference Proceedings.

Beliz Gunel, Jingfei Du, Alexis Conneau, and Veselin Stoyanov. 2020. Supervised contrastive learning for pre-trained language model fine-tuning. In International Conference on Learning Representations.

Demi Guo, Alexander M Rush, and Yoon Kim. 2020. Parameter-efficient transfer learning with diff pruning. arXiv preprint arXiv:2012.07463.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don’t stop pretraining: Adapt language models to domains and tasks. arXiv preprint arXiv:2004.10964.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tao Zhao. 2020. Smart: Robust and efficient fine-tuning for pretrained natural language models through principled regularized optimization. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 2177–2190.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. SciTail: A textual entailment dataset from science question answering. In AAAI.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences, 114(13):3521–3526.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In International Conference on Machine Learning, pages 3519–3529. PMLR.

Jens Kringleup, Sonny Kim Kjaerulf, Søren Brunak, Ole Lund, Tudor I Oprea, and Olivier Taboureau. 2016. Chemprot-3.0: a global chemical biology diseases mapping. Database, 2016.
Cheolhyeong Lee, Kyunghyun Cho, and Wanmo Kang. 2019. Mixout: Effective regularization to finetune large-scale pretrained language models. In International Conference on Learning Representations.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. arXiv preprint arXiv:1901.11504.

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

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

Vivek Madan, Ashish Khetan, and Zohar Karnin. 2021. Tadpole: Task adapted pre-training via anomaly detection. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP).

Vivek Madan, Mohit Singh, Uthaipon Tantipongpipat, and Weijun Xie. 2019. Combinatorial algorithms for optimal design. In Conference on Learning Theory, pages 2210–2258. PMLR.

Amil Merchant, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney. 2020. What happens to bert embeddings during fine-tuning? arXiv preprint arXiv:2004.14448.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In Proceedings of the ACL 2010 Conference Short Papers, pages 220–224, Uppsala, Sweden. Association for Computational Linguistics.

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

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020b. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. In International Conference on Learning Representations.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In Proceedings of the 19th international conference on World wide web, pages 751–760.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. arXiv preprint arXiv:1802.05365.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. arXiv preprint arXiv:2005.00052.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. arXiv preprint arXiv:1811.01089.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ding Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. Science China Technological Sciences, pages 1–26.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. arXiv preprint arXiv:1705.08045.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. arXiv preprint arXiv:1706.05098.

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 Proceedings of ICLR.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017. Instance weighting for neural machine translation domain adaptation. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1482–1488.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. arXiv preprint arXiv:1901.11196.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. Advances in neural information processing systems, 32.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2020. Revisiting few-sample bert fine-tuning. In International Conference on Learning Representations.
A Detailed Derivations

Lemma 1. \( \min_{v \in \mathbb{R}^d} \sum_{j=1}^{N} (y_j - v^T b_j)^2 = ||y - B (B^T B)^{\dagger} B^T y||^2_2 \) where \( y_j \) is the \( j \)-th entry of \( y \).

Proof. Let the loss function be

\[ L = \sum_{j=1}^{N} (y_j - v^T b_j)^2 \]

\( L \) is a smooth function with minimizer \( v^* \). Hence, minimum is achieved at a local minimum. Thus, 

\[ \frac{\delta}{\delta v} L|_{v=v^*} = 0 \]

\[ -2 \sum_{j=1}^{N} b_j (y_j - (v^*)^T b_j) = 0 \]

\[ -2 \sum_{j=1}^{N} b_j y_j - b_j (v^*)^T v^* = 0 \]

\[ \left( \sum_{j=1}^{N} b_j b_j^T \right) y_j = \left( \sum_{j=1}^{N} b_j b_j^T \right)^{\dagger} \left( \sum_{j=1}^{N} b_j b_j^T \right) v^* \]

where \( X^* \) is the pseudo inverse which is equal to the inverse if \( X \) is invertible. Else it spans only the space spanned by \( X \). Note that \( \sum_{j=1}^{N} b_j b_j^T = B^T B \) and \( \sum_{j=1}^{N} b_j y_j = B^T y \). So, \( v^* = (B^T B)^{\dagger} B^T y \). Least square error can be written in terms of vector form to get

\[ \min_{v \in \mathbb{R}^d} \sum_{j=1}^{N} (y_j - v^T b_j)^2 = \min_{v \in \mathbb{R}^d} ||y - B v||^2_2 \]

where \( || \cdot ||^2_2 \) for a vector denote the \( \ell_2 \) norm squared. Substituting \( v^* \) we get 

\[ \min_{v \in \mathbb{R}^d} \sum_{j=1}^{N} (y_j - v^T b_j)^2 = ||y - B (B^T B)^{\dagger} B^T y||^2_2 \]

\[ \square \]

Theorem 2. We show that

\[ E_{u \sim N(0, I_d)} \left[ \min_{v \in \mathbb{R}^d} \sum_{j=1}^{N} (u_j^T z_{pre,j} - v^T z_{fjn,j})^2 \right] = \left\| \left( Z_{fjn} (Z_{fjn}^T Z_{fjn})^{\dagger} Z_{fjn} - I_n \right) Z_{pre} \right\|_2 \]

Proof. To simplify notation, we use \( a_j = z_{pre,j} b_j = z_{fjn,j} B = Z_{fjn} \in \mathbb{R}^{N \times d} \) matrix has \( j \)-th row \( b_j \), \( A = Z_{pre} \in \mathbb{R}^{N \times d} \) matrix has \( j \)-th row \( a_j \) and \( X^* \) is the pseudo-inverse of \( X \).

Let

\[ W = E_{u \sim N(0, I_d)} \left[ \min_{v \in \mathbb{R}^d} \sum_{j=1}^{N} (u_j^T z_{pre,j} - v^T z_{fjn,j})^2 \right] \]

From Lemma 1, we get

\[ W = E_{u \sim N(0, I_d)} \left[ A u - B (B^T B)^{\dagger} B^T A u \right] \]

\[ = E_{u \sim N(0, I_d)} \left[ (A - B (B^T B)^{\dagger} B^T A) u \right] \]

Lemma 2. For any matrix \( M \in \mathbb{R}^{d \times d} \), \( E_{u \sim N(0, I_d)} \left[ ||Mu||^2_2 \right] = ||M||^2_F \) where \( ||M||^2_F \) is the forbenius norm of the matrix \( M \).

Proof. Let the \( i, j \)-th entry of \( M \) be \( m_{i,j} \) and the \( j \)-th entry in \( u \) be \( u_j \). Then,

\[ ||Mu||^2_2 = \sum_{i=1}^{d} \sum_{j=1}^{d} m_{i,j} u_j u_j \]

\[ = \sum_{i=1}^{d} \sum_{j=1}^{d} \sum_{k=1}^{d} m_{i,j} m_{i,k} E[u_j u_k] \]

Since \( u \) is a gaussian random variable with mean 0 and covariance matrix \( I_d \), we have \( E[u_j u_k] = 0 \) for \( j \neq k \) and \( E[u_i^2] = 1 \) for all \( i \in [d] \). Thus,

\[ E \left[ ||Mu||^2_2 \right] = \sum_{i=1}^{d} \sum_{j=1}^{d} m_{i,j}^2 = ||M||^2_F \]

Substituting equality from Lemma 2 to \( W \), we get

\[ W = \left\| A - B (B^T B)^{\dagger} B^T A \right\|_2 \]

Using \( ||M||^2_F = ||M||^2_2 \) and substituting back \( A = Z_{pre} \) and \( B = Z_{fjn} \), we get

\[ E_{u \sim N(0, I_d)} \left[ \min_{v \in \mathbb{R}^d} \sum_{j=1}^{N} (u_j^T z_{pre,j} - v^T z_{fjn,j})^2 \right] = \left\| \left( Z_{fjn} (Z_{fjn}^T Z_{fjn})^{\dagger} Z_{fjn} - I_n \right) Z_{pre} \right\|_2 \]

\[ \square \]
Theorem 3.
\[
\left\| (Z_{\text{fin}}(Z_{\text{fin}}^T Z_{\text{fin}})^{-1} Z_{\text{fin}}^T - I_n) Z_{\text{pre}} \right\|^2_2
\]
\[
= \min_{W \in \mathbb{R}^{d \times d}} \sum_{j=1}^{N} \left\| z_{j,\text{pre}}^j - W z_{j,\text{fin}}^j \right\|^2_2
\]

Proof. To simplify notation, we use \( a_j = z_{j,\text{pre}}^j, b_j = z_{j,\text{fin}}^j, B = Z_{\text{fin}} \in \mathbb{R}^{N \times d} \) matrix has \( j \)-th row \( b_j \), \( A = Z_{\text{pre}} \in \mathbb{R}^{N \times d} \) matrix has \( j \)-th row \( a_j \) and \( X^\dagger \) is the pseudo-inverse of \( X \). Let \( W \in \mathbb{R}^{d \times d} \) have \( i \)-th row \( w_i \). We need to compute
\[
L = \min_{W \in \mathbb{R}^{d \times d}} \sum_{j=1}^{N} \left\| a_j - W b_j \right\|^2_2
\]
\[
= \min_{w_1,\ldots, w_d \in \mathbb{R}^d} \sum_{j=1}^{d} \sum_{i=1}^{N} (a_{j,i} - w_i^T b_j)^2
\]
\[
= \sum_{i=1}^{d} \min_{w_i \in \mathbb{R}^d} \sum_{j=1}^{N} (a_{j,i} - w_i^T b_j)^2
\]
where \( a_{j,i} \) is the \( i \)-th entry of \( a_j \). Applying Lemma 1, we get
\[
L = \sum_{i=1}^{d} \left\| (I_n - B(B^T B)^\dagger B^T) c_i \right\|^2_2 \tag{5}
\]
where \( c_i \) is the \( i \)-th column of \( A \) (\( j \)-th entry of \( c_i \) is \( a_{j,i} \)).

Lemma 3. For a matrix \( M \in \mathbb{R}^{N \times N} \) and a set of vectors \( v_1, \ldots, v_k \in \mathbb{R}^N \),
\[
\sum_{i=1}^{k} \left\| M v_i \right\|^2_2 = \left\| M V \right\|^2_2
\]
where \( V \in \mathbb{R}^{N \times k} \) is the matrix with columns \( v_1, \ldots, v_k \).

Proof. Let \( j \)-th row of \( M \) be \( m_j \). Then,
\[
\sum_{i=1}^{k} \left\| M v_i \right\|^2_2 = \sum_{i=1}^{k} \sum_{j=1}^{N} (m_j^T v_i)^2
\]
For \( j \in [N], i \in [k] \), \( (j, i) \)-the entry of \( MV \) is \( m_j^T v_i \). Thus,
\[
\left\| M V \right\|^2 = \sum_{j=1}^{N} \sum_{i=1}^{k} (m_j^T v_i)^2
\]
Combining the two equalities, we get
\[
\sum_{i=1}^{k} \left\| M v_i \right\|^2_2 = \left\| M V \right\|^2_2
\]

Applying Lemma 3 in eq 5, we get
\[
L = \left\| (I_n - B(B^T B)^\dagger B^T) A \right\|^2_2
\]
This finishes the proof of theorem. \( \square \)

B Missing Related works

Multi-task learning: In multi-task learning, we jointly finetune for many tasks where each task has a classification head but share the backbone with several other tasks (Caruana, 1997; Ruder, 2017; Zhang and Yang, 2017; Liu et al., 2019a). This approach however requires access to a large amount of labeled data from unrelated tasks which is typically unavailable. Since our method focuses on the scenario when a single finetuning task data is available, we focus on comparing it against works of a similar nature and do not provide an extensive comparison with these works. It is likely that the clear difference between the methods makes them complementary, but exploring this is outside the scope of this paper.

Domain shift between pre-training and finetuning data: Even though pretrained models achieve high performance for a large number of NLP tasks, they tend to suffer if there is a significant domain shift between the pretraining data and finetuning data. Domain Adaptation bridges this gap by adapting the model to the finetuning task domain. It can be done by doing additional pre-training on task data if such data is available (Gururangan et al., 2020) or algorithmically finding such data from general domain corpus if such a data is not available (Madan et al., 2021).

Domain shift between finetuning train data and evaluation data: Domain Adaptation typically refers to the scenario where labeled training data is available in one domain and the evaluation is done for data in other domain. Techniques for addressing domain shift include model-centric techniques, data-centric techniques and hybrid techniques. Model centric technique changes the model by changing the feature space, loss function or the structure of the model (Blitzer et al., 2006; Pan et al., 2010; Ganin et al., 2016; BengDavid et al., 2020). Data-centric approaches involve pseudo-labeling (Abney, 2007), using auxiliary tasks (Phang et al., 2018), and data selection (Moore and Lewis, 2010; Wang et al., 2017).

