Efficient Multi-Task Auxiliary Learning: Selecting Auxiliary Data by Feature Similarity

Po-Nien Kung Yi-Cheng Chen Sheng-Siang Yin Tse-Hsuan Yang Yun-Nung Chen
Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
{b06902012, b06902011, b06902103, b06902032}@csie.ntu.edu.tw y.v.chen@ieee.org

Abstract

Multi-task auxiliary learning utilizes a set of relevant auxiliary tasks to improve the performance of a primary task. A common usage is to manually select multiple auxiliary tasks for multi-task learning on all data, which raises two issues: (1) selecting beneficial auxiliary tasks for a primary task is nontrivial; (2) when the auxiliary datasets are large, training on all data becomes time-expensive and impractical. Therefore, this paper focuses on addressing these problems and proposes a time-efficient sampling method to select the data that is most relevant to the primary task. The proposed method allows us to only train on the most beneficial sub-datasets from the auxiliary tasks, achieving efficient multi-task auxiliary learning. The experiments on three benchmark datasets (RTE, MRPC, STS-B) show that our method significantly outperforms random sampling and ST-DNN. Also, by applying our method, the model can surpass fully-trained MT-DNN on RTE, MRPC, STS-B, using only 50%, 66%, and 1% of data, respectively.[1]

1 Introduction

In recent years, language model pre-training has achieved great success in almost all NLP fields (Devlin et al., 2019; Lan et al., 2019; Liu et al., 2019c; Lewis et al., 2020; Radford et al.; Yang et al., 2019). By learning from large corpus text segments without supervision, the models are able to learn the general representation of word tokens and can be further fine-tuned on downstream tasks. Moreover, many downstream tasks have their related tasks, which may benefit from the shared information in the training signal. To better utilize the shared knowledge, multi-task learning (MTL) is a common technique. The recent work (Liu et al., 2019a; Raffel et al., 2020; Aghajanyan et al., 2021) focused on capturing the shared knowledge by learning multiple tasks simultaneously between the pre-training and fine-tuning stage, in order to benefit the downstream tasks. Even though the massive MTL scheme is demonstrated to achieve the improvement in terms of performance, it is either time or computing costly; for example, Aghajanyan et al. (2021) used over 4.8 million total labeled examples for MTL.

On the other hand, in numerous cases, MTL is applied but we only aim at a single task performance. It is usually addressed as **multi-task auxiliary learning**, which targets to introduce auxiliary tasks and datasets to boost the performance of the primary task (Chen et al., 2018; Du et al., 2018; Guo et al., 2019). In this scenario, how to wisely select the auxiliary tasks plays the most important role. One straightforward method is to select the auxiliary tasks according to their relatedness to the primary task. However, selecting the "related" tasks is non-trivial. To address to this, Guo et al. (2019) proposed AutoSem, which learns to automatically select auxiliary tasks and decide the mixing ratio of auxiliary data via a Beta-Bernoulli multi-armed bandit with Thompson Sampling and Gaussian Process, respectively. However, their method cannot decide the specific data samples to use in one auxiliary task, and the sampling approach is extremely time-consuming due to the numerous steps needed to solve the non-stationary multi-armed bandit problem, failing to address the issue of efficiency in MTL.

In this paper, we propose a similarity-based sampling method, along with a two-stage MTL pipeline for efficient multi-task auxiliary learning. The experiments on the GLUE (Wang et al., 2018) benchmark show that the proposed method outperforms single-task models, MT-DNN with random sampling, and even fully-trained MT-DNN with full auxiliary data. The analysis also demonstrates the effectiveness and efficiency of our proposed two-stage MTL pipeline. This paper has three-fold

[1]The source code is available at: https://github.com/MiuLab/FastMTL/.

contributions:

- We propose a time-efficient sampling method to speed up auxiliary MTL learning.
- We propose an automatic auxiliary data sampling method that focuses on deciding the specific data samples instead of the mixing ratio.
- The experiments demonstrate that the proposed approach outperforms the single-task and random-sampling MT-DNN. Furthermore, the model using less data also surpasses the fully-trained MT-DNN.

2 Related Work

2.1 Multi-Task Learning (MTL)

Multi-task learning (MTL) (Caruana, 1997) is an inductive transfer mechanism for improving generalization performance by learning tasks in parallel while using the shared representation. The main idea is that the model can take advantage of information extracted from one task to benefit training on another.

Liu et al. (2019a) proposed a multi-task deep neural network (MT-DNN), which combines MTL and language model pre-training to achieve SOTA results comparing to the original single-task deep neural network (ST-DNN) setting for many natural language understanding tasks. In the framework, a pre-trained model, BERT (Devlin et al., 2019), is trained with multiple tasks (ex. all tasks of GLUE (Wang et al., 2018)) in parallel before fine-tuning.

Recently, massive multi-task learning, which acquires much more tasks for MTL, is gaining popularity. MUPPET (Aghajanyan et al., 2021) proposed an additional stage called pre-finetuning between language model pre-training and fine-tuning. Pre-finetuning is similar to large-scale MTL, which contains around 50 datasets, over 4.8 million labeled examples in total. This method encourages the learning of general representations across different tasks, showing better performance on a wide range of tasks.

In addition, Raffel et al. (2020) proposed text-to-text transfer Transformer (T5), where each NLP task can be formulated as a "text-to-text" problem. Hence, we can leverage all tasks into the same training and decoding procedure while applying a shared model. However, such massive MTL methods required tremendous computation resources and training data, resulting in poor efficiency.

2.2 Multi-Task Auxiliary Learning

When training data is scarce, using auxiliary tasks can provide additional generality and improve the performance of the primary task. However, choosing highly correlated tasks and applying delicately chosen weights are essential.

Chen et al. (2018) balanced task influence by using gradient normalization, which prevents overfitting on single auxiliary tasks. In Shi et al. (2020), the auxiliary tasks are automatically re-weighted to minimize data usage and retain performance on the primary tasks.

