Mixture-of-Domain-Adapters: Decoupling and Injecting Domain Knowledge to Pre-trained Language Models' Memories

Shizhe Diao[♡*], **Tianyang Xu**[♠*], **Ruijia Xu**[♡], **Jiawei Wang**[♣], **Tong Zhang**[♡]
[♡]The Hong Kong University of Science and Technology
`{sdiaoaa, rxuaq, tongzhang}@ust.hk`
[♠]Wuhan University
`tianyangxu@whu.edu.cn`
[♣]Shanghai Jiao Tong University
`wjw_sjt@sjtu.edu.cn`

Abstract

Pre-trained language models (PLMs) demonstrate excellent abilities to understand texts in the generic domain while struggling in a specific domain. Although continued pre-training on a large domain-specific corpus is effective, it is costly to tune all the parameters on the domain. In this paper, we investigate whether we can adapt PLMs both effectively and efficiently by only tuning a few parameters. Specifically, we decouple the feed-forward networks (FFNs) of the Transformer architecture into two parts: the original pre-trained FFNs to maintain the old-domain knowledge and our novel domain-specific adapters to inject domain-specific knowledge in parallel. Then we adopt a mixture-of-adapters gate to fuse the knowledge from different domain adapters dynamically. Our proposed Mixture-of-Domain-Adapters (**MixDA**) employs a two-stage adapter-tuning strategy that leverages both unlabeled data and labeled data to help the domain adaptation: i) domain-specific adapter on unlabeled data; followed by ii) the task-specific adapter on labeled data. MixDA can be seamlessly plugged into the pretraining-finetuning paradigm and our experiments demonstrate that MixDA achieves superior performance on in-domain tasks (GLUE), out-of-domain tasks (ChemProt, RCT, IMDB, Amazon), and knowledge-intensive tasks (KILT). Further analyses demonstrate the reliability, scalability, and efficiency of our method.[1]

1 Introduction

Pre-trained language models (PLMs) have achieved a multitude of successful applications in natural language understanding (Devlin et al., 2018; Liu et al., 2019; He et al., 2021b) and generation (Lewis et al., 2019; Zhang et al., 2020; Yang et al., 2020; Brown et al., 2020). The predominant methodology for domain adaptation is fine-tuning on labeled domain-specific data or continued pre-training (Gururangan et al., 2020) on unlabeled domain-specific data. Although effective, both fine-tuning and continued pre-training methods require tuning all the parameters of a PLM, raising high costs beyond many institutions' reach. To mitigate this, multiple parameter-efficient fine-tuning (PEFT) methods are proposed, including prompt-based tuning (Gao et al., 2021; Liu et al., 2021b; Schick and Schütze, 2021; Li and Liang, 2021; Liu et al., 2021a), and adapter-based tuning (Houlsby et al., 2019; Pfeiffer et al., 2020b; Hu et al., 2021). However, they are more concerned about task adaptation and it is still unclear how to regularly, and inexpensively inject domain knowledge into PLMs for different domain-specific tasks. Moreover, directly tuning PLMs on a domain-specific corpus with PEFT methods will lead to the catastrophic forgetting problem (Yogatama et al., 2019; Gururangan et al., 2020). These limitations highlight an important research question: *how to adapt PLMs with the new domain knowledge while keeping the old-domain knowledge unmodified?*

Inspired by the recent studies (Geva et al., 2021; Cao et al., 2021; Meng et al., 2022) that found knowledge is stored in feed-forward networks (FFNs), we decouple the FFNs into two parts: the original pre-trained FFNs to maintain the old-domain knowledge and our novel domain-specific adapters to inject domain-specific knowledge in parallel. Specifically, we propose Mixture-of-Domain-Adapters (MixDA), a mixture of several domain adapters to inject domain-specific knowledge without affecting the old-domain knowledge. Our model has two stages: (i) domain-specific tuning multiple knowledge adapters on unlabeled data and then (ii) task-specific tuning adapters on labeled data. In the first stage, we train several domain adapters on both domain-specific corpus and pre-training corpus simultaneously while keeping the original feed-forward networks unchanged. In

[*]Equal Contribution.
[1]The code is available at `https://github.com/Amano-Aki/Mixture-of-Domain-Adapters`.

the second stage, we train a mixture-of-adapters gate to dynamically select the desired knowledge adapter and a task-specific adapter for task adaptation.

We conduct experiments on a broad range of tasks, including 4 out-of-domain datasets, 9 in-domain datasets, and 2 knowledge-intensive datasets. Our experimental results demonstrate the effectiveness of MixDA on 15 datasets, spanning biomedical, computer science publications, news, and reviews. Further analysis displays three key properties of our proposed approach: (i) **Reliability**: it shows superior performance on both in-domain and out-of-domain tasks. (ii) **Scalability**: it scales well to the increasing number of domains. (iii) **Efficiency**: it adds only a small number of parameters per domain. We claim that these properties are helpful for language models as a service, where a frozen PLM is served, and multiple adapters are inserted to support different customized services.

2 Related Work

In this section, we will review four research lines related to injecting domain knowledge into pre-trained language models: knowledge injection, domain adaptation, parameter-efficient fine-tuning, and mixture-of-adapters.

2.1 Knowledge Injection

Knowledge can be injected into PLMs by pre-training or fine-tuning, each corresponding to a separate research direction. During pre-training, the knowledge carried by knowledge graphs (Zhang et al., 2019; He et al., 2020), entities (Sun et al., 2019; Xiong et al., 2020), n-grams (Diao et al., 2020), knowledge embedding (Wang et al., 2021b), synonym and hyponym-hypernym relations in WordNet (Lauscher et al., 2019), word-supersense knowledge (Levine et al., 2020), and knowledge bases (Peters et al., 2019) can be injected into PLMs by feeding knowledge inputs and designing new objectives. However, pre-training-based methods are costly, making the application to huge PLMs (e.g., models with 175 Billion parameters) impossible. Fine-tuning-based methods only require an additional fine-tuning process. Some studies inject extra information into the input sentences, like knowledge triples from knowledge graphs (Liu et al., 2020) and knowledge context (Faldu et al., 2021), while other studies explored specific model and training designs, like knowledge adapter networks (Wang et al., 2021a), graph convolutional networks and LSTMs (Lin et al., 2019), and meta-learning (Sinitsin et al., 2020). Zhu et al. (2020) formulated knowledge injection as a constrained optimization problem by adding a constraint on the loss on the unmodified facts. Recent studies (Geva et al., 2021; Cao et al., 2021; Meng et al., 2022) reveal that knowledge is stored in the feed-forward networks in PLMs. Inspired by these studies, we propose a new efficient tuning method to inject domain knowledge into feed-forward networks with minimal costs.