Parameter Efficient Finetuning: Rather than storing a model for each of the finetuning task, some
approaches try to keep most of the model parameters frozen and only tune a subset of parameters. Rebuffi et al. (2017) and Pfeiffer et al. (2020) insert adapter layers between the layers of pretrained model and keep the original parameters frozen. Guo et al. (2020) keep the change in model parameters to be sparse. (Aghajanyan et al., 2020b) learns a small dimensional vector whose projection onto a large dimension space added to pretrained model parameter yields the finetuned model. Ben Zaken et al. (2021) show that fine-tuning only the bias parameters can also lead to competitive performance.

Text-to-text finetuning: Autoregressive model such as T5 and GPT-3 cast the finetuning in a text-to-text format (Raffel et al., 2019; Brown et al., 2020). They can work in the few-shot learning setting by framing the finetuning task as the pretraining task. Autoregressive models make it easier to sample text whereas masked Language models such as BERT and RoBERTa are restricted fill in the blanks.

Measures of representation: (Aghajanyan et al., 2020a) measures the quality of finetuned representations by fitting them on auxiliary tasks. CKA (Kornblith et al., 2019) measures correspondences between representations from different network.

(Merchant et al., 2020) also studies what happens during finetuning via probing classifiers and representation similarity analysis. It argues that finetuning does not necessarily incurs catastrophic forgetting. It analyze the effect for finetuning different tasks on the changes in representation.

C Relation to Weight Consolidation

Apart from working with representations instead of model weights, key difference between our method and weight consolidation is the following: our method tries to control the structural change in representations and does not always keep fine-tuned representations close to the pre-trained representations. Regularizing fine-tuned representations to be close to pre-trained representations is just a special case and the most strict form of our method (CAPCORT-I). The knob (Φ) in our method can be selected to chose a regularizer ranging from this extreme case to the other extreme case with no regularization. Note that this knob is not same as the regularization constant which is used to weigh regularization loss compared to cross-entropy loss.

D Experiment Set up details

D.1 Dataset description

| Task | Train | Dev | C | Metric |
|------|-------|-----|---|--------|
| COLA | 8551  | 1043| 2 | MCC    |
| RTE  | 2490  | 277 | 2 | Accuracy|
| SST  | 67349 | 872 | 2 | Accuracy|
| MNLI-10k | 10000 | 9815| 3 | MCC    |
| MRPC | 3668  | 408 | 2 | F1     |
| QQP-10k | 10000 | 40430| 2 | F1     |
| QNLI-10k | 10000 | 5463| 2 | Accuracy|
| CHEMPROT | 4169 | 2427| 13| Micro F1|
| SciCite | 7320 | 916 | 3 | Macro F1|
| SCITAIL-10k | 10000 | 1304| 2 | Accuracy|
| AGNEWS-10k | 10000 | 5000| 4 | Macro F1|
| YELP-10k | 10000 | 10000| 2 | Accuracy|
| IMDB-10k | 10000 | 5000| 2 | Macro F1|

Table 7: Description of datasets - sizes of train set, development set, number of classes (C) and metrics used (MCC denotes Matthews correlation coefficient).

GLUE benchmark contains seven tasks, which cover paraphrase detection (MRPC, QQP), natural language inference (MNLI, RTE, QNLI), linguistic acceptability (CoLA) and sentiment classification (SST-2). We also assembled non-GLUE benchmark from six distinct tasks which include biomedical relation extraction on CHEMPROT (Kringelum et al., 2016), sentiment classification on YELP (Zhang et al., 2015a) and IMDB (Maas et al., 2011), citation intent classification on SCICITE (Cohan et al., 2019), language inference on SCITAIL (Khot et al., 2018) and article topic classification on AGNEWS (Zhang et al., 2015b).

Batch Size: Different methods have different memory requirement. For instance, R3F has the highest footprint which limits the batch size as we can not process too many inputs at the same time. Table 8 shows the batch size used for each dataset in our experiments.

| Task | Batch size |
|------|------------|
| COLA | 4          |
| RTE  | 1          |
| SST  | 4          |
| MNLI-10k | 1        |
| MRPC | 4          |
| QQP-10k | 1         |
| QNLI-10k | 1         |
| CHEMPROT | 1        |
| SciCite | 2         |
| SCITAIL-10k | 1        |
| AGNEWS-10k | 2        |
| YELP-10k | 1         |
| IMDB-10k | 1         |

Table 8: Batch size used in our experiments

Filtering failed runs and data used for failed runs: For most of the datasets experimented here, available test data split is unlabeled. Thus, we use the validation data split to report performance. It has been observed that different finetuning runs
can result in very different finetuned model performance (Mosbach et al., 2020b). Thus, reporting max test run performance does not truly reflect the effectiveness of the finetuning process and the maximum test run performance across different random seeds can be substantially larger than the mean. So, in our experiments we do not use the val data (on which we report performance) to select the run or any hyperparameter. To select optimal hyperparameters such as regularization coefficient etc., we use a subset of original train data split which is not used for training. Such a data is available as we are typically finetuning with a subset of original train data. Table 9 shows the threshold for failed run for each task.

Table 9: Failed run threshold

| Task       | Threshold |
|------------|-----------|
| COLA       | 0.00      |
| RTE        | 53.70     |
| SST        | 54.00     |
| MNLI-10k   | 30.00     |
| MPQ-10k    | 81.22     |
| QQP-10k    | 0.00      |
| QNLI-10k   | 50.53     |
| CHEMPROT   | 34.45     |
| SciCite     | 24.60     |
| AGNEWS-10k| 10.44     |
| YELP-10k   | 52.80     |
| IMDB-10k   | 33.94     |

Local smoothness inducing regularization R3F (Aghajanyan et al., 2020a) For a classification problem, let \( f \) be the probability prediction function corresponding to model being finetuned. It’s input is the input to the first BERT encoder layer (output of token embedding layer). Let \( x_1, \ldots, x_N \) be the outputs of the token embedding layer for the inputs of the finetuning task. For \( i \in [N] \), let \( \epsilon_{\delta,i} \) be a Gaussian noise term with mean 0 and covariance matrix \( \delta I \). We set \( \delta = 1e - 5 \). Then, the regularization loss is

\[
\mathcal{L}_{R3F} = \sum_{i=1}^{N} KL(f(x_i) \parallel f(x_i + \epsilon_{\delta,i})) + KL(f(x_i + \epsilon_{\delta,i}) \parallel f(x_i))
\]

Data Augmentation: Let \( x_1, \ldots, x_N \) be the outputs of the token embedding layer for the inputs of the finetuning task and \( y_1, \ldots, y_N \) be their associated labels. In data augmentation, we add noise to the data during the training process.

\[
\mathcal{L}_{total} = \sum_{i=1}^{N} \mathcal{L}_{CE}(f(x_i), y_i) + \lambda \mathcal{L}_{CE}(f(x_i + \epsilon_{\delta,i}), y_i)
\]

where \( \mathcal{L}_{CE} \) is the cross entropy loss, \( f \) is the prediction function as per the model and \( \epsilon_{\delta,i} \) is a Gaussian noise with mean 0 and covariance matrix \( \delta I \) added to \( x_i \). We set \( \delta = 1e - 5 \).

Table 10 show the regularization coefficients used for each method.

Table 10: Regularization coefficient for different methods.

| Method       | Regularization coefficient |
|--------------|---------------------------|
| CAPCORT-I    | 0.01, 0.05, 0.1, 0.5      |
| CAPCORT-MLP  | 0.01, 0.05, 0.1, 0.5      |
| DA           | 0.05, 0.1, 0.2, 0.4, 0.8  |
| R3F          | 0.1, 0.5, 1.5             |
| WC           | 0.01, 0.05, 0.1, 0.5      |

E Baselines - detailed

Weight Consolidation: (Kirkpatrick et al., 2017; Chen et al., 2020) Let \( \mathcal{P} \) be the set of all model parameters and \( \mathcal{B} \) be the subset of the bias parameters (affine component in the linear transformations) of the model. For a parameter \( \theta_i \), let \( \theta_i^{\text{pre}} \) be the pre-trained value and \( \theta_i^{\text{fin}} \) be the value during the finetuning process. Then, the regularization loss is.

\[
\mathcal{L}_{WC} = \sum_{i \in \mathcal{P} \setminus \mathcal{B}} ||\theta_i^{\text{fin}} - \theta_i^{\text{pre}}||_2^2
\]

F Detailed Results for non-GLUE datasets

Table 11 shows the results for non-GLUE datasets.

G CAPCORT-MLP - Effect of MLP depth and missing details

Missing Details: We use tanh activation in MLP with learning rate same as the rest of the network. Parameters of MLP are optimized alongside the language model. We use Glorot uniform initializer to initialize the parameters of MLP (Glorot and Bengio, 2010). Bias parameters are initialized to zero.

Table 12 and 13 show that CAPCORT-MLP is resistant to the number of MLP layers chosen. When training with all datapoints, performance is typically within a percentage of each other.

H CAPCORT-I - Which layer to regularize?

Table 14, 15 and 16 compares the result between regularizing the top layer vs regularizing the intermediate layer in CAPCORT-I . We observe that CAPCORT-I consistently outperform when regularizing the intermediate layer.

Table 17, 18, 19 and 20 show the the result for CAPCORT-I with representations chosen from 5th,
Table 11: Performance for our methods as well as baseline methods for non-GLUE datasets. Each reported number is average over five runs. Finetuning runs are filtered if the finetuned model has low performance on data not used for training or for evaluation. Mean/Average Rank-filtered only consider successful runs and discard failed runs as specified in Section 4.1.

Table 12: Performance of CAPCORT-MLP with different number of MLP layers.

### I Supervised Contrastive Learning

Let a mini-batch has $m$ examples, $(x_1, y_1), \ldots, (x_m, y_m)$ and $z_1, \ldots, z_m$ be the representations (output of encoder) using the model being finetuned. Supervised Contrastive Learning encourages the representations of examples of same label in the mini-batch to be close to each other and far from the examples with different label by adding the following loss to the objective:

\[
\mathcal{L}_{SCL} = \sum_{i=1}^{m} \left( -\frac{1}{N_{y_i}} \sum_{j=1}^{m} \left( \frac{1_{i\neq j} \delta_{y_i, y_j}}{\sum_{k=1}^{m} \exp((z_i, z_j)/\tau)} \right) \right)
\]

where $\tau$ is a scalable temperature parameter that controls the separation of classes. Loss function during training is

\[
\mathcal{L} = \lambda \mathcal{L}_{CE} + (1 - \lambda) \mathcal{L}_{SCL}
\]

where $\mathcal{L}_{CE}$ is the cross entropy loss where $\lambda$ is a factor that can be tuned. This was shown to improve finetuning process in (Gunel et al., 2020) for few-shot finetuning.

From the definition of $\mathcal{L}_{SCL}$ we observe that SCL is only effective when the mini-batch size is large and each label class is sufficiently represented in the mini-batch. Otherwise, the loss function
\[ \mathcal{L}_{\text{SCL}} = \sum_{i=1}^{N} \frac{1}{N y_i - 1} \sum_{j=1}^{N} 1_{i \neq j} 1_{y_i = y_j} \log \frac{\exp(\langle z_i, c_{y_j} \rangle / \tau)}{\sum_{k=1}^{N} \exp(\langle z_i, c_k \rangle / \tau)} \]

A naive implementation of this loss function would be very expansive as the centers \( c_1, \ldots, c_\ell \) would change in each iteration. We observe that centers change much slower than the individual examples. This is the reason to replace individual training samples with the centers while computing similarity \( \langle z_i, z_j \rangle \). Thus, we do not update it in each iteration and instead update it only ten times during the finetuning process. Note that it increases the training time by roughly a factor of 10 which is also prohibitive for large datasets. Table 21, 22 and 23 shows the comparison of memory efficient SCL with our methods. We see that both CAPCORT-I and CAPCORT-MLP beat SCL consistently for 250, 500 and 1000 training datapoints. Moreover, SCL incur significant loss for several datasets.

| Tasks     | STD++ | Top | Intermediate |
|-----------|-------|-----|--------------|
| 250 Training Datapoints |       |     |              |
| QNLI      | 75.11 | 76.12 | 78.82 |
| MNLI      | 37.7  | 38.67 | 38.35 |
| AGNEWS    | 88.08 | 87.78 | 88.53 |
| IMDB      | 86.11 | 87.44 | 90.38 |
| SST       | 88.41 | 88.58 | 89.29 |
| COLA      | 41.57 | 44.26 | 43.98 |
| CHEMPROT  | 55.22 | 59.53 | 63.28 |
| MRPC      | 84.43 | 83.91 | 84.65 |
| SCITAIL   | 82.31 | 85.03 | 88.9 |
| SCICITE   | 76.86 | 75.64 | 76.11 |
| RTE       | 59.13 | 61.13 | 60.83 |
| YELP      | 92.51 | 92.23 | 93.13 |
| QQP       | 68.28 | 67.05 | 68.76 |
| Mean      | 71.98 | 72.88 | 74.23 |
| Average Rank | 2.54 | 2.15 | 1.31 |

Table 14: Performance for CONCORT-intermediate vs concort-top.

| Tasks     | STD++ | Top | Intermediate |
|-----------|-------|-----|--------------|
| 500 Training Datapoints |       |     |              |
| QNLI      | 80.86 | 80.7 | 82.56 |
| MNLI      | 41.74 | 41.07 | 49.07 |
| AGNEWS    | 89.09 | 89.34 | 89.53 |
| IMDB      | 83.65 | 89.77 | 91.32 |
| SST       | 89.54 | 89.49 | 89.68 |
| COLA      | 45.29 | 44.22 | 46.95 |
| CHEMPROT  | 65.59 | 61.0 | 73.47 |
| MRPC      | 84.44 | 84.54 | 85.26 |
| SCITAIL   | 85.11 | 88.97 | 90.07 |
| SCICITE   | 78.84 | 79.36 | 79.45 |
| RTE       | 61.01 | 59.39 | 62.45 |
| YELP      | 93.32 | 89.02 | 93.2 |
| QQP       | 61.12 | 69.5 | 70.97 |
| Mean      | 73.81 | 74.34 | 77.23 |
| Average Rank | 2.38 | 2.54 | 1.08 |

Table 15: Performance for CONCORT-intermediate vs concort-top.