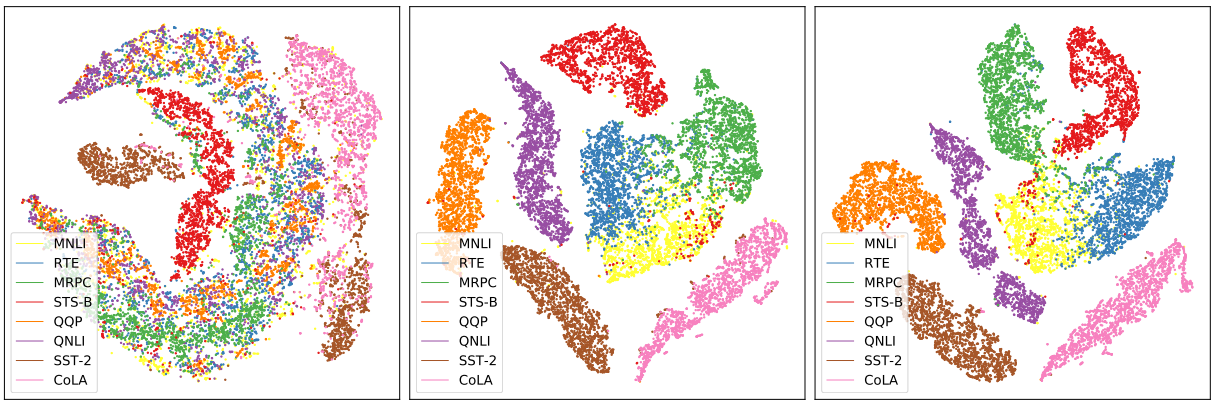
In addition to the methods of weighting training gradient or loss, another way to clinch improvement on the primary task is to select tasks delicately. For example, Du et al. (2018) used cosine similarity to decide whether the auxiliary task is beneficial to the training.

Moreover, AutoSem (Guo et al., 2019) is a pipeline combining both aspects of weighted loss and task selection, which first measures the utility of each candidate task through Thompson sampling and decides the weights of chosen tasks via the Gaussian process. The method, however, is extremely time-consuming due to the complex optimization of the multi-armed bandit problem.

2.3 Data Sampling

In MTL scenarios, using large datasets is getting prevailing, so data sampling has been widely discussed in many machine learning fields, either to reduce the label data or training time. To reduce the usage of labeled data, active learning focuses on sampling the most beneficial data without knowing the labels. The selection mechanisms can be categorized into three types, including uncertainty-based approaches (Xue et al., 2007; Joshi et al., 2009; Wang et al., 2016; Yoo and Kweon, 2019; Gal and Ghahramani, 2016; Gal et al., 2017), expected-change-based approaches (Roy and Mccallum, 2001; Settles et al., 2007; Freytag et al., 2014), and diversity-based approaches (Sener and Savarese, 2018; Nguyen and Smeulders, 2004; Guo, 2010).

Active learning aims to choose the most effective data for training, which is similar to our goal. Nonetheless, under the active learning scheme, the query strategy does not access the labels of the data, while we have full access to them. Moreover, many active learning and core-set sampling methods also face the problem of overwhelming time



(a) Toy MT-DNN Model (b) TD-MTDNN Model – BCE (c) TD-MTDNN Model – CE

Figure 1: T-SNE Visualization of model's last hidden state features. We train these toy models using 500 data from each task's training set and visualize the unused training data using T-SNE clustering. In all figures, we only plot 2000 random sample data points from each task to improve visibility.

consumption, which needs additional effort to deal with (Kirsch et al., 2019; Ni et al., 2015; Coleman et al., 2020).

To better address the issue about *time-efficiency*, we propose a method that can *efficiently* sample the most beneficial data by utilizing **hidden-state similarity**. Our idea is inspired by the fact that information in hidden states has been proven to be beneficial for measuring data similarity. For instance, Manhattan LSTM (Mueller and Thyagarajan, 2016) measures the exponential of L1 distance between the last LSTM hidden states of 2 sentences to learn sentence similarity. Moreover, pair-wise distillation encourages a student model to learn to generate a feature map similar to that of the teacher (Liu et al., 2019b) for semantic segmentation. Recently, Gonzalez et al. (2019) analyzed the embedding of an auto-encoder to optimize data for faster training in single-task scenarios.

3 Proposed Method

This paper proposes a two-stage MTL pipeline for efficient multi-task auxiliary learning. In the first stage, we introduce *similarity sampling*, a simple and time-efficient sampling strategy to select the most beneficial data samples from the auxiliary tasks to benefit the primary task. In the second stage, we train the MT-DNN model using the selected auxiliary data similar to Liu et al. (2019a). After training, we fine-tune the model on the primary task to further optimize the performance.

3.1 Motivation

The idea of *similarity sampling* is based on the assumption that the more similar to the primary task an auxiliary data is, the more benefit it can contribute to the primary task. To verify this assumption, we train a toy MT-DNN (Liu et al., 2019a) and visualize the last hidden states of all data in Figure 1a, which shows that most tasks are mixing and confusing, but the brown (SST-2) and pink (CoLA) points are more separate from the other six tasks. The observation aligns with the results in Table 1, where the performance in these two tasks degrades while MTL on all GLUE tasks.

Furthermore, to better distinguish between the tasks, we borrow the idea from Du et al. (2020) and train a task-aware toy MT-DNN model by multi-task training an additional task-discrimination objective, using the binary cross entropy (BCE) loss and the cross entropy (CE) loss, and the data is visualized in Figure 1b and 1c. We name this task-aware toy MT-DNN as TD-MTDNN. By training the model to distinguish between tasks, the scatter points of each task are more diverse. The QQP, QNLI, SST-2, and CoLA are nearly entirely divided from other tasks but MNLI, RTE, MRPC, and STS-B have some overlap between areas, implying that these four tasks have a degree of similarity to each other and can benefit more from MTL. The finding matches the results in Table 1, where RTE, MRPC, and STS-B are the most MTL-benefit datasets in all tasks. The reason that MNLI is not improved by MTL may be its large training set (393k shown in Table 2), which is hard to further benefit from the other 3 tiny tasks with a total of 13.2k data.