2.2 Domain Adaptation

Previous studies have observed that language models suffer from a significant performance drop during the domain shift (Beltagy et al., 2019; Alsentzer et al., 2019; Huang et al., 2019; Lee et al., 2020; Ke et al., 2022b). Effective strategies that can bridge the domain gap are introduced. Pre-training language models from scratch is effective but costly, like SciBERT (Beltagy et al., 2019), BioBERT (Lee et al., 2020), and ClinicalBERT (Alsentzer et al., 2019). Recent studies explored continued pre-training (Gururangan et al., 2020) and adapter networks (Diao et al., 2021) to save time by training on unlabeled downstream task data. In this paper, we introduce plug-in domain adaptors for domain adaptation, which are effective and mitigate catastrophic forgetting issues because of the explicit learning strategy and efficient model architecture.

2.3 Parameter-Efficient Fine-tuning

Another relevant research direction is parameter-efficient fine-tuning (PEFT), which only fine-tunes a small number of parameters. Existing works solve this problem from two perspectives: prompt-based tuning (Gao et al., 2021; Liu et al., 2021b; Schick and Schütze, 2021; Li and Liang, 2021; Liu et al., 2021a), and adapter-based tuning (Houlsby et al., 2019; Pfeiffer et al., 2020b; Hu et al., 2021). Several works in adapter-based tuning are closely related to ours. AdapterFusion (Pfeiffer et al., 2021) aims to combine multiple task adapters but does not offer specific architecture or training strategies to learn external knowledge. DEMix (Gururangan et al., 2022) and MixDA both train adapters that specialize in domains and use mechanisms to route different adapters, but differ in routing methods, base models, and training strategies. K-Adapter (Wang et al., 2021a) is restricted by its training on T-REx triples and lacks the flexibility to train on unstructured knowledge. Similar to MixDA, CPT (Ke et al., 2022a) integrates domain knowledge into LMs, but it employs a different approach. While MixDA uses domain adapters to substitute FFN layers and task adapters to perform end tasks, CPT adds CL-Plugins that learn domain knowledge. Recent work by He et al. (2021a) presents a unified framework that establishes connections across different PEFT methods. Our work can leverage any PEFT method and complement them.

2.4 Mixture-of-Experts

Mixture-of-Experts (MoE) (Shazeer et al., 2017) is introduced with several expert networks, gating networks, and load-balancing techniques. The following studies improve MoE on initialization and training schemes (Fedus et al., 2022), routing mechanisms (Zuo et al., 2021; Yang et al., 2021), and load-balancing issues (Lewis et al., 2021; Roller et al., 2021). AdaMix (Wang et al., 2022) proposed a mixture of adapters to improve the downstream task performance. Instead of mixing different designs of adapters, our domain adapter is a feed-forward network specifically designed for domain knowledge.

3 Approach

Given a pre-trained language model $\mathcal{M}$, the input is a sentence $\mathcal{X} = t_1 t_2 \cdots t_i \cdots t_T$ (t_i indicates the i-th token) and the output is the representation of each token. The overall architecture of our model is shown in Figure 1. The training process is divided into two-stage. In Stage 1 (Figure 1 (a)), we inject new feed-forward networks (FFNs) (namely domain-adapter) paralleled to the original pre-trained FFNs in some Transformer layers, acting as a key-value memory. The newly injected domain-adapter is trained on both domain-specific unlabeled data and original pre-training unlabeled data to store new factual associations while keeping old-domain ones. All modules are frozen except domain-adapter in this stage. In Stage 2 (Figure 1 (b)), we train a mixture-of-adapters (MoA) gate and a task-adapter on downstream tasks with labeled data, and only these two new modules are updated. The MoA gate receives outputs from the old-domain FFNs and domain-adapter, then outputs a weighted sum of them. An additional task-adapter is inserted in each Transformer block to facilitate downstream tasks. Figure 1 (c) shows the structures of the domain-adapter and the MoA gate.

In this section, we first introduce domain-adapter, which learns and stores domain-specific knowledge, and then describe task-adapters that perform the downstream task. Finally, we discuss how the MoA gate integrates the outputs from the FFN and the domain-adapter.

3.1 Domain-Adapter

Previous studies (Geva et al., 2021; Cao et al., 2021; Meng et al., 2022) suggest that factual associations are stored in the FFNs of some Transformer layers. To help models learn domain-specific knowledge, we propose a lightweight domain-adapter that works parallel to the FFNs, and a training method to learn domain-specific knowledge alongside keeping old-domain ones. Domain-adapter has a simple bottleneck architecture consisting of a down projection layer, a nonlinearity (such as ReLU (Agarap, 2018)), and an up projection layer. This helps keep the parameter size low (Houlsby et al., 2019) with competitive performance.

In Stage 1, the domain-adapter is trained with the domain-specific and old-domain datasets in one batch. Note that all other parameters are frozen except the domain-adapter at this stage. Let $\mathcal{L}_K$ denote the **knowledge loss** related to domain-specific knowledge, and $\mathcal{L}_S$ denote the **sampling loss** related to old-domain knowledge. The knowledge loss is a cross-entropy loss on predicting masked tokens, and the sampling loss is designed to align the latent spaces of the old-domain knowledge and new domain-specific knowledge. The total loss $\mathcal{L}$ is given by a weighted sum of the two, that is:

$$\mathcal{L} = \lambda \cdot \mathcal{L}_K + \mathcal{L}_S, \tag{1}$$

where λ is a weight for the knowledge loss.

The **knowledge loss** is implemented by using cross-entropy loss. Given a sentence with M mask tokens whose answers are $m_1, m_2, \cdots, m_M$, respectively, the knowledge loss $\mathcal{L}_K$ is given by

$$\mathcal{L}_K = -\frac{1}{M} \sum_{i=1}^{M} \log p(m_i), \tag{2}$$

where $p(m_i)$ is the probability for token m_i output by $\mathcal{M}$. Our model accepts two types of domain-specific knowledge as follows, showing improved versatility.