We start by considering \( \mathcal{L}_{\text{SCL}} \) over the entire input set instead of mini-batch and then replace the example \( x_j \) with mean of examples of the same class as \( x_j \) while computing similarity with another example. More formally, let the training data be \((x_1, y_1), \ldots, (x_N, y_N)\), the set of labels be \( \{1, \ldots, \ell\} \) and representation of \( x_i \) from the encoder of finetuning model. Let \( C_j = \{i \mid y_i = j\} \) and \( c_j = \frac{1}{|C_j|} \sum_{i \in C_j} z_i \) be the center of embeddings of inputs with label \( j \). We consider the following relaxation of \( \mathcal{L}_{\text{SCL}} \).

\[ \hat{\mathcal{L}}_{\text{SCL}} = \sum_{i=1}^{N} - \frac{1}{N y_i - 1} \sum_{j=1}^{N} 1_{i \neq j} 1_{y_i = y_j} \log \frac{\exp(\langle z_i, c_{y_j} \rangle / \tau)}{\sum_{k=1}^{N} \exp(\langle z_i, c_k \rangle / \tau)} \]

We consider the following relaxation of \( \mathcal{L}_{\text{SCL}} \).
Table 16: Performance for CONCORT-intermediate vs concort-top.

| Tasks  | STD++ | Top   | Intermediate |
|--------|-------|-------|--------------|
|        |       |       |              |
| 1000 Training Datapoints |       |       |              |
| QNLI   | 81.73 | 69.19 | 83.67        |
| MNLI   | 50.32 | 49.84 | 55.5         |
| AGNEWS | 89.29 | 89.45 | 89.74        |
| IMDB   | 90.9  | 72.39 | 91.35        |
| SST    | 90.69 | 90.12 | 90.32        |
| COLA   | 49.39 | 49.22 | 50.50        |
| CHEMPROT | 75.64 | 76.41 | 79.64        |
| MRPC   | 87.82 | 86.93 | 87.54        |
| SCITAIL | 80.85 | 80.25 | 80.21        |
| RTE    | 63.63 | 63.9  | 67.06        |
| YELP   | 94.3  | 84.45 | 94.78        |
| QQP    | 70.58 | 67.22 | 73.04        |
| Mean   | 77.34 | 74.5  | 79.65        |
| Average Rank | 2.08 | 2.62 | 1.31 |

Table 17: Effect of embedding layer to be regularized in CAPCORT-I.

| Tasks | STD++ | 5 | 10 | 20 |
|-------|-------|---|----|----|
|       |       |  |    |    |
| 250 Training datapoints |       |   |    |    |
| QNLI  | 41.57 | 41.23 | 41.94 |
| MRPC  | 84.43 | 83.99 | 84.65 |
| SST   | 88.41 | 88.42 | 89.29 |
| SCITAIL | 82.31 | 89.48 | 89.87 |
| AGNEWS | 88.08 | 88.53 | 88.07 |
| RTE   | 59.13 | 60.83 | 59.78 |
| MNLI  | 37.7  | 38.35 | 39.81 |
| QQP   | 68.28 | 66.66 | 68.76 |
| IMDB  | 86.11 | 88.9  | 90.57 |
| CHEMPROT | 55.22 | 58.0  | 60.53 |
| SCICITE | 76.86 | 76.35 | 78.95 |
| Mean  | 71.98 | 73.28 | 74.21 | 73.82 |

Table 18: Comparison of CAPCORT-I and CAPCORT-MLP against each baseline.

Table 24 show that both CAPCORT-I and CAPCORT-MLP outperform each baseline method in majority of the datasets/

K 250, 500 and 1000 Training Datapoints

Table 25, 26, and 27 show the performance for few-sample finetuning setting.

L Learning rate and Epochs as hyper-parameters

Table 28 discusses the results when we search over optimal learning rate and number of epochs for each task and method. For learning rate, we perform the search over [5e-6,1e-5,2e-5,4e-5] and for epochs, we search over [5,10].

M Results without any filtered runs

Table 29, 30, 31, and 32 shows performance without filtering out failed runs.

N Detailed results for label noise

Table 33, 34, 35, 36 shows detailed results with varying amount of label noise in the training data.

O Representation Collapse - Continual learning perspective

Table 37, 38, 39, 40, 41, 42, 43 shows results for representation collapse when we finetune the model for task A using different methods and then finetune the top layer for task B.

P Measuring representation collapse

Table 44, 45, 46, 47, 48 show the sum of top-k normalized eigenvalues (divide each eigenvalue by the sum of eigenvalues) for k=1, 2, 5, 10, and 20. From this, we can observe that almost all the normalized eigenvalues after the first twenty are close to zero

Table 49, 50 and 51 show the GM-k for k=5, 10 and 20. Table 52, 53, 54 show the HM-k for k=5, 10 and 20. We observe that CAPCORT-I achieves the highest value and thus is most effective in reducing representation collapse.

Q Walltime Analysis

Walltime analysis: STD++ uses a single forward and backward pass with simplest loss function and thus has the least training time. ReInit is a close second as it only differs in the initialization of the model. WC also uses a single forward and backward pass but is slower due to the regularization loss function computation. R3F and DA use two forward passes and two (effective) backward passes. Our method on the other hand use only one forward and backward pass. In addition to that we use only an extra forward pass of the pretrained model. Thus, our method is slower than STD++ , ReInit and WC and is faster than R3F and DA. Table 55 show the training time for all the methods. We observe that R3F consistently takes more time than all the methods. CAPCORT-I runs faster than R3F.
| Tasks  | STD++ | 5   | 10  | 20  |
|--------|--------|-----|-----|-----|
| 500 Training datapoints |
| COLA   | 45.29  | 46.95| 44.58| 44.34|
| QNLI   | 80.86  | 82.56| 81.38| 80.7 |
| MRPC   | 84.44  | 85.26| 84.78| 84.31|
| SST    | 89.54  | 89.68| 89.24| 89.66|
| SCITAIL| 85.11  | 90.36| 90.07| 89.2 |
| YELP   | **93.32** | 92.88| 93.2 | 93.2 |
| AGNEWS | 89.09  | 89.53| 89.19| 88.99|
| RTE    | 61.01  | 62.45| **63.63** | 61.52|
| MNLI   | 41.74  | 44.61| **49.07** | 44.57|
| QQP    | 61.12  | **72.6** | 71.79| 70.97|
| IMDB   | 83.65  | 90.84| 90.31| **91.32** |
| CHEMPROT | 65.59 | 73.34| **73.47** | 71.98|
| SCICITE| 78.84  | 79.39| 78.76| **79.45** |
| Mean   | 73.81  | 76.96| 76.88| 76.17|

Table 18: Effect of embedding layer to be regularized in CAPCORT-I.

and DA but slower than STD++. WC and ReInit.

CAPCORT-MLP runs slower than CAPCORT-I.

### R Connection of GM and HM to parameter estimation error

Let the pseudo linear regression task on finetuned representations be defined by $w \in \mathbb{R}^d$ and the noisy labels observed on $z_i$’s be $y_i = z_i^T w + \epsilon_i$ where $\epsilon_i$’s are the gaussian noise centered around 0 with identity covariance matrix. If $\hat{w}$ is the least square minimizer (same as log-likelihood maximizer), then $\hat{w} = w + N(0, G^{-1})$.

GM corresponds to minimizing the confidence ellipsoid corresponding to the error $\hat{w} - w$. HM corresponds to minimizing the expected $\ell_2^2$ norm of the error vector $\hat{w} - w$. Derivation of $\hat{w}$ and the explanation can be found in Madan et al. (2019).

| Tasks  | STD++ | 5   | 10  | 20  |
|--------|--------|-----|-----|-----|
| 1000 Training Datapoints |
| COLA   | 49.39  | 51.05| 48.16| 47.74|
| QNLI   | 81.73  | 83.11| 83.67| 83.33|
| MRPC   | 87.82  | 87.54| **87.84** | 87.08|
| SST    | 90.69  | **90.75** | 90.32| 90.29|
| SCITAIL| 80.31  | 91.09| 91.51| **91.73** |
| YELP   | 94.3   | **94.78** | 94.39| 94.32|
| AGNEWS | 89.29  | 89.47| 89.74| **89.84** |
| RTE    | 63.63  | **67.06** | 63.1 | 66.28|
| MNLI   | 50.32  | 55.5 | **55.88** | 53.6|
| QQP    | 70.58  | 70.86| 72.21| **73.04** |
| IMDB   | 90.9   | 90.9 | 91.35| **91.47** |
| CHEMPROT | 75.64 | 79.64| 79.66| 78.72|
| SCICITE| **80.85** | 80.83| 80.35| 80.25|
| Mean   | 77.34  | 79.43| 79.09| 79.05|

Table 19: Effect of embedding layer to be regularized in CAPCORT-I.

| Tasks  | STD++ | SCL | CONCORT | CAPCORT |
|--------|--------|-----|---------|---------|
| 250 datapoints |
| QNLI   | 75.11  | 73.79| **78.82** | 76.13|
| SST    | 88.41  | 87.27| **89.29** | 88.59|
| QQP    | 68.28  | 68.79| 68.76| **70.38** |
| SCITAIL| 82.31  | 86.13| **88.9**  | 86.5|
| MNLI   | 37.7   | 38.07| 38.35| **41.18** |
| IMDB   | 86.11  | 90.27| **90.38** | 90.33|
| RTE    | 59.13  | 60.77| **60.83** | 59.3|
| MRPC   | 84.43  | **85.67** | 84.65| 84.08|
| COLA   | 41.57  | 31.91| 43.98| **45.28** |
| CHEMPROT | 55.22 | 32.0 | **63.28** | 62.32|
| Mean   | 67.83  | 65.47| **70.72** | 70.41|
| Average Rank | 4.3 | 3.7 | **1.6** | 2.35|

Table 21: Performance of memory-efficient SCL.
### Table 22: Performance of memory-efficient SCL.

| Tasks      | STD++ | SCL  | CAPCORT-I | CAPCORT-MLP |
|------------|-------|------|-----------|-------------|
| QNLI       | 80.86 | 80.48| **82.56** | 82.43       |
| SST        | 89.54 | **90.25** | 89.68 | 89.4        |
| QQP        | 61.12 | 71.58| 70.97     | **71.67**   |
| SCITAIL    | 85.11 | 87.38| **90.07** | 89.3        |
| MNLI       | 41.74 | 44.81| **49.07** | 44.73       |
| IMDB       | 83.65 | 90.27| **91.32** | 90.49       |
| RTE        | 61.01 | **64.44** | 62.45 | 62.94       |
| MRPC       | 84.44 | 85.34| 85.26     | **85.62**   |
| COLA       | 45.29 | **47.44** | 46.95 | 46.76       |
| CHEMPROT   | 65.59 | 66.55| **73.47** | 72.9        |
| **Mean**   | 69.83 | 72.85| **74.18** | 73.62       |
| **Average Rank** | 4.5 | 2.6 | **1.8** | 2.5 |