Inspired by the above findings, the proposed method uses the last hidden state as features to determine whether the data samples are similar to the data of the primary task and may benefit its per-

	MNLI-m/mm	RTE	MRPC	STS-B	QQP	QNLI	SST-2	CoLA
ST-DNN	83.61 / 83.05	64.97	87.05 / 82.56	84.65 / 82.74	70.44 / 88.91	90.49	93.95	53.32
MT-DNN	83.42 / 82.59	**75.57**	**88.67 / 84.83**	**86.23 / 85.42**	70.38 / 88.97	90.60	93.43	46.44

Table 1: The test results of 8 datasets in GLUE. ST-DNN is a BERT-Base model with fine-tuning for a single task, and MT-DNN model is a multi-task model learned on all tasks and further fine-tuned on each single task. The reported scores are the average over 10 runs.

Corpus	Task	#Train	#Test	Metrics	Domain
MNLI	NLI	393k	20k	Matched / Mismatched Acc.	Misc.
RTE	NLI	2.5k	3k	Acc.	News, Wikipedia
MRPC	Paraphrase	3.7k	1.7k	Acc./F1	News
STS-B	Sentence Similarity	7k	1.4k	Pearson/Spearman Corr.	Misc.
QQP	Paraphrase	364k	391k	Acc./F1	Social QA
QNLI	QA/NLI	105k	5.4k	Acc.	Wikipedia
SST-2	Sentiment	67k	1.8k	Acc.	Movie Review
CoLA	Acceptability	8.5k	1k	Matthews Corr.	Misc.

Table 2: 8 datasets in GLUE benchmark used in our experiments.

formance. Here we use the task-discriminator to predict the similarity value that indicates whether a data sample is similar to the data in each task, and the top-ranked data samples are used for multi-task auxiliary learning. In this paper, we extend the original MT-DNN training process to a *two-stage multi-task auxiliary learning pipeline* illustrated in Figure 2.

3.2 Stage 1: Task-Discriminative MT-DNN & Similarity Ranking

In the first stage, our goal is to build a model that can efficiently measure the similarity between the auxiliary data and the primary data. As shown in the left of Figure 2, we first train a task-discriminative MT-DNN (TD-MTDNN) by using small sets of all datasets (500 samples for each), which is a tiny MT-DNN model with an additional task-discriminator. In TD-MTDNN, the primary task, all auxiliary tasks, and a task discriminator are learned in an MTL setting. The reason for using MTL for all tasks instead of only training the discriminator is to allow the model to encode the *task information* into the model weights, which is the knowledge of the primary task and all auxiliary tasks.[2] Without learning the task information, the model may learn how to discriminate tasks only based on the data context instead of the task-specific knowledge. The trained task discriminator is to determine how much similar to each task a data sample is, and such prediction results can be viewed as the similarity for the following sampling process (Coleman et al., 2020).

To better describe our proposed method, we define notations as follows. In multi-task auxiliary learning, T^p denotes a primary task with training data D^p and N auxiliary tasks $T^{Ai}, i \in \{1, 2, ..., N\}$ with training data $D^{Ai}, i \in \{1, 2, ..., N\}$. To train TD-MTDNN, we randomly sample 500 training data from all tasks to form sub-datasets $D^p_{sub500} \subset D^p$ and $D^{Ai}_{sub500} \subset D^{Ai}, i \in \{1, 2, ..., N\}$. These sub-datasets are used for training a TD-MTDNN model via MTL similar to Liu et al. (2019a).

After training TD-MTDNN, we input all remaining auxiliary data $D^A_{unused} = \bigcup_{i=1}^{N} D^{Ai} \setminus D^{Ai}_{sub500}$ and allow the task discriminator to predict which task a data sample belongs to. The output of the discriminator is a $N + 1$ dimension vector, where each element indicates how much similar to a task and can be viewed as the similarity to a task for the input data.

As illustrated in Figure 2, we can easily rank all auxiliary data samples by their similarity to the primary task, and the top-ranked data samples are selected as D^A_{best}, the subset of auxiliary data that can benefit the primary task most. Hence, the second stage only needs to utilize the relatively small set D^A_{best} instead of the full sets to achieve efficient multi-task auxiliary learning.

[2]The claim is validated though the experiments in 5.3.

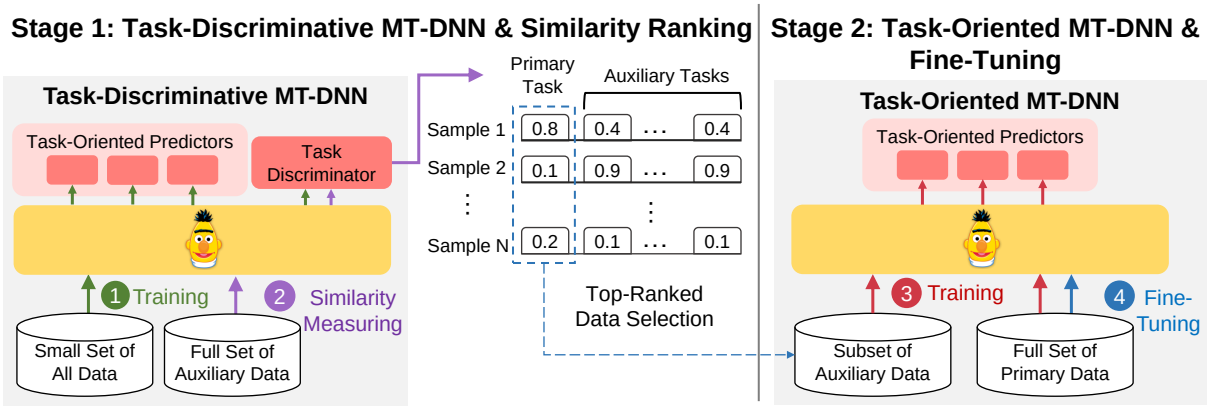


Figure 2: The proposed two-stage multi-task auxiliary learning pipeline.