- **Structured knowledge** If the knowledge dataset is structured (e.g., ConceptNet (Speer et al.,

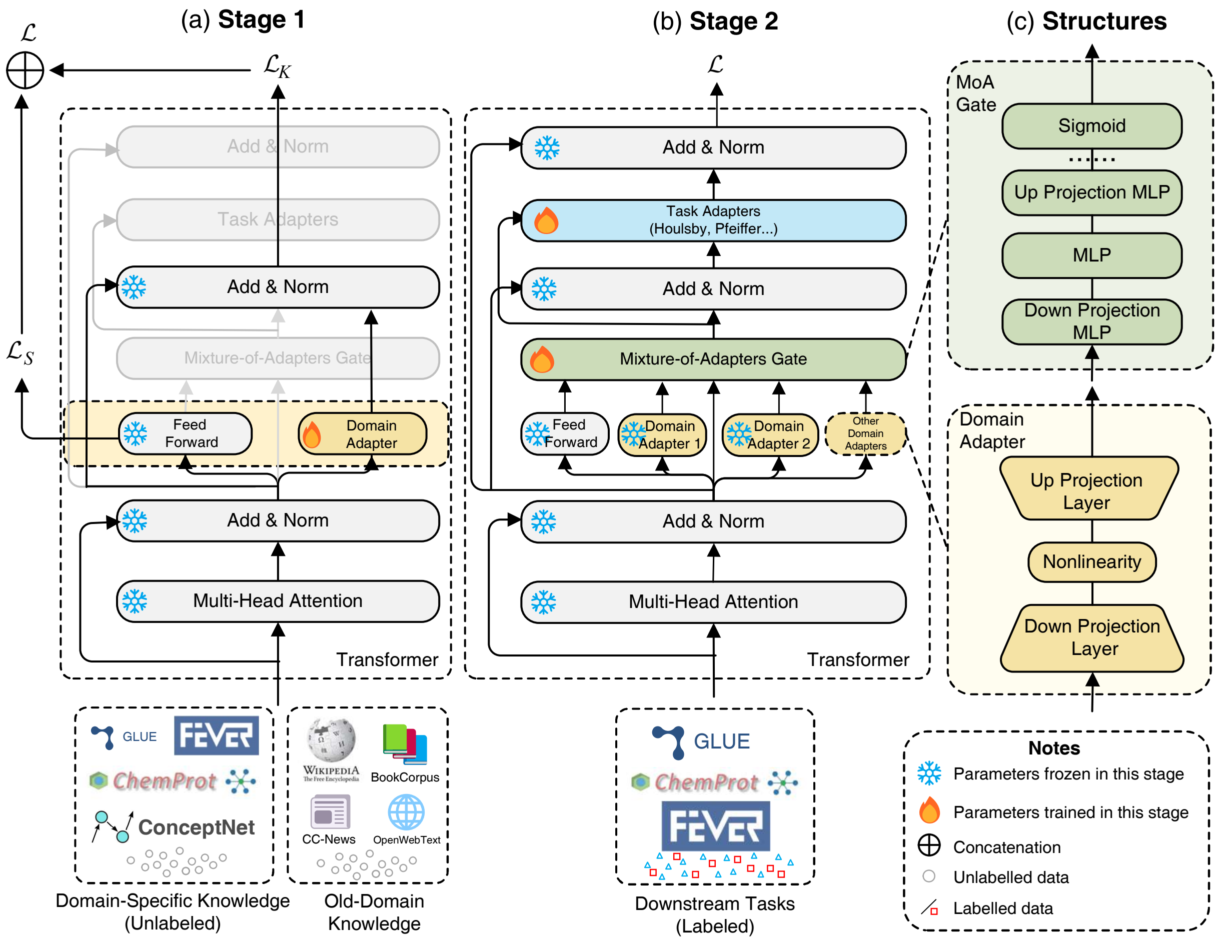

Figure 1: The overall structure of the model. Our training method includes two stages: (a) In Stage 1, we introduce domain-adapters into the model and freeze other parameters. The model learns from domain-specific knowledge (knowledge loss $\mathcal{L}_K$) and keeps similar outputs with the FFN on old-domain knowledge (sample loss $\mathcal{L}_S$). $\mathcal{L}_K$ and $\mathcal{L}_S$ are then combined into the total loss $\mathcal{L}$. (b) In Stage 2, we introduce the mixture-of-adapters gate and task-adapters, then freeze the domain-adapter. The model is trained to perform downstream tasks, which gives us the total loss $\mathcal{L}$. (c) shows the detailed structures of the domain-adapter and the MoA gate.

2016)), we translate each relation into a sentence, and then mask out its object. For example, the relation "the Eiffel tower–/r/LocatedAt–Paris" is translated into "The Eiffel Tower is located at Paris.", then "Paris" is substituted with the mask token, and the model is trained to fill the mask.

- **Unstructured knowledge** For unstructured knowledge (e.g., downstream unlabeled texts), we use the masked language model (MLM) similar to RoBERTa pretraining. Some tokens are randomly sampled from the input sentence and replaced with the special token `<mask>`, and the model is trained to predict the masked token. The cross-entropy loss is calculated to optimize the model.