### Table 23: Performance of memory-efficient SCL.

| Tasks      | STD++ | SCL  | CAPCORT-I | CAPCORT-MLP |
|------------|-------|------|-----------|-------------|
| QNLI       | 81.73 | 83.3 | **83.67** | 83.25       |
| SST        | 90.69 | **90.77** | 90.32 | 90.62       |
| QQP        | 70.58 | 72.79| **73.04** | 70.67       |
| SCITAIL    | 80.31 | 88.91| **91.51** | 90.9        |
| MNLI       | 50.32 | 55.12| 55.5      | **56.36**   |
| IMDB       | 90.9  | 90.85| **91.35** | 91.19       |
| RTE        | 63.63 | 64.98| 67.06     | 66.7        |
| MRPC       | 87.82 | **88.01** | 87.54 | 87.97       |
| COLA       | 49.39 | 46.47| **51.05** | 49.39       |
| CHEMPROT   | 75.64 | 77.21| **79.64** | 79.03       |
| **Mean**   | 74.1  | 75.84| **77.07** | 76.61       |
| **Average Rank** | 4.3 | 3.1 | **1.9** | 2.75 |

### Table 24: Number of tasks for which CAPCORT-I or CAPCORT-MLP outperform the baseline method.

| GLUE datasets (out of 7) | STD++ | DA | WC | ReInit | R3F |
|--------------------------|-------|----|----|--------|-----|
| STD++                    | 7     | 5  | 5  | 7      | 7   |
| DA                       | 5     | 5  |    |        |     |
| WC                       | 5     |    | 6  |        |     |
| ReInit                   | 7     | 7  |    |        |     |
| R3F                      | 7     |    | 7  |        |     |

| Non-GLUE datasets (out of 6) | STD++ | DA | WC | ReInit | R3F |
|------------------------------|-------|----|----|--------|-----|
| STD++                        | 6     | 6  |    |        |     |
| DA                           | 6     | 6  |    |        |     |
| WC                           | 3     | 5  |    |        |     |
| ReInit                       | 6     | 6  |    |        |     |
| R3F                          | 6     | 6  |    |        |     |

Table 24: Number of tasks for which CAPCORT-I or CAPCORT-MLP outperform the baseline method.
| Tasks        | STD++ | DA  | WC   | ReInit | R3F  | CAPCORT-I | CAPCORT-MLP |
|--------------|-------|-----|------|--------|------|------------|--------------|
| **250 Training datapoints** |
| QQP          | 68.28 | 65.63 | 69.01 | 68.46 | 67.45 | 68.76 | **70.38**   |
| COLA         | 44.57 | 40.39 | 44.14 | 43.73 | 40.76 | 43.98 | **45.28**   |
| RTE          | 61.73 | 58.99 | 60.29 | 60.02 | 60.83 | 59.3  |             |
| MNLI         | 37.7 | 40.24 | 33.71 | **41.47** | 37.8 | 38.35 | 41.18       |
| YELP         | 92.51 | 92.68 | 92.97 | **93.64** | 93.13 | 93.13 | 93.3        |
| CHEMPROT     | 55.22 | 58.6  | 59.52 | **64.92** | 59.6 | 63.28 | 62.32       |
| QNLI         | 75.11 | 78.72 | 75.27 | 77.62 | 77.43 | **78.82** | 76.13       |
| IMDB         | 86.11 | 89.17 | 89.51 | 89.44 | 89.09 | **90.38** | 90.33       |
| SCICITE      | 76.86 | 77.2  | 76.23 | **78.86** | 76.11 | 75.76 |             |
| SST          | 88.41 | 88.1  | 88.14 | 88.3  | 88.47 | **89.29** | 88.59       |
| MRPC         | 84.43 | 83.81 | **84.93** | 84.87 | 84.71 | 84.65 | 84.08       |
| SCITAIL      | 82.31 | 87.65 | 86.2  | 87.8  | 88.6  | **88.9**  | 86.5        |
| AGNEWS       | 88.08 | 87.51 | 87.55 | 87.36 | 87.99 | **88.53** | 88.13       |
| **Mean**     | 71.98 | 73.16 | 72.86 | **74.37** | 73.27 | 74.23 | 73.94       |
| **Average Rank** | 5.62 | 4.92 | 4.62 | 3.0 | 4.04 | **2.5** | 3.31 |

Table 25: Performance for different regularization methods.

| Tasks        | STD++ | DA  | WC   | ReInit | R3F  | CAPCORT-I | CAPCORT-MLP |
|--------------|-------|-----|------|--------|------|------------|--------------|
| **500 Training datapoints** |
| QQP          | 61.12 | 71.5 | 70.13 | **72.03** | 71.61 | 70.97 | 71.67       |
| COLA         | 45.29 | 46.27 | **47.34** | 46.03 | 45.36 | 46.95 | 46.76       |
| RTE          | 61.01 | 60.53 | 61.23 | **63.33** | 60.41 | 62.45 | 62.94       |
| MNLI         | 41.74 | 49.05 | 46.32 | **51.11** | 42.25 | 49.07 | 44.73       |
| YELP         | 93.32 | **93.46** | 92.87 | 93.04 | 93.27 | 93.2  | 93.35       |
| CHEMPROT     | 65.59 | **74.2** | 70.78 | 73.2  | 70.04 | 73.47 | 72.9        |
| QNLI         | 80.86 | 82.4  | 82.19 | 81.84 | 82.42 | **82.56** | 82.43       |
| IMDB         | 83.65 | 90.42 | **91.47** | 90.57 | 89.45 | 91.32 | 90.49       |
| SCICITE      | 78.84 | 79.19 | 79.21 | 79.14 | 78.38 | 79.45 | **79.8**    |
| SST          | 89.54 | 89.36 | 90.02 | **90.37** | 90.1  | 89.68 | 89.4        |
| MRPC         | 84.44 | 84.69 | 85.4  | 84.93 | 84.77 | 85.26 | **85.62**   |
| SCITAIL      | 85.11 | 87.41 | **90.23** | 89.38 | 80.73 | 90.07 | 89.3        |
| AGNEWS       | 89.09 | 88.93 | 89.0  | 89.09 | **89.59** | 89.53 | 89.54       |
| **Mean**     | 73.81 | 76.72 | 76.63 | **77.24** | 75.26 | 77.23 | 76.84       |
| **Average Rank** | 6.08 | 4.38 | 3.69 | 3.31 | 4.85 | **2.77** | 2.92 |

Table 26: Performance for different regularization methods.
## Table 27: Performance for different regularization methods.

| Tasks   | STD++   | DA     | WC     | ReInit  | R3F     | CAPCORT-I | CAPCORT-MLP |
|---------|---------|--------|--------|---------|---------|-----------|-------------|
| QQP     | 70.58   | 71.6   | 70.64  | 71.87   | 71.69   | **73.04** | 70.67       |
| COLA    | 49.39   | 50.46  | 50.79  | 49.47   | 50.01   | **51.05** | 49.39       |
| RTE     | 63.63   | 65.7   | 62.09  | 65.78   | 63.54   | **67.06** | 66.7        |
| MNLI    | 50.32   | 56.1   | 55.28  | **57.9**| 55.51   | 55.5      | 56.36       |
| YELP    | 94.3    | 93.83  | 94.41  | 93.99   | 94.51   | **94.78** | 94.19       |
| CHEMPROT| 75.64   | 78.88  | 78.37  | 78.12   | 77.98   | **79.64** | 79.03       |
| QNLI    | 81.73   | 83.46  | 83.51  | **84.14**| 83.11   | 83.67     | 83.25       |
| IMDB    | 90.9    | 91.22  | 90.99  | 91.29   | 90.8    | **91.35** | 91.19       |
| SCICITE | 80.85   | 80.93  | **81.55**| 80.39   | 80.15   | 80.25     | 81.09       |
| SST     | 90.69   | 90.8   | 90.81  | 90.67   | 90.5    | 90.32     | 90.62       |
| MRPC    | 87.82   | 87.49  | **88.12**| 87.29   | 87.92   | 87.54     | 87.97       |
| SCITAIL | 80.31   | 89.72  | **91.68**| 91.0    | 91.38   | 91.51     | 90.9        |
| AGNEWS  | 89.29   | 89.38  | 89.25  | **89.91**| 89.62   | 89.74     | 89.32       |
| Mean    | 77.34   | 79.2   | 79.04  | 79.37   | 78.98   | **79.65** | 79.28       |
| Average Rank | 5.69 | 4.0 | 3.62 | 3.54 | 4.62 | **2.69** | 3.85 |

## Table 28: Results with HPO over epochs and learning rate

| Tasks   | STD++   | DA     | WC     | ReInit  | R3F     | CAPCORT-I | CAPCORT-MLP |
|---------|---------|--------|--------|---------|---------|-----------|-------------|
| qqp-10k | 78.58   | 78.79  | 78.20  | 78.74   | 78.67   | **79.71** | 79.23       |
| rte     | 72.92   | 74.45  | 74.30  | 74.11   | 74.20   | **75.09** | 74.95       |
| cola    | 56.67   | 62.36  | 61.79  | 61.50   | 61.74   | **62.45** | 61.82       |
| mrpc    | 90.30   | 90.50  | 90.44  | 90.79   | 90.41   | 90.79     | **90.83**   |

## Table 29: Performance for different regularization methods without filtering failed runs.

| Tasks   | STD++   | DA     | WC     | ReInit  | R3F     | CAPCORT-I | CAPCORT-MLP |
|---------|---------|--------|--------|---------|---------|-----------|-------------|
| YELP    | 92.51   | 92.68  | 92.97  | 93.64   | 93.13   | 93.13     | 93.57       |
| RTE     | 55.6    | 58.12  | 55.96  | **60.29**| 58.88   | 59.78     | 60.14       |
| MNLI    | 25.75   | 30.22  | 28.42  | **41.47**| 31.94   | 39.2      | 38.58       |
| QNLI    | 72.71   | 77.5   | 75.27  | 77.62   | 77.43   | **78.82** | 76.13       |
| SCITAIL | 82.31   | 87.65  | 86.2   | 87.8    | 88.6    | **88.9**  | 86.5        |
| SST     | 88.41   | 88.1   | 88.14  | 88.3    | 88.47   | **89.29** | 88.59       |
| CHEMPROT| 55.22   | 58.6   | 59.52  | **64.92**| 59.6    | 60.72     | 62.32       |
| AGNEWS  | 88.08   | 87.51  | 87.55  | 87.36   | 87.99   | **88.53** | 88.13       |
| SCICITE | 76.86   | 77.2   | 76.23  | 78.86   | 77.39   | **78.95** | 75.76       |
| IMDB    | 83.5    | 80.13  | 89.51  | 89.44   | 89.05   | **90.38** | 90.33       |
| COLA    | 41.57   | 40.39  | 45.14  | 43.73   | 40.76   | 43.98     | **45.28**   |
| MRPC    | 84.43   | 83.81  | **84.93**| 84.87   | 84.71   | 84.65     | 84.08       |
| QQP     | 35.94   | 64.08  | 51.92  | 68.46   | 55.81   | **68.76** | 65.5        |
| Mean    | 67.91   | 71.23  | 70.9   | **74.37**| 71.83   | 74.24     | 73.45       |
| Average Rank | 5.92 | 5.46 | 4.85 | 2.69 | 3.88 | **1.96** | 3.23 |
| Tasks    | STD++ | DA  | WC   | ReInit | R3F  | CAPCORT-I | CAPCORT-MLP |
|----------|-------|-----|------|--------|------|------------|--------------|
| **500 Training Datapoints** |       |     |      |        |      |            |              |
| YELP     | 85.57 | **93.46** | 92.87 | 93.04 | 93.27 | 93.2       | 93.35        |
| RTE      | 52.47 | 53.86 | 59.81 | 60.87 | 55.64 | **61.52**  | 58.3         |
| MNLI     | 23.87 | 32.25 | 39.53 | **51.11** | 42.19 | 43.81      | 43.19        |
| QNLI     | 74.76 | 75.45 | 80.64 | **81.84** | 81.76 | 81.38      | 81.25        |
| SCITAIL  | 85.11 | 87.41 | **90.23** | 89.38 | 80.73 | 90.07      | 89.3         |
| CHEMPROT | 65.59 | **74.2** | 70.78 | 73.2  | 70.04 | 73.47      | 72.9         |
| SST      | 89.54 | 89.36 | 90.02 | **90.37** | 90.1  | 89.68      | 89.4         |
| AGNEWS   | 89.09 | 88.93 | 89.0  | 89.09 | **89.59** | 89.53      | 89.54        |
| SCICITE  | 78.84 | 79.19 | 79.21 | 79.14 | 78.38 | 79.45      | **79.8**     |
| IMDB     | 78.68 | 90.42 | **91.47** | 90.57 | 89.45 | 91.32      | 90.49        |
| COLA     | 45.29 | 46.27 | **47.34** | 46.03 | 45.36 | 46.95      | 46.76        |
| MRPC     | 84.44 | 84.69 | 85.4  | 84.93 | 84.77 | 85.26      | **85.62**    |
| QQP      | 25.73 | 56.3  | 68.63 | **72.03** | 42.07 | 70.97      | 71.48        |
| Mean     | 67.61 | 73.21 | 75.76 | **77.05** | 72.57 | 76.66      | 76.26        |
| Average Rank | 6.54   | 4.85  | 3.46  | 2.92  | 4.62  | **2.54**   | 3.08         |