3.3 2nd Stage: Task-Oriented MT-DNN & Finetuning

In the second stage, we use the D^A_{best} acquired in the previous stage with the full primary task data D^p to train a primary task-oriented MT-DNN (TO-MTDNN). The training process is basically the same as Liu et al. (2019a), which alternately trains all tasks. Note that this model is different from the one in the first stage and the auxiliary data used in this model is significantly less than the prior work. After training TO-MTDNN, we further fine-tune the model on D^p to boost the performance of the primary task.

4 Experiments

To evaluate the proposed method, we conduct the experiments detailed below.

4.1 Data

Following the setting in MT-DNN (Liu et al., 2019a), we used eight datasets (MNLI, RTE, MRPC, STS-B, QQP, QNLI, SST-2, CoLA) from GLUE (General Language Understanding Evaluation Benchmark) (Wang et al., 2018) in our experiment. The data statistics can be found in Table 2.

In our multi-task auxiliary learning setting, we first select primary tasks and use other datasets as auxiliary tasks. According to the results in Table 1, it is seen that only RTE, MRPC, and STS-B significantly benefit from multi-task learning, so we choose these three datasets as our primary tasks to evaluate the usefulness of our proposed MTL method.

4.2 Experimental Setup

For all experiments, we use BERT-Base as the backbone structure and add a linear layer for each task-oriented predictor or a task-discriminator. We conduct the following experiments in different settings.

TD-MTDNN When training TD-MTDNN, two loss functions are applied for task discrimination, **Binary Cross Entropy Loss (BCE)** and **Cross Entropy Loss (CE)**, the former of which learns to discriminate tasks as a multi-label classification problem, and the prediction of task similarity will be inclusive, and the latter learns to predict task similarity exclusively.

TD-MTDNN provides the similarity scores for all auxiliary data (excluding the data for training TD-MTDNN), and we further sample the $N \in \{500, 1000, ..., 512000\}$ top-ranked data samples for training TO-MTDNN.

TO-MTDNN The data amount N for training TO-MTDNN starts from 500, and double each time until reaching 512,000. Considering that the size of full auxiliary data is about 900,000, we also perform on the settings with $N = \{600000, 700000, 800000\}$.

We provide the detail of the used model structure, evaluation metrics, hyperparameter search, and other training details in Appendix A B.

4.3 Baselines

- **ST-DNN** is the single-task deep neural network fine-tuned on each task separately, which is a weak baseline to show the overall efficiency of MTL.
- **Random sampling** baselines follow the similar setting as our proposed method but without

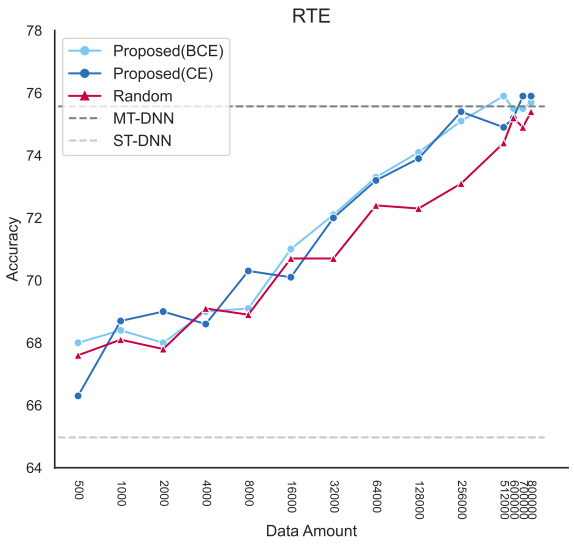


Figure 3: RTE Accuracy with different sampling data amount for all methods.

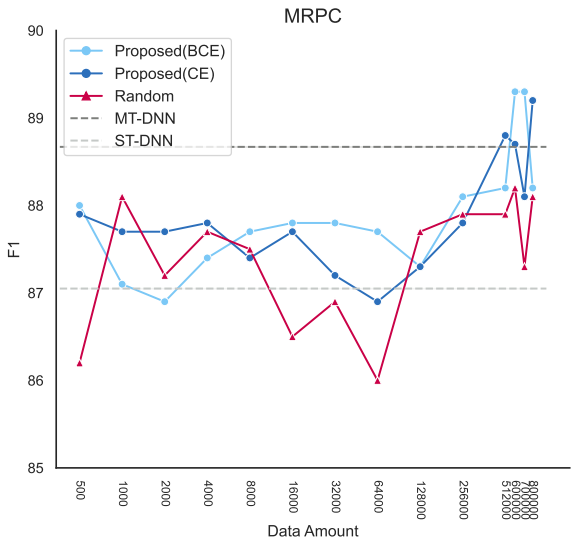


Figure 4: MRPC F1 with different sampling data amount for all methods. We only plot the F1 curve since the Accuracy curve is nearly identical.

selecting auxiliary data by a task discriminator. Here we randomly sample the auxiliary data samples and use them for TO-MTDNN. This method has been proved to be a strong baseline in the scenarios of multi-task learning GLUE benchmark (Glover and Hokamp, 2019).

- **Fully-trained MT-DNN** uses all auxiliary data at the multi-task training stage, which is regarded as the performance upper-bound but suffers from its poor efficiency.