For old-domain knowledge and sampling loss, we train the model on general corpora including Wikipedia and BookCorpus (Zhu et al., 2015). Specifically, for each batch, sentences randomly sampled from the dataset are input into the model. Given L layers that have domain-adapters installed, for each such layer l, we collect token representations from the FFN F_l, and representations from the domain-adapter K_l. The goal is to keep them as similar as possible. Thus, we calculate the **sampling loss** $\mathcal{L}_S$ with L2 loss:

$$\mathcal{L}_S = \frac{1}{L}\sum_{l=1}^{L} ||F_l - K_l||_2^2. \quad (3)$$

3.2 Task-Adapter

After training domain-adapters, the model is aware of the domain knowledge, which is not directly related to downstream tasks though. Therefore, we add task-adapters on top of the domain-adapter to adapt to downstream tasks. For example, a domain-adapter trained in biomedical knowledge can sup-

Domain	Tasks	Domain Knowledge	# Tokens	Size
Biomed	ChemProt, RCT	2.68K papers about biology and chemistry from S2ORC (Lo et al., 2020)	33.6M	144MB
Review	Amazon, IMDB	24.75K randomly selected Amazon reviews	7.4M	34MB
ID	GLUE tasks	Corpus of GLUE tasks	29.0M	146MB
KI	FEVER, CSQA	Corpus of both CommonsenseQA and FEVER datasets	5.9M	34MB

Table 1: Domain knowledge in Stage 1 training.

port different tasks in the domain, while training it on a task limits its capability to the specific task. Task-adapters can be any adapter architecture or other parameter-efficient fine-tuning methods, such as the Houlsby adapter (Houlsby et al., 2019), Pfeiffer adapter (Pfeiffer et al., 2020b), prefix-tuning (Li and Liang, 2021), and so on. At Stage 2, all parameters other than the task-adapters and the MoA gate (Section 3.3) are frozen. The training of the adapter follows its corresponding approach, despite the addition of domain-adapters. For example, for a text classification task, we add a classification layer on top of the model, freeze all parameters other than the classification layer, the MoA gate, and the task-adapters, feed input texts into the model, and use cross-entropy as the loss.

3.3 Mixture-of-Adapters Gate

On downstream tasks, it is possible that the output from the FFN, or a weighted sum of the two, produces better results. Therefore, in Stage 2, we train an additional **mixture-of-adapters** (MoA) gate simultaneously. The MoA gate receives the outputs from the attention layer q, the domain-adapter K, and the FFN F. $\mathbf{q}$ is first sent into a multi-layer perceptron (MLP):

$$\mathbf{h} = \mathrm{MLP}(\mathbf{q}). \tag{4}$$

The MLP is composed of a down-projection layer W_d and an up-projection layer W_u, and $\mathbf{h} = W_u\sigma(W_d\mathbf{q})$, where σ is the nonlinearity function. Then, $\mathbf{h}$ is input into a Sigmoid layer to generate the weights of the FFNs and other domain-adapters:

$$\mathbf{w} = \mathrm{Sigmoid}(\mathbf{h}). \tag{5}$$

The final output $\mathbf{o}$ is a weighted sum of the outputs of the FFNs and the domain-adapter:

$$\mathbf{o} = \mathbf{w}[K; F], \tag{6}$$

where $[;]$ denotes matrix concatenation.

4 Experimental Settings

In this section, we first introduce the datasets, then the baseline models, the evaluation metrics, and implementation details in the following four subsections, respectively.

4.1 Datasets

We conduct experiments on three types of datasets: **in-domain (ID)** tasks that require general-domain knowledge; **out-of-domain (OOD)** tasks that require domain-specific knowledge; **knowledge-intensive (KI)** tasks that require commonsense knowledge.

- **ID**: GLUE Benchmark (Wang et al., 2018) including MNLI (Williams et al., 2017), CoLA (Warstadt et al., 2019), MRPC (Dolan and Brockett, 2005), SST-2 (Socher et al., 2013), RTE (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), STS-B (Cer et al., 2017), WNLI (Levesque et al., 2012), QNLI (Rajpurkar et al., 2016), and QQP (Iyer et al., 2017).
- **OOD**: ChemProt (Kringelum et al., 2016), RCT (Dernoncourt and Lee, 2017), IMDB (Maas et al., 2011), and Amazon (He and McAuley, 2016). ChemProt is a manually annotated chemical-protein interaction dataset extracted from 5,031 abstractions. RCT is a dataset based on PubMed for sentence classification. IMDB provides 25,000 movie reviews for sentiment analysis. Amazon is a dataset containing product reviews from Amazon, annotated with user ratings.
- **KI**: FEVER (Thorne et al., 2018) and CommonsenseQA (CSQA) (Talmor et al., 2019). FEVER consists of 185,445 claims that correspond to Wikipedia articles and are classified as supported, refuted, and not enough information. CommonsenseQA consists of 12,247 questions with 5 choices and requires commonsense knowledge to predict the correct answers.

For Stage 1, we train the domain-adapter with unstructured knowledge related to the dataset following Section 3.1. The unstructured knowledge used is listed in Table 1. We also experiment with structured knowledge in Section 6.2. For Stage 2, we adopt the true few-shot setting following (Perez et al., 2021) to demonstrate the effectiveness of MixDA. For each dataset class, we randomly sample $K = 16$ examples from the original training set as the new training set, and another different $K = 16$ examples as the validation set. The original validation set will be used as the test set. The Pfeiffer adapter is used in Stage 2 unless stated otherwise.

4.2 Baselines

In our experiments, we use the following models as the main baselines. For convenience, we refer to them with the abbreviations in the parentheses later.

- **HOULSBY (HO)**: Houlsby adapter (Houlsby et al., 2019) plugged into the RoBERTa-large model for downstream tasks. Only adapter parameters are trained. It adds two adapter blocks consisting of bottleneck networks in each Transformer block.
- **PFEIFFER (PF)**: Pfeiffer adapter (Pfeiffer et al., 2020b) plugged into the RoBERTa-large model. This is similar to the Houlsby adapter, but with a different architecture. Pfeiffer adapter has only one adapter layer in each Transformer block, while Houlsby has two. Also, Pfeiffer makes minor tweaks in the adapter architecture, such as the layer norm and nonlinearity.
- **LORA (LO)**: LoRA (Hu et al., 2021) applied to the RoBERTa-large model. LoRA freezes the MLP modules and represents updates to the attention weights with two low-rank matrices, thus saving space.
- **PREFIX-TUNING (PT)**: Prefix-Tuning (Li and Liang, 2021) with the RoBERTa-large model. Prefix-Tuning trains a number of prompt embeddings for each task and pre-pends it before tokens.
- **FINE-TUNING (FT)**: Fine-tuning all of the parameters of the RoBERTa-large model on downstream tasks.

4.3 Evaluation Metrics

We adopt the Pearson correlation for STS-B since it is a regression task. The remaining are text classification tasks. Following Wang et al. (2018); Gururangan et al. (2020); Diao et al. (2021), we adopt macro-F1 for MRPC and QQP, and micro-F1 for others as evaluation metrics. Macro-F1 computes the F1 independently for each metric, while micro-F1 computes an average metric of all classes. To account for the instability of small datasets, we report the average performance and the standard deviation of 3 runs with different random seeds.