Table 30: Performance for different regularization methods without filtering failed runs.

| Tasks    | STD++ | DA  | WC   | ReInit | R3F  | CAPCORT-I | CAPCORT-MLP |
|----------|-------|-----|------|--------|------|------------|--------------|
| **1000 Training Datapoints** |       |     |      |        |      |            |              |
| YELP     | 94.3  | 93.83 | 94.24 | 93.99 | 94.51 | **94.78**  | 94.19        |
| RTE      | 49.74 | 53.26 | 59.09 | 62.94 | 52.11 | **65.78**  | 57.16        |
| MNLI     | 32.07 | 31.16 | 53.25 | **57.38** | 45.06 | 55.88      | 54.73        |
| QNLI     | 69.93 | 82.7  | 82.34 | **84.14** | 77.63 | 83.67      | 83.23        |
| SCITAIL  | 80.31 | 89.72 | **91.68** | 91.0  | 91.38 | 91.51      | 90.9         |
| SST      | 90.69 | 90.8  | **90.81** | 90.67 | 90.5  | 90.75      | 90.62        |
| CHEMPROT | 75.64 | 78.88 | 78.37 | 78.12 | 77.98 | **79.64**  | 79.03        |
| AGNEWS   | 89.29 | 89.38 | 89.25 | **89.91** | 89.62 | 89.74      | 89.32        |
| SCICITE  | 80.85 | 80.93 | **81.55** | 80.39 | 80.82 | 80.25      | 81.09        |
| IMDB     | 82.3  | 91.22 | 90.99 | 91.29 | 90.8  | **91.35**  | 91.19        |
| COLA     | 49.39 | 50.46 | 50.79 | 49.47 | 50.01 | **51.05**  | 49.39        |
| MRPC     | 86.84 | 86.27 | **88.12** | 87.29 | 87.75 | 87.54      | 87.97        |
| QQP      | 26.0  | 56.63 | 70.26 | 70.36 | 38.79 | **73.04**  | 72.16        |
| Mean     | 69.8  | 75.02 | 78.52 | 79.0  | 74.38 | **79.61**  | 78.54        |
| Average Rank | 6.0    | 4.54  | 3.23  | 3.54  | 4.77  | **2.15**   | 3.77         |

Table 31: Performance for different regularization methods without filtering failed runs.
| Tasks  | STD++  | DA    | WC    | ReInit | R3F   | CAPCORT-I | CAPCORT-MLP |
|--------|--------|-------|-------|--------|-------|-----------|-------------|
| MRPC   | 86.84  | 90.67 | 88.6  | 90.98  | 89.9  | **91.49** | 91.12       |
| IMDB-10k | 66.59  | 93.24 | 93.69 | 92.7   | 93.1  | **93.96** | 93.87       |
| YELP-10k | 72.65  | 95.55 | 95.81 | 95.56  | 95.42 | 95.78     | **95.96**   |
| SCICITE | 81.87  | 82.13 | 82.23 | 82.41  | 81.97 | 82.74     | **83.15**   |
| QNLI-10k | 64.93  | 81.64 | 86.1  | 86.73  | 78.43 | 86.85     | **87.36**   |
| CHEMPROT | 72.59  | 82.57 | **83.91** | 82.46 | 82.73 | 83.49     | 83.67       |
| MNLI-10k | 14.37  | 45.98 | 55.94 | 46.54  | 21.66 | **65.48** | 64.81       |
| COLA   | 59.69  | 63.45 | 61.5  | 61.25  | 62.04 | 62.34     | 62.47       |
| RTE    | 51.35  | 56.14 | 52.87 | 66.86  | 56.68 | **71.26** | 61.44       |
| AGNEWS-10k | 91.67  | 91.82 | 91.92 | 91.67  | 91.73 | **92.07** | 91.93       |
| SST    | 81.95  | 84.13 | 92.32 | 92.28  | 83.88 | 92.71     | **93.23**   |
| QQP-10k | 5.9    | 47.77 | 76.25 | 55.16  | 27.8  | 79.03     | **79.3**    |
| SCITAIL-10k | 76.01  | 93.74 | 94.03 | 93.36  | 86.54 | 93.74     | **94.75**   |
| Mean   | 63.57  | 77.6  | 81.17 | 79.84  | 73.22 | **83.92** | 83.31       |
| Average Rank | 6.92 | 4.38  | 3.46  | 4.38   | 5.31  | 1.92      | **1.62**    |

Table 32: Performance for different regularization methods without filtering failed runs.

| Tasks  | STD++  | DA    | WC    | ReInit | R3F   | CAPCORT-I | CAPCORT-MLP |
|--------|--------|-------|-------|--------|-------|-----------|-------------|
| SCICITE | 81.29  | 56.46 | 81.34 | 81.11  | 81.03 | 81.36     | **81.88**   |
| QNLI-10k | 57.35  | 50.18 | 68.59 | 67.16  | 64.33 | **83.87** | **86.08**   |
| MRPC   | 87.4   | 87.48 | 86.66 | **89.11** | 88.41 | 88.41     | 86.81       |
| RTE    | 51.35  | 48.59 | 51.55 | 65.63  | 48.65 | **67.58** | 61.16       |
| IMDB-10k | 72.43  | 68.28 | 77.4  | 62.14  | 91.63 | **92.84** | 92.56       |
| CHEMPROT | 71.57  | 81.42 | **83.88** | 81.81 | 81.48 | 81.64     | 82.18       |
| AGNEWS-10k | 91.1   | 91.11 | **91.55** | 91.07 | 91.26 | 91.21     | 91.47       |
| YELP-10k | 49.92  | 72.44 | 86.24 | 85.4   | 72.38 | 95.1      | **95.36**   |
| MNLI-10k | 0.0    | 0.0   | 31.9  | 41.99  | 47.51 | **63.26** | 24.96       |
| SST-10k | 91.12  | 83.17 | **91.83** | 90.05 | 91.31 | 83.21     | 90.73       |
| SCITAIL-10k | 58.21  | 57.94 | 92.91 | 83.36  | 83.93 | 92.32     | **93.48**   |
| QQP-10k | 0.0    | 0.0   | 76.24 | 57.38  | 0.0   | 77.78     | **78.82**   |
| COLA   | 46.38  | 47.84 | 59.33 | 43.47  | 45.15 | **59.53** | 48.29       |
| Mean   | 58.32  | 57.3  | 75.34 | 72.28  | 68.24 | **81.39** | 77.98       |
| Average Rank | 5.5  | 5.96  | 2.92  | 4.38   | 4.35  | **2.42**  | 2.46        |

Table 33: Training with at most 10k training datapoints on 13 datasets.
| Tasks       | STD++ | DA   | WC   | ReInit | R3F   | CAPCORT-I | CAPCORT-MLP |
|-------------|-------|------|------|--------|-------|-----------|-------------|
| SST-10k     | 88.0  | 66.93| **91.58** | 88.97 | 81.02 | 82.73     | 82.64       |
| MNLI-10k    | 12.44 | 0.13 | 57.55| 48.92  | 24.31 | **59.67** | 12.38       |
| SCITAIL-10k | 65.86 | 63.01| 74.68| 80.92  | 69.86 | 90.97     | **92.82**   |
| IMDB-10k    | 33.33 | 33.33| 44.86| 67.47  | 76.92 | 90.96     | **91.88**   |
| RTE         | 50.54 | 49.46| 48.38| 61.52  | 50.54 | **64.77** | 60.58       |
| MRPC        | 84.11 | 83.69| 85.11| **88.49** | 84.33 | 86.06     | 87.4        |
| AGNEWS-10k | 90.19 | 90.24| 90.54| 90.23  | 90.32 | 90.51     | **90.58**   |
| CHEMPROT    | 80.75 | 68.56| 82.25| 80.56  | 69.38 | **82.49** | 71.92       |
| QQP-10k     | 0.0   | 0.0  | 75.49| 14.99  | 14.92 | 76.17     | **76.76**   |
| YELP-10k    | 60.78 | 60.77| 83.65| 76.4   | 73.85 | 94.22     | **94.64**   |
| COLA        | 56.75 | 45.85| **58.65** | 52.8  | 45.43 | 44.12     | 55.6        |
| QNLI-10k    | 50.32 | 50.54| 65.61| 63.15  | 66.69 | 74.24     | **83.83**   |
| SCICITE     | 80.79 | 69.21| 80.45| 80.71  | 80.29 | 78.76     | **81.52**   |

Mean 57.99 52.44 72.22 68.86 63.68 **78.13** 75.58
Average Rank 4.96 6.46 3.23 3.46 4.73 2.77 **2.38**

Table 34: Training with at most 10k training datapoints on 13 datasets.

| Tasks       | STD++ | DA   | WC   | ReInit | R3F   | CAPCORT-I | CAPCORT-MLP |
|-------------|-------|------|------|--------|-------|-----------|-------------|
| YELP-10k    | 49.91 | 49.64| **94.1** | 56.72 | 50.14 | 92.83     | 93.63       |
| RTE         | 48.65 | 47.29| 50.54| 55.38  | 47.29 | **64.98** | 58.84       |
| QQP-10k     | 0.07  | 0.0  | 19.12| 0.0    | 0.0   | 72.29     | **73.52**   |
| CHEMPROT    | 70.05 | 70.33| **81.22** | 79.75 | 80.03 | 80.44     | 69.08       |
| IMDB-10k    | 33.33 | 33.33| 45.03| 33.39  | 33.33 | 77.62     | **89.3**    |
| AGNEWS-10k | 85.77 | 86.9 | **87.54** | 87.52 | 71.6  | 86.6      | 86.8        |
| SST-10k     | **87.73** | 58.51| 79.91| 86.51  | 79.91 | 87.61     | 73.26       |
| MNLI-10k    | 0.0   | 0.0  | **48.74** | 10.46 | 0.0   | 0.0       | 0.0         |
| QNLI-10k    | 56.13 | 50.27| 69.56| 49.89  | 56.04 | 61.93     | **81.28**   |
| MRPC        | 84.45 | 82.25| **85.62** | 83.49 | 82.37 | 72.71     | 84.88       |
| SCITAIL-10k | 49.62 | 49.62| 74.85| 49.62  | 49.62 | 69.5      | **88.91**   |
| COLA        | 17.64 | 9.53 | 20.23| **47.09** | 19.7  | 20.16     | 35.51       |
| SCICITE     | 45.74 | 38.42| 77.13| **77.57** | 68.46 | 74.41     | 76.89       |

Mean 48.39 44.31 64.12 55.18 49.11 66.24 **70.15**
Average Rank 4.88 5.92 **2.19** 3.58 5.27 3.31 2.85

Table 35: Training with at most 10k training datapoints on 13 datasets.

24
| Tasks       | STD++ | DA | WC | ReInit | R3F | CAPCORT-I | CAPCORT-MLP |
|------------|--------|----|----|--------|-----|------------|-------------|
| QQP-10k    | 0.0    | 0.0| **68.62** | 13.08 | 16.01 | 68.31      | 67.36       |
| SCICITE    | 34.18  | 24.67 | 43.99 | **73.7** | 62.73 | 67.3       | **73.7**    |
| COLA       | 3.42   | 6.67 | **35.97** | 22.14 | 12.8  | 15.01      | 24.99       |
| IMDB-10k   | 33.33  | 33.33 | **47.15** | 43.14 | 33.33 | 33.33      | 35.6        |
| MNLI-10k   | 0.0    | 0.0 | **1.29** | 0.0   | 0.0   | 0.22       | 0.0         |
| CHEMPROT   | 41.96  | 54.1 | 69.21 | **75.81** | 45.41 | 67.7       | 53.96       |
| QNLI-10k   | 50.11  | 49.89 | 56.27 | 55.13 | 50.27 | 50.18      | **77.96**   |
| AGNEWS-10k| 70.31  | **83.14** | 67.91 | 81.75 | 82.41 | 45.87      | 68.06       |
| RTE        | 49.46  | 49.46 | 51.48 | 53.14 | 49.46 | **60.83**  | 54.51       |
| YELP-10k   | 50.02  | 49.56 | 56.02 | 60.22 | 49.79 | 51.99      | **91.44**   |
| SCITAIL-10k| 49.62  | 49.62 | 58.45 | 49.62 | 49.62 | 56.83      | **86.71**   |
| MRPC       | 80.92  | **81.38** | 81.26 | 78.42 | 81.22 | 79.54      | 80.63       |
| SST-10k    | 58.23  | 64.05 | 57.73 | **75.53** | 63.03 | 57.08      | 57.98       |
| Mean       | 40.12  | 41.99 | 53.49 | 52.44 | 45.85 | 50.32      | **59.45**   |
| Average Rank | 5.5   | 4.88 | **2.77** | 3.23 | 4.54 | 4.04       | 3.04        |