4.4 GLUE Results

We show the performance of three primary tasks in Figure 3, 4, and 5.[3] ST-DNN performs worst compared to other MTL methods. Compared to random sampling baselines, our approaches perform significantly better for all primary tasks when the data is sufficient (>64000). In RTE and STS-B, our methods are consistently better for all sampling amounts. Also, when sampling sufficient data (50%, 60%, 1%), our method can even outperform the fully-trained MT-DNN, which is the strong baseline trained on full auxiliary datasets. The finding indicates the effectiveness of our sampling method in multi-task auxiliary learning settings. Parenthetically, the fact that our proposed method achieves better performance than fully-trained MT-DNN tells that using too much data may not benefit the model performance due to noises. When there exists some data causing negative transfer, our method can distinguish valuable data samples from distracting ones.

To explicate the strengths of our method, we further investigate the distribution of the sampled data and show the results in Figure 6. Rather than sampling in the same proportion for each task, our method has a preference to sample data from specific tasks. For RTE and MRPC, we can see that both BCE and CE focus on MNLI and STS-B. The distribution conforms to the similarity of tasks presented in Figure 1b, suggesting RTE, MRPC, and MNLI data or tasks are more alike. This can also explain the performance progress of our method.

In all three tasks, the performances of BCE-trained TD-MTDNN are more satisfied and stable than CE-trained ones. We conjecture the reason is due to the properties of those two loss functions. CE leads to exclusive prediction, causing the predicted similarity affected by any other. On the contrary, using BCE allows the model to predict the similarity of each task independently, which meets our desire that the discriminator focuses on which data are more related to the primary task.

4.5 Time Efficiency

The main goal is to efficiently perform multi-task auxiliary learning, so we show the time consumption of each stage of all methods in Table 3. The results are based on TD-MTDNN trained with 500 training instances and TO-MTDNN with 10,000 sampled data run on the same machine. In the second stage for TO-MTDNN, the time consumption

[3]Readers can refer to Appendix D for more concrete scores and scores of other metrics.

	Stage 1: TD-MTDNN		Stage 2: TO-MTDNN		Total Runtime(s)
	Training	Ranking	Training	Fine-Tuning	
MT-DNN	- -	- -	15,801	90 / 120 / 190	15,891 / 15,921 / 15,991
Random			200 / 220 / 260		**290 / 340 / 450**
Proposed	95	775			510 / 560 / 670

Table 3: Runtime(s) of different models on each training stage. The three numbers separated by slash refers to the consumption of RTE / MRPC / STS-B, respectively.

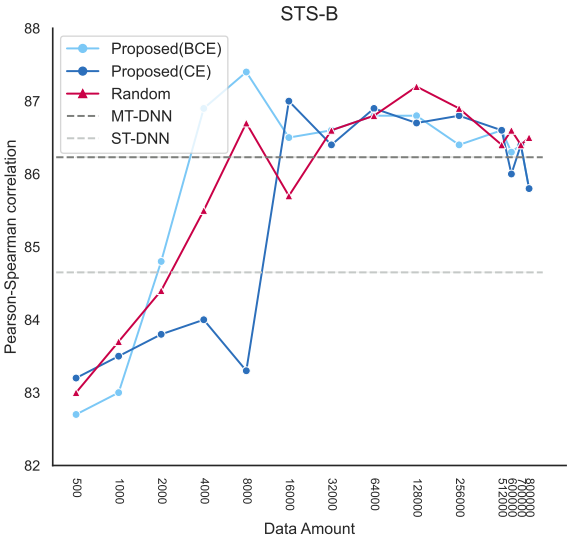


Figure 5: STS-B Pearson Correlation with different sampling data amount for all methods. We only plot the Pearson Correlation curve since the Spearman Correlation curve is nearly identical.

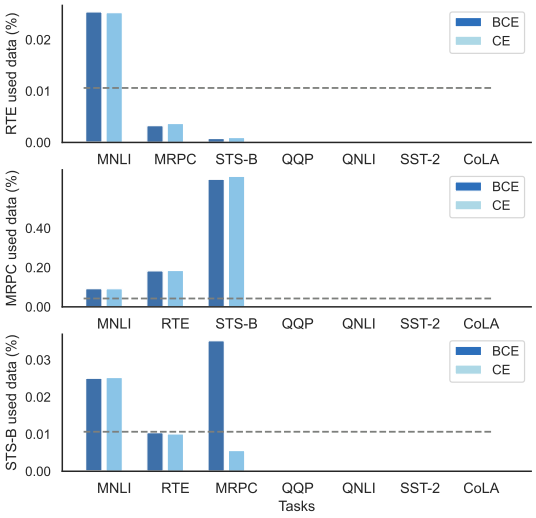


Figure 6: The sampled data distribution in our experiments when sampling 10,000 data for TO-MTDNN. The y-axis is the percentage of the sampled data in an auxiliary dataset. Dotted lines denote the data distribution sampled by random sampling.

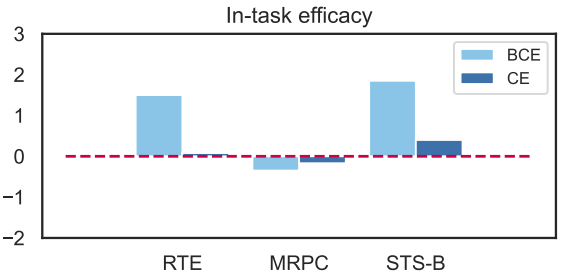


Figure 7: In-task efficacy (performance gain) of using the selected samples on RTE, MRPC, and STS-B. The unit of the y-axis is the percentage of the evaluation metrics, which are accuracy, F1, and Pearson-Spearman Corr, respectively.

of our method is much less than that of fully-trained MT-DNN, because only a subset of auxiliary data is utilized. Furthermore, the additional cost for the first stage of TD-MTDNN is negligible comparing to the reduced time in TO-MTDNN. From the above results, we demonstrate that the proposed approach is able to achieve comparable performance with fully-trained MT-DNN while using less data (27% in RTE, 53% in MRPC, and 1.7% in STS-B), which is approximately proportional to the total training time. Generally, the outcomes also align with our theoretical analysis of time complexity in Appendix C.