4.4 Implementation

We implement our RoBERTa-large model based on the Transformers library from HuggingFace[2]. The Houlsby adapter, the Pfeiffer adapter, and Prefix-Tuning are implemented based on the adapter-transformers library (Pfeiffer et al., 2020a). LoRA is implemented based on OpenDelta (Ding et al., 2022). During Stage 1, we train the domain-adapter with learning rate 1e-4, batch size 20, and weight decay 0.05. The knowledge loss factor λ is set to 0.5. We train the 7 and 11 layers of RoBERTa-large with domain-adapter in 10 epochs. In Stage 2, we use the Pfeiffer adapter as the default task-adapter and train 20 epochs. All the experiments are conducted on Nvidia 2080Ti GPUs. We find the best hyper-parameters through grid search and the best results are listed in Appendix A. The computation time can be found in Appendix B.

5 Experimental Results

We compare the performance of MixDA with our baselines on 15 datasets. First, we train the domain-adapter for each domain individually and then perform each task with its corresponding domain-adapter, which shows significant improvement over our baselines. Next, we plug in several domain-adapters trained on different domains parallelly to verify the scalability of our model.

5.1 Single Domain Adapter

Table 2 shows the performance of a single domain adapter compared with baselines. It is only trained on unstructured knowledge during Stage 1 in the following experiments. Results show that Mixture-of-Domain-Adapters outperforms our baselines in most datasets, with an average of 3.5% improvement over the best baseline adapter (i.e., Pfeiffer), and 3.3% over fine-tuning. Our method even outperforms fine-tuning in most datasets, despite far less training time and smaller parameter size. Over the datasets, MixDA shows the most significant improvement on ChemProt, with 6.9% over Pfeiffer and 2.7% over fine-tuning. One possible reason is that MixDA learns the necessary knowledge to

[2] https://github.com/huggingface/transformers

	Datasets	HO	PF	LO	PT	FT	MixDA
OOD	ChemProt	47.1±12.2	53.7±8.2	24.2±11.6	17.1±7.3	57.9±4.0	**60.6±4.9**
	RCT	25.2±2.6	21.9±2.5	18.5±3.7	24.4±3.7	21.0±3.2	**26.4±0.8**
	IMDB	56.0±5.7	55.4±5.6	43.7±7.7	53.3±2.6	46.3±13.7	**58.1±5.1**
	Amazon	48.8±3.2	49.7±1.4	51.5±3.9	52.7±2.8	51.7±6.2	**54.7±1.6**
	Avg.	44.3	45.2	34.5	36.9	44.2	**50.0**
ID	MNLI	37.2±0.3	35.7±0.1	34.8±1.5	35.4±0.0	35.3±0.2	**37.3±1.5**
	COLA	17.6±5.5	9.1±5.0	7.1±3.0	12.1±5.5	**21.3±1.7**	20.1±6.3
	MRPC	81.2±0.2	80.7±0.6	64.0±20.5	**81.6±0.5**	81.3±0.1	**81.6±0.5**
	SST2	54.7±3.6	53.3±1.9	50.5±1.0	52.5±1.2	54.8±1.4	**56.4±3.5**
	RTE	53.5±1.4	54.1±1.3	53.4±2.1	53.4±1.1	54.7±1.3	**54.9±1.5**
	STS-B	88.1±1.6	**90.6±0.1**	89.5±0.8	85.6±4.1	78.4±8.0	89.8±0.4
	WNLI	57.3±1.3	58.1±2.5	59.1±1.2	57.3±0.7	58.7±1.8	**60.1±1.8**
	QNLI	53.0±0.1	51.9±0.8	53.3±1.3	52.1±0.9	51.3±0.1	**54.8±1.8**
	QQP	54.7±0.6	55.2±1.0	53.0±2.0	55.3±0.3	55.3±0.3	**56.1±0.6**
	Avg.	55.3	54.3	51.6	53.9	54.6	**56.8**
KI	FEVER	20.2±4.3	27.4±7.5	22.6±10.6	31.1±3.6	**36.1±6.7**	32.6±9.4
	CSQA	27.3±0.7	34.1±8.7	20.3±10.9	29.6±4.2	29.6±3.0	**38.9±4.0**
	Avg.	23.8	30.8	21.5	30.4	32.9	**35.8**
	Avg.	48.1	48.7	43.0	46.2	48.9	**52.2**

Table 2: The overall performance of single MixDA and baselines on the downstream tasks. We use $K = 16$ (per class) for few-shot experiments. The best result for each dataset is made bold. We report mean and standard deviation over 3 runs with different random seeds.

	Amazon	IMDB	FEVER	WNLI	QQP	RTE	MRPC	**Avg.**
Pfeiffer	49.7±1.4	55.4±5.6	27.4±7.5	58.1±2.5	55.2±1.0	54.1±1.3	80.7±0.6	54.4
Single	**54.7±1.6**	**58.1±5.1**	32.6±9.4	**60.1±1.7**	56.1±0.6	**54.9±1.5**	**81.6±0.5**	**56.9**
Parallel	51.6±2.4	47.9±2.1	**34.5±0.5**	58.7±1.8	**57.8±3.5**	53.8±0.9	81.0±0.2	55.0

Table 3: The performance of parallel domain-adapters on the chosen downstream tasks. Parallel, Single, and Pfeiffer denote parallel domain-adapters, single domain-adapter, and vanilla RoBERTa + Pfeiffer, respectively. The best result for each dataset is made bold.

detect the chemical-protein interaction. For example, MixDA shows more familiarity with words associated with that field, such as "gefitinib" and "tyrosine kinase inhibitor". In contrast, MixDA falters on STS-B, falling behind Pfeiffer by 0.8%. This is because the knowledge in Stage 1 is not effectively utilized. STS-B consists of sentence pairs like "The cat sat on the mat" and "The cat did not sit on the mat", with little need for additional knowledge. Across the three task domains, MixDA has an average improvement of 4.8% over RoBERTa + Pfeiffer on out-of-domain tasks, 2.5% on in-domain tasks, and 5.0% on knowledge-intensive tasks. It shows that MixDA is not only effective for out-of-domain tasks and knowledge-intensive tasks that require additional knowledge but is helpful for general-domain language tasks as well, demonstrating its ability to excel at both in-domain and out-of-domain tasks (reliability).