Table 36: Training with at most 10k training datapoints on 13 datasets.

| Tasks       | STD++ | DA | WC | ReInit | R3F | CAPCORT-I | CAPCORT-MLP |
|------------|--------|----|----|--------|-----|------------|-------------|
| COLA       | -0.38  | -0.63 | 3.39 | -1.17 | 0.0  | 0.93       | **5.08**    |
| MRPC       | 81.62  | 81.22 | **83.13** | 81.57 | 81.44 | 82.12      | 82.42       |
| QQP-10k    | 29.68  | 28.97 | 43.13 | 31.71 | 22.18 | 46.74      | **59.26**   |
| YELP-10k   | 50.86  | 51.55 | 55.13 | 51.46 | 50.99 | 52.63      | **60.04**   |
| SCITAIL-10k| 59.55  | 58.44 | **74.46** | 61.32 | 49.62 | 67.38      | 71.24       |
| SCICITE    | 24.82  | 24.84 | **28.74** | 24.75 | 24.67 | 25.02      | 27.71       |
| AGNEWS-10k| 20.22  | 20.5  | 36.61 | 16.92 | 20.15 | 31.01      | **62.9**    |
| IMDB-10k   | 41.36  | 33.33 | 49.87 | 41.0  | 43.36 | 43.57      | **55.72**   |
| CHEMPROT   | 33.1   | 32.46 | **33.23** | 32.92 | 32.34 | 33.09      | 33.15       |
| MNLI-10k   | 8.07   | 8.27  | 16.76 | 8.36  | 6.92  | 14.58      | **17.69**   |
| SST-10k    | 51.25  | 54.55 | 58.21 | 52.24 | 52.2  | 55.93      | **66.36**   |
| RTE        | 51.32  | 51.62 | 51.5  | 51.26 | **53.55** | 47.83      | 52.17       |
| Mean       | 37.62  | 37.09 | 44.51 | 37.69 | 36.45 | 41.74      | **49.48**   |
| Average Rank | 5.17  | 5.17 | 1.92  | 5.33  | 5.67  | 3.33       | **1.42**    |

Table 37: Results for training top layer for different task after finetuning entire model for QNLI-10k
| Tasks        | STD++ | DA    | WC    | ReInit | R3F    | CAPCORT-I | CAPCORT-MLP |
|--------------|-------|-------|-------|--------|--------|-----------|-------------|
| MRPC         | 81.45 | 81.4  | 82.67 | 81.18  | 81.22  | 82.13     | 81.15       |
| CHEM PROT    | 33.46 | 33.46 | 33.46 | 33.46  | 33.46  | 33.59   | 33.46       |
| QNLI-10k     | 59.06 | 63.74 | 67.98 | 61.65  | 50.54  | 64.95     | 64.58       |
| YELP-10k     | 50.18 | 50.02 | 55.91 | 51.43  | 50.06  | 64.07     | 51.32       |
| SCITAIL-10k  | 62.54 | 72.01 | 68.1  | 65.55  | 49.62  | 76.55     | 71.55       |
| COLA         | 0.23  | -0.79 | 0.0   | -0.01  | 0.0    | 5.17      | -0.82       |
| AGNEWS-10k  | 16.02 | 19.77 | 32.56 | 19.6   | 10.76  | 64.32     | 38.5        |
| IMDB-10k     | 39.92 | 45.49 | 45.45 | 39.17  | 33.33  | 66.28     | 34.55       |
| SCICITE      | 24.67 | 24.67 | 24.65 | 24.67  | 24.67  | 28.51     | 24.67       |
| MNLI-10k     | 11.73 | 17.18 | 18.91 | 15.72  | 0.0    | 19.57     | 19.38       |
| SST-10k      | 49.69 | nan   | 52.24 | 50.89  | 51.03  | 69.87     | 58.08       |
| RTE          | 48.86 | 52.71 | 52.89 | 51.44  | 50.9   | 53.52     | 52.17       |
| Mean         | 39.82 | 41.79 | 44.57 | 41.23  | 36.3   | **52.38** | 44.05       |
| Average Rank | 4.96  | 3.79  | 3.08  | 4.79   | 5.67   | **1.17**  | 4.04        |

Table 38: Results for training top layer for different task after finetuning entire model for QQP-10k

| Tasks        | STD++ | DA    | WC    | ReInit | R3F    | CAPCORT-I | CAPCORT-MLP |
|--------------|-------|-------|-------|--------|--------|-----------|-------------|
| RTE          | 51.35 | 56.82 | 52.08 | 67.33  | 56.82  | **68.95** | 64.44       |
| CHEM PROT    | 33.46 | **33.49** | 33.31 | 33.18  | 33.2   | 33.33     | 33.21       |
| QNP-10k      | 0.0   | 14.81 | 15.45 | 59.92  | 24.49  | **60.55** | 48.03       |
| QNLI-10k     | 50.54 | 51.22 | 52.57 | 53.88  | 52.06  | 52.6      | **54.51**   |
| SCITAIL-10k  | 49.62 | 56.42 | 57.32 | 78.47  | 68.79  | **78.58** | 71.15       |
| MRPC         | **81.22** | 80.14 | 81.16 | 81.08  | 80.3   | 80.12     | 79.73       |
| AGNEWS-10k  | 10.0  | 18.45 | 19.69 | **43.77** | 21.62 | 34.52     | 28.13       |
| IMDB-10k     | 33.33 | 42.36 | 41.43 | **57.45** | 43.13 | 54.98     | 51.87       |
| SCICITE      | 24.67 | 26.03 | 26.02 | 25.69  | 24.65  | 26.63     | **29.63**   |
| SST-10k      | 50.92 | 55.09 | 61.81 | **66.49** | 55.64 | 62.27     | 59.63       |
| YELP-10k     | 50.02 | 52.67 | 54.75 | **62.68** | 54.7  | 58.66     | 61.74       |
| COLA         | 0.0   | 0.0   | 0.0   | 0.0    | 0.0    | **0.51**  | -0.69       |
| Mean         | 36.26 | 40.63 | 41.3  | **52.49** | 42.95 | 50.98     | 48.45       |
| Average Rank | 5.75  | 4.88  | 4.33  | 2.58   | 4.71   | **2.25**  | 3.5         |

Table 39: Results for training top layer for different task after finetuning entire model for MNLI-10k

| Tasks        | STD++ | DA    | WC    | ReInit | R3F    | CAPCORT-I | CAPCORT-MLP |
|--------------|-------|-------|-------|--------|--------|-----------|-------------|
| SCICITE      | 33.28 | 33.85 | 36.38 | 29.25  | 31.38  | **37.88** | 33.58       |
| COLA         | 1.1   | 5.61  | 4.66  | 3.01   | 3.59   | 8.11      | **15.16**   |
| QNLI-10k     | 53.98 | 56.17 | 57.89 | 59.45  | 54.23  | 58.6      | **61.1**    |
| YELP-10k     | 57.78 | 58.5  | 59.56 | 58.38  | 55.79  | **61.11** | 60.11       |
| SCITAIL-10k  | 50.3  | 56.81 | 51.76 | 57.21  | 53.59  | 55.23     | **67.82**   |
| RTE          | 52.67 | 52.64 | 52.6  | **54.51** | 54.24 | 53.13     | 54.39       |
| IMDB-10k     | 57.55 | 60.21 | 59.33 | 57.61  | 56.16  | **60.9**  | 59.42       |
| CHEM PROT    | 33.69 | 33.65 | 33.44 | 33.17  | 33.47  | **34.32** | 33.54       |
| SST-10k      | 64.64 | 64.43 | 64.99 | 63.9   | 64.46  | 66.34     | **66.79**   |
| MRPC         | 81.01 | 81.17 | 81.21 | **81.72** | 81.11 | 81.3      | 81.04       |
| MNLI-10k     | 5.46  | 5.79  | 6.14  | **10.54** | 6.05  | 7.2       | 9.64        |
| QQP-10k      | 1.95  | 1.19  | 0.05  | 11.24  | 3.2    | 0.51      | **31.22**   |
| Mean         | 41.12 | 42.5  | 42.34 | 43.33  | 41.44  | 43.72     | **47.82**   |
| Average Rank | 5.58  | 4.17  | 4.42  | 3.83   | 5.17   | 2.5       | **2.33**    |

Table 40: Results for training top layer for different task after finetuning entire model for AGNEWS-10k
| Tasks       | STD++ | DA    | WC    | ReInit | R3F    | CAPCORT-I | CAPCORT-MLP |
|-------------|-------|-------|-------|--------|--------|-----------|-------------|
| CHEMPROT   | nan   | 31.23 | 32.66 | 32.14  | 32.55  | **33.13** | 32.94       |
| COLA       | 1.28  | -0.71 | 4.41  | 0.0    | 0.0    | 1.93      | **7.01**    |
| QQP-10k    | 10.1  | 18.43 | 1.98  | 16.43  | 22.99  | 5.02      | 21.81       |
| QNLI-10k   | 52.3  | 54.59 | **59.36** | 52.59  | 49.81  | 55.78     | 54.57       |
| YELP-10k   | 91.28 | 91.96 | 78.81 | 83.41  | 91.41  | 93.03     | **93.44**   |
| SCITAIL-10k| 51.76 | 69.71 | 61.04 | 49.56  | 51.46  | **73.47** | 50.77       |
| RTE        | 53.61 | **59.21** | 50.54 | 54.3   | 53.79  | 55.6      | 57.4        |
| AGNEWS-10k | 31.43 | 37.43 | 40.37 | 23.64  | 29.79  | 44.51     | **49.81**   |
| SCICITE    | **32.46** | 25.24 | 25.13 | 24.67  | 24.67  | 29.07     | 31.67       |
| MNLI-10k   | 6.78  | 9.17  | 7.88  | 7.1    | 7.89   | **12.17** | 12.16       |
| SST-10k    | 85.44 | 87.16 | 76.41 | 80.0   | 87.96  | **89.39** | 89.11       |
| MRPC       | **81.22** | nan   | 80.56 | 80.64  | 81.17  | 81.17     | 80.65       |
| Mean       | 45.24 | 43.95 | 43.26 | 42.04  | 44.46  | 47.86     | **48.44**   |
| Average Rank | 4.17  | 3.42  | 4.67  | 5.58   | 4.5    | **2.25**  | 2.42        |

Table 41: Results for training top layer for different task after finetuning entire model for IMDB-10k

| Tasks       | STD++ | DA    | WC    | ReInit | R3F    | CAPCORT-I | CAPCORT-MLP |
|-------------|-------|-------|-------|--------|--------|-----------|-------------|
| COLA       | 4.89  | 4.86  | 9.5   | 1.09   | -0.44  | **12.78** | 12.6        |
| CHEMPROT   | 32.29 | 32.09 | 31.95 | 32.17  | **32.98** | 31.7     | 31.41       |
| QQP-10k    | 0.49  | 9.97  | 0.03  | 6.07   | 2.12   | 0.0       | 0.26        |
| QNLI-10k   | 57.18 | 58.82 | 60.03 | 56.73  | 53.81  | **62.24** | 60.5        |
| YELP-10k   | 62.97 | 65.96 | 63.1  | 62.67  | 57.9   | 67.26     | **76.47**   |
| SCITAIL-10k| 49.37 | 49.31 | 57.88 | 49.46  | 49.62  | **66.74** | 49.9        |
| RTE        | **54.66** | 53.07 | 52.49 | 50.76  | 51.08  | 54.01     | 53.88       |
| AGNEWS-10k | 48.5  | 49.79 | 55.09 | 38.46  | 34.05  | 57.98     | **60.96**   |
| IMDB-10k   | 62.83 | 61.85 | 65.36 | 53.94  | 48.51  | 65.12     | **70.33**   |
| MRPC       | 81.12 | 81.08 | 81.37 | 81.12  | 81.12  | 81.02     | 81.28       |
| MNLI-10k   | **9.36** | 7.45  | 8.05  | 8.27   | 4.1    | 8.21      | 7.32        |
| SST-10k    | 63.3  | 62.58 | 70.41 | 55.46  | 54.3   | 72.05     | **72.78**   |
| Mean       | 43.91 | 44.74 | 46.27 | 41.35  | 39.09  | **48.26** | 48.14       |
| Average Rank | 3.75  | 4.42  | 3.42  | 5.0    | 5.5    | 3.08      | **2.83**    |