5 Discussion

To better investigate whether the selected samples are more beneficial than others, we investigate the in-task efficacy of our method. Also, considering that multi-task learning often benefits more the low-resource tasks by preventing overfitting through learning other tasks, we analyze the performance of the proposed method in the low-resource scenarios.

Furthermore, we conduct ablation experiments to justify that multi-task learning GLUE (Wang et al., 2018) tasks in the TD-MTDNN stage does help the discriminator to learn the relation between tasks better.

5.1 In-Task Efficacy

For multi-task auxiliary learning, the prior work (Guo et al., 2019; Glover and Hokamp, 2019) focused on deciding the mixing ratios of all auxiliary tasks. Different from it, our proposed method not only decides the mixing ratio but also selects the **specific data** to use in each auxiliary task. That leads to a further question: *Does our method pick out the most beneficial data samples in one auxiliary task, or does the improvement only come from the proper mixing ratio?*

To answer this question, we evaluate the usefulness of the sampled data in each auxiliary task. Here we first apply our similarity sampling to select 10,000 auxiliary data samples, and then we obtain the mixing ratios for auxiliary tasks. In order to check whether our selected data samples are better, we fix the mixing ratio and resample data in each auxiliary task. We train TO-MTDNN with 10,000 data samples in these two settings and show the results in Figure 7. **In-Task Efficacy** is defined as the performance gain when training on our selected data compared to the re-sampled data, so larger in-task efficacy indicates that the selected data is more beneficial.

For BCE, the results show that in both RTE and STS-B, the performance drops significantly when using the resampled data. However, in MRPC, there is no in-task efficacy of our sampling method. The model can achieve similar performance using either our selected data or the resampled data, which is not surprising considering that the BCE method does not improve much on MRPC when training on only 10,000 auxiliary data samples.

For CE, there is nearly no in-task efficacy in all three tasks, probably because of its relatively poor performance compared to BCE. The reason that CE is worse than BCE is that the model using CE as the task-discrimination loss predicts the similarity value between tasks exclusively, so the original similarity may be normalized when considering all tasks at the same time and make the comparison among data samples inaccurate.

These results show that when using our proposed method with BCE-TD, our method reveals in-task efficacy on some tasks and can sample the most beneficial data in those tasks.

5.2 Few-Shot on Primary Tasks

We conduct experiments in a few-shot scenario to evaluate our method. The data amount of primary tasks is restricted in every stage. For the first stage for TD-MTDNN, we use 500 samples for each auxiliary tasks and $\min(500, |D^p|)$ samples for primary tasks. We apply the weighted loss to balance the ratio between tasks if the data amount of the primary task is less than 500. We show the performance of the proposed method and the random sampling baseline with different data sizes of primary tasks in Figure 8, where the performance difference is illustrated in the bar format for better comparison. The results show that in few-shot settings, our work outperforms the random sampling baseline with a greater margin compared to the original setting. That indicates when the primary task data is scarce, our proposed method can better utilize the auxiliary task knowledge to improve the primary task more. Also, Table 3 tells that our method can efficiently reduce the usage of auxiliary data, implying that time reduction of multi-task auxiliary learning using our method is more significant when the primary task is small. Considering the advantage of our method in both performance and computation aspects, our method is highly suitable for multi-task auxiliary learning in a low primary task resource setting.

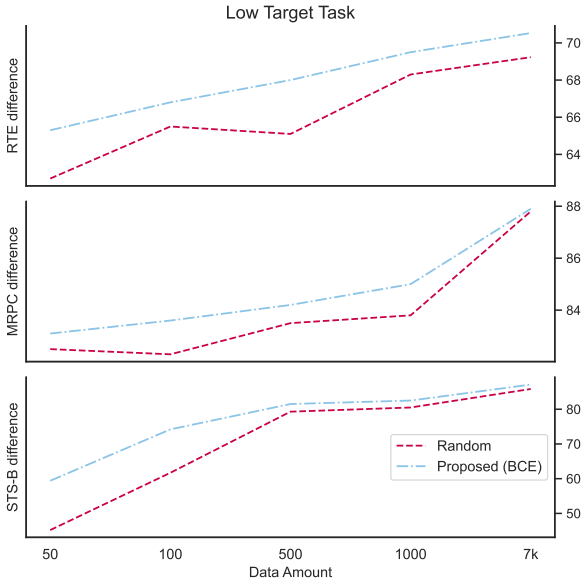


Figure 8: The performance of the proposed method and random sampling baseline with restricted data amount of the primary task. The result with original amount of data is demonstrated at the rightmost.

5.3 Multi-Task Learning in TD-MTDNN

In the proposed method described in 3.2, we multi-task learn all GLUE (Wang et al., 2018) tasks and the task-discriminative loss together to train a Task-

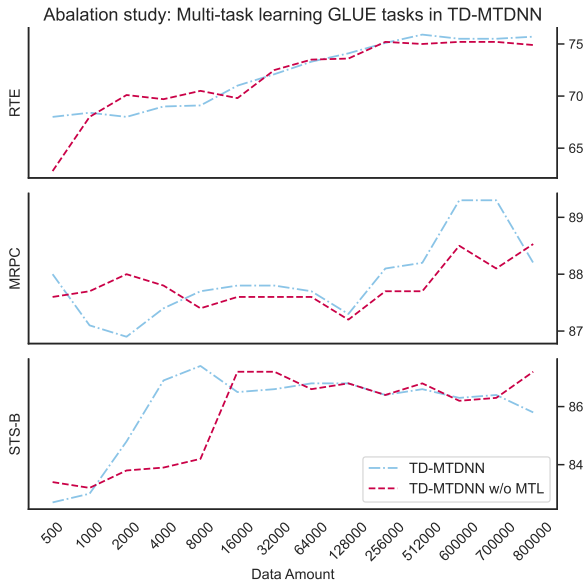


Figure 9: The performance of the proposed method (TD-MTDNN) and MTL-ablation version (TD-MTDNN w/o MTL).