5.2 Parallel Domain Adapters

In the previous section, we explored using a single domain-adapter for each downstream task. Next, we show the scalability of MixDA by using parallel domain-adapters and only train the MoA layer and task-adapters in Stage 2. The training process in Stage 2 follows the previous experiments. Table 3 shows the comparison across single domain-adapter, parallel domain-adapters, and RoBERTa + Pfeiffer on 7 datasets. On average, parallel domain-adapters show an improvement of 0.6% over vanilla RoBERTa + Pfeiffer, even though they fall behind the single domain adapter by 1.9%. This could be attributed to the MoA gate choosing the sub-optimal domain-adapter for some test data. Still, considering its improvement over Pfeiffer, the MoA gate chooses the correct domain-adapter in most cases. Therefore, MixDA demonstrates its scalability, allowing end users to train Stage 1 on different datasets and combine them later. Overall, in both single and parallel situations, MixDA significantly improves upon the vanilla RoBERTa + Pfeif-

Datasets	ChemProt	IMDB	MRPC	STS-B	CSQA	**Avg.**
MixDA	**60.6**±4.9	**58.1**±5.1	**81.6**±0.5	89.8±0.4	**38.8**±4.0	**65.8**
– MoA	55.7±1.8	49.8±0.0	80.9±0.5	88.4±0.4	28.3±1.3	60.6
– Old	54.3±6.5	41.4±3.6	78.7±3.7	**90.0±0.4**	27.1±0.3	58.3
– DA	21.2±4.7	56.8±3.9	81.0±0.5	80.8±2.4	27.4±0.5	53.4
AdapterFusion	47.7±0.1	54.4±2.0	78.0±1.5	90.3±0.3	25.0±1.7	59.1
K-Adapter	58.2±5.0	55.6±4.5	53.9±5.9	89.7±0.4	26.2±4.7	56.7
CPT	45.9±0.3	56.1±5.2	81.0±0.5	90.2±0.1	33.7±2.7	61.4

Table 4: Ablations of the MoA gate, old-domain knowledge, and the domain-adapter structure and comparisons with other adapter-based tuning methods. For **– Old**, we omit old-domain knowledge in Stage 1 training. For **– DA**, we remove the domain-adapter structure and conduct both stages of training only with Pfeiffer adapters. The best results for each dataset are made bold.

Datasets	MRPC	STS-B	FEVER	CSQA	**Avg.**
MixDA	81.6±0.5	89.8±0.4	20.2±4.3	38.8±4.0	57.6
+ ConceptNet	**81.7±0.3**	**90.1±0.1**	**30.5±3.1**	**40.0±0.2**	**60.6**

Table 5: The results of MixDA trained on structured and unstructured knowledge. **+ ConceptNet** stands for domain-adapters trained on both the unstructured knowledge and ConceptNet.

fer model with a small increase in model size. This is due to the ability of MixDA to capture knowledge and the MoA to select useful knowledge for downstream tasks.

6 Analysis

In this section, we analyze the respective contributions of each part of MixDA through detailed analysis, including the Stage 1 training, task-adapters in Stage 2, and the mixture-of-adapters gate.

6.1 Ablation Study

In this section, we conduct an ablation study to reveal the contributions of each part of the model. There are three variants: (1) We remove the MoA gate and choose the domain-adapter instead of the RoBERTa feed-forward layer (–MoA). (2) We exclude old-domain knowledge during Stage 1 (–Old). (3) To examine whether the training procedures, rather than the MixDA structure, contribute the most to our results, we conduct Stage 1 and Stage 2 training only with task-adapters (–DA). Table 4 shows the results of the ablation study. As expected, the average performance drops in all three settings. Without MoA gate, old-domain knowledge FFNs, and structure knowledge, it is observed a drop of 5.2%, 7.5%, and 12.4%, respectively, showing that the MoA gate, the old-domain knowledge, and the MixDA structure are all fundamental in the model. Relatively, the MoA has the smallest impact because the old-domain knowledge in Stage 1 can also help the model retain the knowledge in RoBERTa. The domain-adapter has the largest impact since it only stores domain knowledge and can keep it during Stage 2. In contrast, conducting Stage 1 and 2 training on the Pfeiffer adapter causes catastrophic forgetting.

6.2 Structured and Unstructured Knowledge

In Section 5, the MixDA is only trained on unstructured knowledge. As a comparison, we also train the domain adapter on ConceptNet, a structured knowledge dataset, and then attach both the unstructured and structured to our model and train the MoA layer and the task-adapter during Stage 2.

Table 5 shows the result of combining structured and unstructured knowledge in Stage 1. FEVER and CSQA, two knowledge-intensive tasks, have the greatest improvement: 10.3% for FEVER and 1.2% for CSQA. This is because ConceptNet stores commonsense knowledge that can help both tasks. Meanwhile, MRPC and STS-B also obtain improvement, showing that ConceptNet can benefit general language tasks as well. In conclusion, the experiment demonstrates the ability of MixDA to utilize structured knowledge, the extensibility of our model, and the possible benefits of structured knowledge.

6.3 Effectiveness of Task-Adapters

In most experiments of this paper, we adopt Pfeiffer as the task-adapter unless otherwise specified. In this section, we test the performance of MixDA combined with other kinds of task-adapters, including Houlsby, Prefix-Tuning, LoRA, and Pfeiffer. Table 6 gives the result of different task-adapters. Pfeiffer surpasses others by at least 6.3%. Even though Houlsby is on par with Pfeiffer, Pfeiffer only requires half the number of newly introduced

Datasets	ChemProt	IMDB	MRPC	STS-B	CSQA	**Avg.**
Houlsby	47.1$\pm$12.2	48.1$\pm$4.5	80.0$\pm$1.5	86.6$\pm$3.2	35.8$\pm$8.9	59.5
Prefix-Tuning	17.1$\pm$7.3	39.1$\pm$7.2	81.6$\pm$0.4	88.6$\pm$0.5	33.3$\pm$0.0	51.9
LoRA	19.5$\pm$11.1	36.1$\pm$4.1	81.2$\pm$0.0	86.7$\pm$1.4	20.3$\pm$10.9	48.8
Pfeiffer	**60.6**$\pm$**4.9**	**58.1**$\pm$**5.1**	**81.6**$\pm$**0.5**	**89.8**$\pm$**0.4**	**38.8**$\pm$**4.0**	**65.8**

Table 6: The results of MixDA combined with different kinds of task-adapters. By default, we use Pfeiffer in previous experiments.

parameters compared to Houlsby, making it the optimal choice of task-adapters in our experiment.