Table 42: Results for training top layer for different task after finetuning entire model for SCICITE

| Tasks       | STD++ | DA    | WC    | ReInit | R3F    | CAPCORT-I | CAPCORT-MLP |
|-------------|-------|-------|-------|--------|--------|-----------|-------------|
| COLA       | 0.0   | 0.0   | 1.48  | -0.59  | 0.0    | **5.06**  | 1.91        |
| MRPC       | 81.22 | 81.22 | 81.31 | 80.75  | 81.22  | 81.68     | **81.91**   |
| QQP-10k    | 0.0   | 0.0   | 18.89 | 60.51  | 0.0    | **60.87** | 36.29       |
| QNLI-10k   | 50.54 | 50.54 | 57.24 | 58.24  | 50.54  | **62.8**  | 55.56       |
| YELP-10k   | 50.02 | 50.02 | 50.15 | 58.96  | 53.13  | **63.63** | 57.1        |
| SCITAIL-10k| 49.62 | 49.62 | 57.21 | 79.02  | 49.62  | **82.36** | 67.64       |
| AGNEWS-10k | 10.0  | 10.0  | 21.94 | 44.49  | 15.14  | **52.33** | 21.44       |
| IMDB-10k   | 33.33 | 33.33 | 38.1  | 56.09  | 35.22  | **57.18** | 44.89       |
| SCICITE    | 24.67 | 24.67 | 25.14 | 25.33  | 24.67  | **31.81** | 25.96       |
| CHEMPROT   | 33.46 | 33.46 | 33.46 | 33.26  | 33.46  | **33.56** | 33.46       |
| MNLI-10k   | 0.0   | 0.0   | 6.4   | **29.24** | 2.67  | 27.0      | 20.51       |
| SST-10k    | 50.92 | 50.92 | 55.59 | 60.92  | 53.36  | **63.88** | 56.86       |
| Mean       | 31.98 | 31.98 | 37.24 | 48.85  | 33.25  | **51.85** | 41.96       |
| Average Rank | 5.96  | 5.88  | 3.75  | 3.25   | 5.08   | **1.17**  | 2.92        |

Table 43: Results for training top layer for different task after finetuning entire model for RTE
| Tasks            | STD++ | DA   | WC   | ReInit | R3F   | CAPCORT-I | CAPCORT-MLP |
|------------------|-------|------|------|--------|-------|-----------|-------------|
| RTE              | 1.0   | 1.0  | 0.98 | 0.94   | 1.0   | 0.91      | 0.98        |
| MRPC             | 0.97  | 0.92 | 0.9  | 0.93   | 0.94  | 0.95      | **0.99**    |
| QNLI-10k         | 0.97  | 0.97 | 0.96 | **0.98** | 0.98  | 0.96      | 0.96        |
| SCITAIL-10k      | 0.98  | 0.95 | 0.96 | 0.96   | 0.97  | 0.96      | **0.98**    |
| IMDB-10k         | 0.97  | 0.97 | 0.98 | **0.98** | 0.98  | 0.96      | 0.97        |
| SST-10k          | 0.94  | 0.94 | 0.95 | 0.96   | 0.94  | 0.95      | **0.97**    |
| COLA             | 0.94  | 0.94 | 0.93 | 0.94   | **0.96** | 0.94      | 0.95        |
| AGNEWS-10k      | 0.65  | 0.66 | 0.66 | **0.68** | 0.66  | 0.65      | 0.64        |
| QQP-10k          | 0.98  | 0.97 | 0.94 | 0.98   | **1.0** | 0.93      | 0.97        |
| MNL-10k          | 0.98  | 0.93 | 0.94 | 0.89   | 0.97  | 0.85      | 0.91        |
| YELP-10k         | 0.98  | 0.97 | 0.97 | 0.99   | **0.99** | 0.97      | 0.98        |
| CHEMPROT         | 0.59  | 0.49 | 0.49 | 0.52   | 0.51  | 0.51      | **0.68**    |
| SCICITE          | 0.86  | 0.87 | 0.87 | **0.92** | 0.9   | 0.84      | 0.86        |

Table 44: Normalized average of top-1 eigenvalues

| Tasks            | STD++ | DA   | WC   | ReInit | R3F   | CAPCORT-I | CAPCORT-MLP |
|------------------|-------|------|------|--------|-------|-----------|-------------|
| RTE              | 1.0   | 1.0  | 0.99 | 0.97   | 1.0   | 0.95      | **0.99**    |
| MRPC             | 0.98  | 0.97 | 0.95 | 0.94   | 0.98  | 0.97      | **0.99**    |
| QNLI-10k         | 0.99  | 0.99 | 0.98 | **0.99** | 0.99  | 0.98      | 0.98        |
| SCITAIL-10k      | **0.99** | 0.98 | 0.98 | 0.99   | 0.99  | 0.98      | **0.99**    |
| IMDB-10k         | 0.99  | 0.99 | 0.99 | **0.99** | 0.99  | 0.98      | 0.99        |
| SST-10k          | 0.97  | 0.97 | 0.97 | 0.98   | 0.97  | 0.98      | **0.98**    |
| COLA             | 0.97  | 0.96 | 0.96 | 0.97   | **0.98** | 0.98      | 0.97        |
| AGNEWS-10k      | 0.9   | 0.9  | 0.9  | **0.95** | 0.91  | 0.91      | 0.81        |
| QQP-10k          | 1.0   | 0.99 | 0.98 | 0.99   | **1.0** | 0.96      | 0.99        |
| MNL-10k          | **0.99** | 0.96 | 0.98 | 0.96   | 0.98  | 0.93      | 0.97        |
| YELP-10k         | 0.99  | 0.99 | 0.98 | 1.0    | **1.0** | 0.99      | 0.99        |
| CHEMPROT         | 0.7   | 0.61 | 0.61 | 0.64   | 0.64  | 0.63      | **0.77**    |
| SCICITE          | 0.92  | 0.92 | 0.92 | **0.96** | 0.95  | 0.9       | 0.92        |

Table 45: Normalized average of top-2 eigenvalues

| Tasks            | STD++ | DA   | WC   | ReInit | R3F   | CAPCORT-I | CAPCORT-MLP |
|------------------|-------|------|------|--------|-------|-----------|-------------|
| RTE              | **1.0** | 1.0  | 1.0  | 0.99   | 1.0   | 0.98      | 1.0         |
| MRPC             | 0.99  | 0.99 | 0.98 | 0.95   | 0.99  | 0.99      | **1.0**     |
| QNLI-10k         | 1.0   | 1.0  | 0.99 | **1.0** | 1.0   | 1.0       | 0.99        |
| SCITAIL-10k      | 1.0   | 0.99 | 0.99 | **1.0** | 1.0   | 1.0       | 1.0         |
| IMDB-10k         | 1.0   | 1.0  | 1.0  | **1.0** | 1.0   | 0.99      | 1.0         |
| SST-10k          | 0.99  | 0.99 | 0.99 | **1.0** | 0.99  | 0.99      | 1.0         |
| COLA             | 0.99  | 0.99 | 0.98 | 0.99   | **1.0** | 0.99      | 0.99        |
| AGNEWS-10k      | 0.97  | 0.97 | 0.96 | **0.99** | 0.97  | 0.97      | 0.97        |
| QQP-10k          | 1.0   | 1.0  | 0.99 | 1.0    | **1.0** | 0.99      | 1.0         |
| MNL-10k          | **1.0** | 0.99 | 0.99 | 0.99   | 1.0   | 0.98      | 0.99        |
| YELP-10k         | 1.0   | 1.0  | 0.99 | **1.0** | 1.0   | 1.0       | 1.0         |
| CHEMPROT         | 0.89  | 0.85 | 0.85 | 0.88   | 0.87  | 0.86      | **0.92**    |
| SCICITE          | 0.98  | 0.97 | 0.97 | **0.99** | 0.99  | 0.97      | 0.98        |

Table 46: Normalized average of top-5 eigenvalues
| Tasks       | STD++ | DA   | WC   | ReInit | R3F   | CAPCORT-I | CAPCORT-MLP |
|-------------|-------|------|------|--------|-------|------------|--------------|
| RTE         | 1.0   | 1.0  | 1.0  | 0.99   | 1.0   | 0.99       | 1.0          |
| MRPC        | 1.0   | 0.99 | 0.99 | 0.96   | 1.0   | 1.0        | 1.0          |
| QNLI-10k    | 1.0   | 1.0  | 1.0  | 1.0    | 1.0   | 1.0        | 1.0          |
| SCITAIL-10k | 1.0   | 1.0  | 1.0  | 1.0    | 1.0   | 1.0        | 1.0          |
| IMDB-10k    | 1.0   | 1.0  | 1.0  | 1.0    | 1.0   | 1.0        | 1.0          |
| SST-10k     | 1.0   | 1.0  | 0.99 | 1.0    | 1.0   | 1.0        | 1.0          |
| COLA        | 0.99  | 0.99 | 0.98 | 0.99   | 1.0   | 1.0        | 1.0          |
| AGNEWS-10k  | 0.98  | 0.98 | 0.98 | 0.99   | 0.98  | 0.98       | 0.99         |
| QQP-10k     | 1.0   | 1.0  | 1.0  | 1.0    | 1.0   | 0.99       | 1.0          |
| MNLI-10k    | 1.0   | 0.99 | 0.99 | 0.99   | 1.0   | 0.99       | 1.0          |
| YELP-10k    | 1.0   | 1.0  | 1.0  | 1.0    | 1.0   | 1.0        | 1.0          |
| CHEMPROT    | 0.98  | 0.97 | 0.97 | 0.98   | 0.98  | 0.98       | 0.99         |
| SCICITE     | 0.99  | 0.99 | 0.98 | 1.0    | 1.0   | 0.98       | 0.99         |

Table 47: Normalized average of top-10 eigenvalues

| Tasks       | STD++ | DA   | WC   | ReInit | R3F   | CAPCORT-I | CAPCORT-MLP |
|-------------|-------|------|------|--------|-------|------------|--------------|
| RTE         | 1.0   | 1.0  | 1.0  | 1.0    | 1.0   | 0.99       | 1.0          |
| MRPC        | 1.0   | 1.0  | 1.0  | 0.97   | 1.0   | 1.0        | 1.0          |
| QNLI-10k    | 1.0   | 1.0  | 1.0  | 1.0    | 1.0   | 1.0        | 1.0          |
| SCITAIL-10k | 1.0   | 1.0  | 1.0  | 1.0    | 1.0   | 1.0        | 1.0          |
| IMDB-10k    | 1.0   | 1.0  | 1.0  | 1.0    | 1.0   | 1.0        | 1.0          |
| SST-10k     | 1.0   | 1.0  | 0.99 | 1.0    | 1.0   | 1.0        | 1.0          |
| COLA        | 1.0   | 1.0  | 0.99 | 0.99   | 1.0   | 1.0        | 1.0          |
| AGNEWS-10k  | 0.99  | 0.99 | 0.98 | 0.99   | 0.99  | 0.99       | 0.99         |
| QQP-10k     | 1.0   | 1.0  | 1.0  | 1.0    | 1.0   | 1.0        | 1.0          |
| MNLI-10k    | 1.0   | 0.99 | 0.99 | 0.99   | 1.0   | 0.99       | 1.0          |
| YELP-10k    | 1.0   | 1.0  | 1.0  | 1.0    | 1.0   | 1.0        | 1.0          |
| CHEMPROT    | 0.99  | 0.99 | 0.98 | 0.99   | 0.99  | 0.99       | 1.0          |
| SCICITE     | 0.99  | 0.99 | 0.98 | 1.0    | 1.0   | 0.99       | 0.99         |

Table 48: Normalized average of top-20 eigenvalues

| Tasks       | STD++ | DA   | WC   | ReInit | R3F   | CAPCORT-I | CAPCORT-MLP |
|-------------|-------|------|------|--------|-------|------------|--------------|
| RTE         | 0.06  | 0.05 | 151.08 | 356.83 | 4.78  | 497.38     | 140.33       |
| MRPC        | 248.44 | 489.14 | 573.15 | 254.92 | 437.79 | 368.16     | 181.92       |
| QNLI-10k    | 198.62 | 192.21 | 296.25 | 144.09 | 197.74 | 316.57     | 296.03       |
| SCITAIL-10k | 182.57 | 369.51 | 263.48 | 252.59 | 234.66 | 354.42     | 252.86       |
| IMDB-10k    | 248.12 | 244.33 | 179.67 | 142.64 | 211.15 | 299.57     | 293.71       |
| SST-10k     | 365.78 | 400.84 | 333.75 | 281.04 | 403.95 | 360.22     | 348.41       |
| COLA        | 405.79 | 444.48 | 406.54 | 301.56 | 292.11 | 413.04     | 402.62       |
| AGNEWS-10k  | 1124.36 | 1126.63 | 1083.96 | 818.83 | 1099.94 | 1272.38   | 1272.38      |
| QQP-10k     | 109.81 | 163.71 | 380.98 | 211.71 | 0.07  | 497.27     | 234.59       |
| MNLI-10k    | 119.46 | 393.29 | 287.30 | 671.92 | 175.47 | 849.31     | 432.63       |
| YELP-10k    | 177.91 | 244.31 | 238.38 | 97.23  | 87.91  | 259.35     | 198.54       |
| CHEMPROT    | 1103.13 | 1351.65 | 1347.29 | 1369.47 | 1390.28 | 1352.48   | 1193.84      |
| SCICITE     | 867.7  | 844.45 | 752.98 | 630.59 | 633.57 | 906.15     | 775.52       |