Discriminative MT-DNN. The hypothesis here is that the model can better predict the similarity scores for all data points when knowing both semantics (text information) and task information. To further verify the above hypothesis, we compare the performance of two models, the proposed one and one without multi-task learning on all GLUE tasks when training TD-MTDNN.

Figure 9 shows the results of the ablation study. Overall, two methods (TD-MTDNN with & w/o MTL) obtain similar performance curves, especially for RTE. For MRPC, *TD-MTDNN* performs better than *TD-MTDNN w/o MTL* when we use a larger amount of data for TO-MTDNN. In contrast, the same trend is observed with a smaller amount of data for STS-B. Among three tasks, all best scores are performed by *TD-MTDNN*, showing the usefulness of multi-task learning on all GLUE tasks in the TD-MTDNN training stage.

6 Conclusion

This paper introduces a novel two-stage multi-task auxiliary learning framework that utilizes similarity sampling to select the most beneficial auxiliary data for efficiently training an MT-DNN model. Our experiments on benchmark GLUE datasets demonstrate that our proposed method outperforms random sampling and further surpasses the fully-trained MT-DNN with significantly fewer data and time. Moreover, we show that our selected samples are the most beneficial data in the auxiliary task and that the proposed method works much better when few-shot scenarios, proving the strong in-task efficacy and the great potential of practical usage.

Acknowledgments

We thank reviewers for their insightful comments. This work was financially supported from the Young Scholar Fellowship Program by Ministry of Science and Technology (MOST) in Taiwan, under Grant 110-2636-E-002-003.

References

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021. Muppet: Massive multi-task representations with pre-finetuning. *arXiv preprint arXiv:2101.11038*.

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

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pages 794–803. PMLR.

C Coleman, C Yeh, S Mussmann, B Mirzasoleiman, P Bailis, P Liang, J Leskovec, and M Zaharia. 2020. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations (ICLR)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Chunning Du, Haifeng Sun, Jingyu Wang, Qi Qi, and Jianxin Liao. 2020. Adversarial and domain-aware bert for cross-domain sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4019–4028.

Yunshu Du, Wojciech M Czarnecki, Siddhant M Jayakumar, Mehrdad Farajtabar, Razvan Pascanu, and Balaji Lakshminarayanan. 2018. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*.

Alexander Freytag, Erik Rodner, and Joachim Denzler. 2014. Selecting influential examples: Active learning with expected model output changes. In *European conference on computer vision*, pages 562–577. Springer.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR.

John Glover and Chris Hokamp. 2019. Task selection policies for multitask learning. *arXiv preprint arXiv:1907.06214*.

Santiago Gonzalez, Joshua Landgraf, and Risto Miikkulainen. 2019. Faster training by selecting samples using embeddings. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2019. Autosem: Automatic task selection and mixing in multi-task learning. *arXiv preprint arXiv:1904.04153*.

Yuhong Guo. 2010. Active instance sampling via matrix partition. *Advances in Neural Information Processing Systems*, 23:802–810.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. 2009. Multi-class active learning for image classification. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2372–2379. IEEE.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. 2019. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 32:7026–7037.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496.

Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. 2019b. Structured knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2604–2613.

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

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Hieu T Nguyen and Arnold Smeulders. 2004. Active learning using pre-clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 79.

Chongjia Ni, Cheung-Chi Leung, Lei Wang, Nancy F Chen, and Bin Ma. 2015. Unsupervised data selection and word-morph mixed language model for tamil low-resource keyword search. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4714–4718. IEEE.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners.

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

Nicholas Roy and Andrew Mccallum. 2001. Toward optimal active learning through monte carlo estimation of error reduction.

Ozan Sener and Silvio Savarese. 2018. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*.

Burr Settles, Mark Craven, and Soumya Ray. 2007. Multiple-instance active learning. *Advances in neural information processing systems*, 20:1289–1296.

Baifeng Shi, Judy Hoffman, Kate Saenko, Trevor Darrell, and Huijuan Xu. 2020. Auxiliary task reweighting for minimum-data learning. *arXiv preprint arXiv:2010.08244*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang, and Liang Lin. 2016. Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2591–2600.

Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. 2007. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8(1).

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.

Donggeun Yoo and In So Kweon. 2019. Learning loss for active learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 93–102.

A Training Details

We use Adam (Kingma and Ba, 2015) as the optimizer with a learning rate 2e-5. The training batch size is 32, and we train the model for 5, 3, and 5 epochs for these three stages respectively.

We have done the hyperparameter search on $Batch\_Size \in \{4, 8, 16, 32, 64\}, Epoch \in \{3, 5, 10\}, lr \in \{2e{-}5, 5e{-}5\}$ on the fully-trained MT-DNN. For the model selection strategy in each stage, we use the last epoch of TD-MTDNN and TO-MTDNN. For the last fine-tuning stage, we select the best epoch by the score of Dev-set and then use the model to predict the Test-set and submit the results to GLUE Benchmark. For all experiments, a machine with CPU - AMD R7 3700X(8 cores); GPU - RTX2080S; 16GB RAM is used.

B Evaluation

We follow the standard GLUE metrics to evaluate the performance of our models. These include accuracy, F1, Pearson-Spearman correlation (PSC), and Mathews correlation coefficient (MCC).

Accuracy The basic evaluation metric for a classification task, which is simply the proportion of correct predictions. It is used in SST-2, MRPC, QQP, MNLI, QNLI, and RTE.

F1 The metric penalizes models which tend to predict the class with a higher probability to obtain high accuracy but fail to predict the minor class correctly. It is used in MRPC and QQP.

PSC The correlation to evaluate the relationship between prediction and ground truth values in a regression task, ranging from -1 to 1. Pearson correlation coefficient assesses linear relationships, whilst Spearman's assesses monotonic relationships which is not limited to linear one. It is used in STS-B.

MCC The correlation ranging from -1 to 1 to evaluate a classification task. Similar to F1, it requires correct predictions on both classes. Nonetheless, it is independent of which class is defined as positive. It is used in CoLA.