7 Conclusion

In this paper, we proposed MixDA, a mixture of adapters for domain adaptation. We first decouple the knowledge modules (i.e., FFNs) into the old-domain and domain-specific FFNs. Then we propose a two-stage adapter tuning strategy: first tuning the domain adapter on each domain and then tuning the task adapter on each task. Moreover, our model could be scaled to multiple domains easily with the introduction of the mixture-of-adapters gate. Empirically, MixDA achieved significant improvement over in-domain tasks, out-of-domain tasks, and knowledge-intensive tasks. Further analyses demonstrate the reliability, scalability, and efficiency of our method.

Limitations

Although MixDA achieves promising results on domain adaptation compared with baseline models, there are certain limitations. MixDA is a two-stage approach, which is not fully end-to-end. Our approach requires training a domain adapter and task adapter, respectively. In the future, we will explore the unifying domain and task adapters by merging them into one.

Acknowledgments

We thank the anonymous reviewers for their valuable suggestions. This work was supported by the General Research Fund (GRF) of Hong Kong (No. 16310222 and No. 16201320). Shizhe Diao and Ruijia Xu were supported by the Hong Kong Ph.D. Fellowship Scheme (HKPFS).

References

Abien Fred Agarap. 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly Available Clinical BERT Embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3606–3611.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Franck Dernoncourt and Ji Young Lee. 2017. PubMed 200k RCT: a dataset for sequential sentence classification in medical abstracts. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 308–313, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of

Deep Bidirectional Transformers for Language Understanding. *arXiv*.

Shizhe Diao, Jiaxin Bai, Yan Song, Tong Zhang, and Yonggang Wang. 2020. Zen: Pre-training chinese text encoder enhanced by n-gram representations. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4729–4740.

Shizhe Diao, Ruijia Xu, Hongjin Su, Yilei Jiang, Yan Song, and Tong Zhang. 2021. Taming pre-trained language models with n-gram representations for low-resource domain adaptation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3336–3349.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. 2022. Delta tuning: A comprehensive study of parameter efficient methods for pre-trained language models. *arXiv preprint arXiv:2203.06904*.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

Keyur Faldu, Amit Sheth, Prashant Kikani, and Hemang Akbari. 2021. Ki-bert: Infusing knowledge context for better language and domain understanding. *arXiv preprint arXiv:2104.08145*.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague. Association for Computational Linguistics.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A Smith, and Luke Zettlemoyer. 2022. Demix layers: Disentangling domains for modular language modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5557–5576.

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. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360.

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, volume 7.

Bin He, Di Zhou, Jinghui Xiao, Xin Jiang, Qun Liu, Nicholas Jing Yuan, and Tong Xu. 2020. Bert-mk: Integrating graph contextualized knowledge into pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2281–2290.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021a. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021b. DEBERTA: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 507–517, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. *arXiv*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. *arXiv*.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. 2019. ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission. *arXiv preprint arXiv:1904.05342*.

Shankar Iyer, Nikhil Dandekar, Kornél Csernai, et al. 2017. First quora dataset release: Question pairs. *data. quora. com*.

Zixuan Ke, Haowei Lin, Yijia Shao, Hu Xu, Lei Shu, and Bing Liu. 2022a. Continual training of language models for few-shot learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10205–10216, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zixuan Ke, Yijia Shao, Haowei Lin, Hu Xu, Lei Shu, and Bing Liu. 2022b. Adapting a language model while preserving its general knowledge. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10177–10188.

Jens Kringelum, Sonny Kim Kjaerulff, Søren Brunak, Ole Lund, Tudor I. Oprea, and Olivier Taboureau. 2016. ChemProt-3.0: a global chemical biology diseases mapping. *Database*, 2016:bav123.

Anne Lauscher, Ivan Vulić, Edoardo Maria Ponti, Anna Korhonen, and Goran Glavaš. 2019. Informing unsupervised pretraining with external linguistic knowledge.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: A Pre-Trained Biomedical Language Representation Model for Biomedical Text Mining. *Bioinformatics*, 36(4):1234–1240.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*.

Yoav Levine, Barak Lenz, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2020. Sensebert: Driving some sense into bert. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4656–4667.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pages 6265–6274. PMLR.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. *arXiv*.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. Kagnet: Knowledge-aware graph networks for commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2829–2839.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ArXiv preprint*, abs/2107.13586.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2901–2908.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. GPT Understands, Too. *ArXiv preprint*, abs/2103.10385.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv*.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization.

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.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and Editing Factual Associations in GPT. *arXiv*.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *Advances in Neural Information Processing Systems*, 34:11054–11070.

Matthew E Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. Adapterfusion: Non-destructive task composition for transfer learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 487–503.

Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020a. Adapterhub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020b. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. 2021. Hash layers for large sparse models. *Advances in Neural Information Processing Systems*, 34:17555–17566.

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitry Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. In *International Conference on Learning Representations*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2016. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. *arXiv*.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *NAACL-HLT*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuan-Jing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021a. K-adapter: Infusing knowledge into pre-trained models with adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1405–1418.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021b. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. *arXiv preprint arXiv:2210.17451*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Wenhan Xiong, Jingfei Du, William Yang Wang, and Veselin Stoyanov. 2020. Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model. In *International Conference on Learning Representations*.

An Yang, Junyang Lin, Rui Men, Chang Zhou, Le Jiang, Xianyan Jia, Ang Wang, Jie Zhang, Jiamang Wang, Yong Li, et al. 2021. M6-t: Exploring sparse expert models and beyond. *arXiv preprint arXiv:2105.15082*.