Mean: 396.29 | 481.89 | 484.22 | 425.65 | 397.65 | 584.28     | 463.33       |
Average Rank: 4.85 | 3.31 | 3.85 | 5.15 | 4.92 | 2.0 | 3.92

Table 49: GM-5
| Tasks     | STD++ | DA    | WC    | ReInit | R3F   | CAPCORT-I | CAPCORT-MLP |
|-----------|-------|-------|-------|--------|-------|-----------|-------------|
| RTE       | 0.01  | 0.01  | 28.58 | 54.68  | 0.54  | **113.27** | 24.64       |
| MRPC      | 47.83 | 85.21 | 108.67| 65.41  | 59.4  | 61.13     | 19.25       |
| QNLI-10k  | 29.3  | 34.28 | 54.86 | 16.92  | 29.65 | 47.15     | 46.11       |
| SCITAIL-10k| 25.23 | 58.43 | 47.44 | 26.62  | 31.48 | 49.61     | 33.93       |
| IMDB-10k  | 32.82 | 32.53 | 29.47 | 12.69  | 23.31 | **48.51** | 35.45       |
| SST-10k   | 58.83 | 68.08 | 65.03 | 27.32  | 67.7  | 59.49     | 44.76       |
| COLA      | 69.32 | 82.06 | 84.79 | 44.59  | 42.05 | 68.46     | 71.23       |
| AGNEWS-10k| 220.03| **221.59**| 186.54| 95.6   | 52.22| 90.66     | 179.2       |
| QQP-10k   | 10.5  | 25.4  | 64.85 | 22.53  | 0.01  | **164.62**| 71.13       |
| MNLI-10k  | 20.29 | 82.66 | 46.99 | 115.1  | 34.8  | 33.49     | 25.33       |
| YELP-10k  | 19.27 | 33.91 | 39.9  | 8.13   | 10.12 | **190.12**| 446.36      |
| CHEMPROT  | 499.58| **646.59**| 613.5 | 603.17 | 601.67| 619.9     | 446.36      |
| SCICITE   | 164.7 | 171.53| 163.86| 88.84  | 108.6 | **152.38**|             |

| Mean      | 92.13 | 118.64| 118.04| 90.89  | 93.2  | **134.96**|             |
| Average Rank | 5.08  | 2.62  | 2.85  | 5.38   | 5.31  | **2.31**  | 4.46        |

Table 50: GM-10

| Tasks     | STD++ | DA    | WC    | ReInit | R3F   | CAPCORT-I | CAPCORT-MLP |
|-----------|-------|-------|-------|--------|-------|-----------|-------------|
| RTE       | 0.0   | 0.0   | 5.75  | 12.17  | 0.09  | **24.83** | 4.34        |
| MRPC      | 7.55  | 15.06 | 20.88 | **24.8**| 8.25  | 10.41     | 2.65        |
| QNLI-10k  | 4.01  | 5.22  | **9.67**| 2.0    | 4.2   | 6.66      | 7.5         |
| SCITAIL-10k| 2.89 | 7.58  | **8.7**| 3.04   | 3.58  | 5.72      | 4.64        |
| IMDB-10k  | 3.38  | 3.55  | 5.36  | 1.03   | 2.38  | **7.34**  | 4.65        |
| SST-10k   | 9.62  | 11.93 | **14.43**| 3.63   | 11.54 | 10.26     | 6.77        |
| COLA      | 12.85 | 14.76 | **20.1**| 10.88  | 6.18  | 11.65     | 12.35       |
| AGNEWS-10k| **38.7**| 38.02| 37.23 | 15.59  | 35.16 | 36.44     | 32.92       |
| QQP-10k   | 0.89  | 2.69  | 10.25 | 2.28   | 0.0   | **16.01** | 5.4         |
| MNLI-10k  | 3.44  | 15.14 | 9.09  | 16.15  | 6.35  | **29.53** | 12.0        |
| YELP-10k  | 1.78  | 3.68  | **6.86**| 0.54   | 1.05  | 4.18      | 3.36        |
| CHEMPROT  | 69.87 | **93.37**| 81.62 | 65.05  | 80.71 | 86.34     | 48.23       |
| SCICITE   | 29.61 | 30.0  | 32.32 | 12.79  | 17.54 | **37.96** | 27.58       |

| Mean      | 14.2  | 18.54 | 20.17 | 13.07  | 13.62 | **22.1**  | 13.26       |
| Average Rank | 5.08  | 3.08  | **2.08**| 5.38   | 5.38  | 2.46      | 4.54        |

Table 51: GM-20

30
| Tasks          | STD++ | DA   | WC    | ReInit | R3F   | CAPCORT-I | CAPCORT-MLP |
|---------------|-------|------|-------|--------|-------|-----------|-------------|
| RTE           | 0.0   | 0.0  | 46.98 | 129.47 | 0.63  | **210.7** | 50.95       |
| MRPC          | 95.55 | 198.64 | **228.53** | 75.07 | 138.94 | 127.19    | 29.27       |
| QNLI-10k      | 69.03 | 67.1  | 106.82 | 36.64  | 58.54  | **108.56** | 88.44       |
| SCITAIL-10k   | 52.56 | **123.99** | 105.75 | 61.73  | 74.35  | 111.24    | 71.91       |
| IMDB-10k      | 69.28 | 67.86 | 60.84  | 24.22  | 58.72  | **90.53**  | 64.7        |
| SST-10k       | 118.65 | 145.93 | 124.11 | 47.52  | **153.48** | 129.43    | 90.6        |
| COLA          | 122.08 | 171.42 | 157.71 | 63.75  | 95.61  | 127.75    | 133.4       |
| AGNEWS-10k    | 540.72 | 570.89 | 495.34 | 179.74 | 514.01 | **702.69** | 702.69      |
| QQP-10k       | 21.54 | 55.95 | 125.51 | 54.76  | 0.0    | **172.94** | 60.37       |
| MNLI-10k      | 59.91 | 193.92 | 84.33  | 262.68 | 84.15  | **416.25** | 166.81      |
| YELP-10k      | 37.49 | **78.53** | 78.31  | 17.42  | 19.99  | 64.93     | 44.16       |
| CHEMPROT      | 980.72 | 1196.85 | 1201.61 | 1199.8 | **1222.6** | 1187.46  | 924.5       |
| SCICITE       | 414.13 | 418.66 | 338.99 | 250.67 | 272.92 | **476.9**  | 394.16      |

| Mean          | 198.59 | 253.06 | 242.68 | 184.88 | 207.23 | **290.74** | 217.07      |
| Average Rank  | 5.0    | 2.85   | 3.31   | 5.62   | 4.77   | **2.31**   | 4.15        |

Table 52: HM-5

| Tasks          | STD++ | DA   | WC    | ReInit | R3F   | CAPCORT-I | CAPCORT-MLP |
|---------------|-------|------|-------|--------|-------|-----------|-------------|
| RTE           | 0.0   | 0.0  | 9.0   | 14.79  | 0.1   | **41.73** | 7.15        |
| MRPC          | 15.09 | 24.73 | **33.48** | 26.9  | 13.53  | 17.29     | 3.31        |
| QNLI-10k      | 6.99  | 9.65  | **16.49** | 3.13  | 7.1    | 11.6      | 12.37       |
| SCITAIL-10k   | 5.46  | **14.88** | 14.11  | 4.35   | 6.74   | 10.75     | 7.91        |
| IMDB-10k      | 6.7   | 6.83  | 7.81  | 1.7    | 4.09   | **11.92** | 6.44        |
| SST-10k       | 15.31 | 19.3  | **21.21** | 4.19  | 18.37  | 16.18     | 9.32        |
| COLA          | 19.72 | 24.21 | **28.76** | 11.5  | 9.88   | 19.34     | 20.15       |
| AGNEWS-10k    | 65.88 | **67.33** | 54.52  | 18.78  | 60.41  | 62.36     | 44.45       |
| QQP-10k       | 1.56  | 5.6   | 17.67 | 3.68   | 0.0    | **26.38** | 9.33        |
| MNLI-10k      | 5.68  | 27.28 | 13.1  | 28.45  | 10.77  | **50.47** | 20.74       |
| YELP-10k      | 3.28  | 7.42  | **11.0** | 0.95  | 1.88   | 7.14      | 4.95        |
| CHEMPROT      | 302.28 | **429.53** | 333.57 | 356.82 | 332.11 | 389.25    | 226.06      |
| SCICITE       | 48.68 | 54.7  | 56.06 | 18.57  | 29.54  | **59.65** | 45.86       |

| Mean          | 38.2  | 53.19 | 47.44 | 37.99  | 38.04  | **55.7**  | 32.16       |
| Average Rank  | 5.08  | 2.77  | **2.31** | 5.31  | 5.46   | 2.46      | 4.62        |

Table 53: HM-10
| Tasks         | STD++ | DA | WC | ReInit | R3F | CAPCORT-I | CAPCORT-MLP |
|--------------|-------|----|----|--------|-----|------------|--------------|
| RTE          | 0.0   | 0.0| 1.93| 4.48   | 0.02| **9.05**   | 1.24         |
| MRPC         | 2.04  | 4.6| 6.63| **13.83**| 2.13| 3.05       | 0.61         |
| QNLI-10k     | 0.9   | 1.31| **2.96**| 0.39   | 0.98| 1.5        | 2.04         |
| SCITAIL-10k  | 0.55  | 1.53| **2.65**| 0.6    | 0.68| 1.04       | 1.1          |
| IMDB-10k     | 0.52  | 0.62| 1.63| 0.13   | 0.37| **1.84**   | 1.02         |
| SST-10k      | 2.61  | 3.52| **5.33**| 0.83   | 3.29| 2.94       | 1.73         |
| COLA         | 3.99  | 4.49| **7.88**| 4.37   | 1.52| 3.28       | 3.6          |
| MRPC         | 2.04  | 4.6| 6.63| **13.83**| 2.13| 3.05       | 0.61         |
| QNLI-10k     | 0.9   | 1.31| **2.96**| 0.39   | 0.98| 1.5        | 2.04         |
| SCITAIL-10k  | 0.55  | 1.53| **2.65**| 0.6    | 0.68| 1.04       | 1.1          |
| IMDB-10k     | 0.52  | 0.62| 1.63| 0.13   | 0.37| **1.84**   | 1.02         |
| SST-10k      | 2.61  | 3.52| **5.33**| 0.83   | 3.29| 2.94       | 1.73         |
| COLA         | 3.99  | 4.49| **7.88**| 4.37   | 1.52| 3.28       | 3.6          |

Table 54: HM-20

| Tasks         | STD++ | DA | WC | ReInit | R3F | CAPCORT-I | CAPCORT-MLP |
|--------------|-------|----|----|--------|-----|------------|--------------|
| CHEMPROT     | 584.01| 1015.37| 702.38| 589.39| 1415.28| 826.09     | 1141.63      |
| MNLI-10k     | 1506.74| 2643.21| 1723.52| 1514.14| 3746.99| 1832.83    | 3022.97      |
| SCITAIL-10k  | 1678.89| 2962.31| 1896.3| 1684.06| 4217.93| 2418.12    | 3130.63      |
| MRPC         | 162.09| 258.92| 214.94| 157.42| 362.83| 226.59     | 299.13       |
| QNLI-10k     | 1683.58| 2966.56| 1894.97| 1681.34| 4232.07| 2414.78    | 3124.71      |
| QQP-10k      | 1272.46| 2194.95| 1487.52| 1276.92| 3078.56| 2307.62    | 2302.41      |
| SST          | 5050.32| 8537.84| 5997.04| 5080.05| 11878.49| 6019.1     | 9568.79      |
| COLA         | 239.78| 375.78| 349.94| 240.61| 515.27| 292.32     | 398.9        |
| SCICITE      | 961.03| 1746.23| 1066.78| 962.4| 2838.07| 1850.68    | 1835.6       |
| IMDB-10k     | 1679.74| 2975.29| 1899.58| 1692.33| 4220.9| 3149.3     | 3124.5       |
| YELP-10k     | 1681.65| 2975.09| 1894.85| 1686.21| 4224.52| 2426.62    | 3130.62      |
| RTE          | 307.13| 504.8| 396.35| 309.26| 698.16| 527.76     | 529.66       |
| AGNEWS-10k   | 589.23| 1046.27| 675.37| 595.06| 1449.9| 1125.27    | 1227.09      |

Table 55: Training time for different methods