C Time Complexity Analysis

Table 4 shows our analysis of the time complexity of each stage and method, and the denotation is described in detail in the caption. For AutoSem (Guo et al., 2019), the first term $T_{s1} \cdot (K + N_{s1} \cdot C_T)$ is the cost of stage-1 (non-stationary multi-armed bandit), the second term $T_{s2} \cdot N_{s2} \cdot C_T$ is the cost of drawing samples for stage-2 (Gaussian Process), and the last term T_{s2}^3 is the cost of solving Gaussian Process. We recommend readers refer to the paper for more details.

Method / Stage	Time Complexity
Stage 1: TD-MTDNN	
Training	$O(T_{TD} \cdot N_{TD} \cdot C_T)$
Ranking	$O(N_{all} \cdot C_P)$
Stage 2: TO-MTDNN	
Training	$O(T_{TO} \cdot N_{TO} \cdot C_T)$
Fine-Tuning	$O(T_{FT} \cdot N_{pri} \cdot C_T)$
AutoSem	$O(T_{s1} \cdot (K + N_{s1} \cdot C_T) + T_{s2} \cdot N_{s2} \cdot C_T + T_{s2}^3)$

Table 4: Time complexity analysis of each stage/method. T's: steps/epochs. N's: number of instances/mini-batches. C's: cost of training/prediction. K: number of tasks. Note that a fully-trained MT-DNN uses N_{all} data in the TO-MTDNN stage, much larger than N_{TO} in our proposed method.

For a fully-trained MT-DNN, $N_{TO} = N_{all}$, making the TO-MTDNN stage dominant term and training-expensive. Our proposed method selects only a few data for the TO-MTDNN stage, substantially decreasing N_{TO} and significantly alleviating training cost. In exchange, it costs the additional TD-MTDNN stage for our proposed method. Nevertheless, the training instances used in the TD-MTDNN training phase (N_{TD}) are further fewer, and on the other hand, the cost of prediction (C_P) is much less than that of training (C_T). These make the additional cost still dominated by the TO-MTDNN stage with full data, resulting in a worthy trade-off. Our experimental results in 4.5 also verify our analysis.

The method proposed in AutoSem is similar in complexity terms to the stages of our method. However, it takes many steps to steadily solve a non-stationary multi-armed bandit problem, leading to heavy training cost as T_{s1} becomes much larger than other T's. This forces some trade-offs to make the algorithm feasible, such as using a few mini-batches to train then observe the reward (reducing N_{s1}), and using a simpler model like LSTM (Hochreiter and Schmidhuber, 1997) (reducing C_T)[4]. In comparison, our method not only

[4] Again the readers can check these details in the original paper of AutoSem.

# Data	RTE			MRPC			STS-B		
	BCE	CE	Random	BCE	CE	Random	BCE	CE	Random
500	68.0	66.3	67.6	88.0 / 83.4	87.9 / 83.5	86.2 / 81.8	82.7 / 81.0	83.2 / 81.8	83.0 / 81.4
1,000	68.4	68.7	68.1	87.1 / 87.7	87.7 / 83.6	88.1 / 83.5	83.0 / 81.4	83.5 / 82.0	83.7 / 82.1
2,000	68.0	69.0	67.8	86.9 / 82.7	87.7 / 83.7	87.2 / 82.8	84.8 / 83.0	83.8 / 82.1	84.4 / 82.9
4,000	69.0	68.6	69.1	87.4 / 82.7	87.8 / 83.5	87.7 / 83.4	86.9 / 85.7	84.0 / 82.5	85.5 / 84.0
8,000	69.1	70.3	68.9	87.7 / 83.4	87.4 / 83.1	87.5 / 83.2	87.4 / 86.4	83.3 / 81.7	86.7 / 85.5
16,000	71.0	70.1	70.7	87.8 / 83.4	87.7 / 83.6	86.5 / 82.2	86.5 / 85.5	87.0 / 86.1	85.7 / 84.5
32,000	72.1	72.0	70.7	87.8 / 83.1	87.2 / 83.0	86.9 / 82.5	86.6 / 85.8	86.4 / 85.8	86.6 / 85.5
64,000	73.3	73.2	72.4	87.7 / 83.7	86.9 / 82.3	86.0 / 81.2	86.8 / 86.2	86.9 / 86.4	86.8 / 85.9
128,000	74.1	73.9	72.3	87.3 / 83.1	87.3 / 83.1	87.7 / 83.7	86.8 / 86.3	86.7 / 86.2	87.2 / 86.5
256,000	75.1	75.4	73.1	88.1 / 84.2	87.8 / 83.9	87.9 / 83.9	86.4 / 85.9	86.8 / 86.3	86.9 / 86.0
512,000	75.9	74.9	74.4	88.2 / 84.0	88.8 / 85.0	87.9 / 83.9	86.6 / 85.9	86.6 / 85.8	86.4 / 85.7
600,000	75.5	75.2	75.2	89.3 / 85.7	88.7 / 84.9	88.2 / 84.4	86.3 / 85.8	86.0 / 85.1	86.6 / 85.9
700,000	75.5	75.9	74.9	89.3 / 85.6	88.1 / 84.3	87.3 / 83.4	86.4 / 85.6	86.4 / 85.8	86.4 / 85.8
800,000	75.7	75.9	75.4	88.2 / 84.4	89.2 / 85.7	88.1 / 84.3	85.8 / 85.2	85.8 / 84.9	86.5 / 85.8

Table 5: Detailed performance with different amounts of data. For MRPC, the two scores correspond to accuracy/F1. For STS-B, the two scores correspond to Pearson/Spearman correlation coefficient.

runs in favorable time but also successfully scales to more complicated models (BERT).

D Experiment Results in Details

We show our detailed experiment results in Table 5. We use these scores to plot the line graphs in Figure 3, Figure 4, and Figure 5.