Ze Yang, Wei Wu, Can Xu, Xinnian Liang, Jiaqi Bai, Liran Wang, Wei Wang, and Zhoujun Li. 2020. StyleDGPT: Stylized Response Generation with Pre-trained Language Models. *ACL Anthology*, pages 1548–1559.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, et al. 2019. Learning and evaluating general linguistic intelligence. *arXiv preprint arXiv:1901.11373*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and William B Dolan. 2020. DIALOGPT: Large-Scale Generative Pre-training for Conversational Response Generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. *arXiv*.

Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Ruofei Zhang, Jianfeng Gao, and Tuo Zhao. 2021. Taming sparsely activated transformer with stochastic experts. In *International Conference on Learning Representations*.

A Experimental Setup

Our domain adapter has a reduction factor of 16, consisting of two linear layers 4096×256 and 256×1024 (1.31M parameters). With each domain adapter also comes a MoA gate which has an FFN with 4096×2 (number of MixDAs) parameters. Since domain adapters are placed in Layers 7 and 11, they have 2.6M parameters in total. Therefore, the domain adapters (excluding task-adapters) only add 0.7% additional parameters to RoBERTa-large.

We preprocess the unstructured data in Stage 1 similar to the masked language model directive in RoBERTa. From the text, we choose 15% of tokens uniformly to perform possible alterations. In those tokens, 85% are replaced with `<mask>`, 10% are left unchanged, and 5% are replaced with a random token. The preprocessing is implemented with `DataCollatorForLanguageModeling` in Huggingface Transformers. In Stage 2, we use few-shot setting with $K = 16$. For each class of the dataset, we randomly select 16 examples before run.

In Stages 1 and 2, we use a linear weight scheduler. All the models are optimized by AdamW (Loshchilov and Hutter, 2017) with weight decay 0.05. The best hyperparameters for Stage 2 are found with grid search, with batch size $\{2, 4, 8, 16\}$ and learning rate $\{5e-5, 1e-4, 5e-4\}$. The details can be found in Tables 7 and 8.

B Computational Budget

Stage 1 training takes relatively longer time, while Stage 2 is fast due to the few-shot setting. The training time of Stage 1 is proportional to the number of tokens. For reference, with 4 Nvidia RTX 2080Ti, Stage 1 training for Biomed (33.6M tokens) takes ~45min per epoch, and training for Review (7.4M tokens) takes ~5min per epoch. Stage 2 training is generally fast: The 20-epoch training process usually takes less than 5min with 4 Nvidia RTX 2080Ti.

C Details of Datasets

We conduct experiments on three types of datasets: **in-domain (ID)** tasks that require general-domain knowledge; **out-of-domain (OOD)** tasks that require domain-specific knowledge; **knowledge-intensive (KI)** tasks that require commonsense knowledge.

For in-domain tasks, we evaluate our model on the GLUE Benchmark (Wang et al., 2018). It includes MNLI (Williams et al., 2017), CoLA (Warstadt et al., 2019), MRPC (Dolan and Brockett, 2005), SST-2 (Socher et al., 2013), RTE (Dagan et al., 2005; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), STS-B (Cer et al., 2017), WNLI (Levesque et al., 2012), QNLI (Rajpurkar et al., 2016), and QQP (Iyer et al., 2017). They are all single-sentence or sentence pair classification tasks except STS-B, which is a regression task.

We also evaluate our model on several out-of-domain tasks, including ChemProt (Kringelum et al., 2016), RCT (Dernoncourt and Lee, 2017), IMDB (Maas et al., 2011), and Amazon (He and McAuley, 2016). ChemProt is a manually annotated chemical-protein interaction dataset extracted from 5,031 abstractions. RCT is a dataset based on PubMed for sentence classification. IMDB provides 25,000 movie reviews for sentiment analysis. Amazon is a dataset containing product reviews from Amazon, annotated with user ratings.

For knowledge-intensive tasks, we evaluate our model on **FEVER** (Thorne et al., 2018) and **CommonsenseQA** (CSQA) (Talmor et al., 2019). FEVER consists of 185,445 claims that correspond to Wikipedia articles and are classified as supported, refuted, and not enough information. CommonsenseQA consists of 12,247 questions with 5 choices, each of which requires commonsense knowledge to predict the correct answers.

For Stage 1, we train domain-adapters with unstructured knowledge related to the dataset following Section 3.1. The unstructured knowledge used is listed in Table 1. We also experiment with structured knowledge in Section 6.2. For Stage 2, we adopt the true few-shot setting following (Perez et al., 2021) to demonstrate the effectiveness of MixDA. For each class of each dataset, we randomly sample $K = 16$ examples from the original training set as the new training set, and another different $K = 16$ examples as the validation set. The original validation set will be used as the test set. The Pfeiffer adapter is used in Stage 2 unless stated otherwise.

Config	Value
Optimizer	AdamW
Learning rate	1e-4
Weight decay	0.05
Optimizer momentum	β_1, β_2=$0.9, 0.999$
Batch size	{2, 5}
Learning rate schedule	linear decay
training epochs	10

Table 7: Stage 1 training: experimental setup.

Config	Value
Optimizer	AdamW
Learning rate	{5e-5, 1e-4, 5e-4}
Weight decay	0.05
Optimizer momentum	β_1, β_2=$0.9, 0.999$
Batch size	{2, 4, 8, 16}
Learning rate schedule	linear decay
warmup epochs	2
training epochs	20

Table 8: Stage 2 training: experimental setup.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Refer to Section "Limitations" at the end of paper.

☒ A2. Did you discuss any potential risks of your work?
This paper mainly focuses on improving performance on downstream tasks with external knowledge. We have not found risks of the potential use cases.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Refer to Abstract and Introduction (Section 1).

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

B ☑ Did you use or create scientific artifacts?

Refer to Section 4.1 and Appendix C.

☑ B1. Did you cite the creators of artifacts you used?
Refer to Section 4.1 and Appendix C.

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
We use publicly available datasets. Non-commercial and academic use of them are approved by their respective authors.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
We use publicly available datasets. Non-commercial and academic use of them are approved by their respective authors.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
We use publicly available datasets.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Refer to Section 4.1 and Appendix C.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Refer to Section 4.1, Table 1, and Appendix C.

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

C ☑ **Did you run computational experiments?**

Refer to Sections 5 and 6.

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

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Refer to Appendix A.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Refer to Sections 5 and 6.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Refer to Section 4.4.

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

Left blank.

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

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

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

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

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