UDAPTER - Efficient Domain Adaptation Using Adapters

Bhavitvya Malik[*α], **Abhinav Ramesh Kashyap** [*βγ],
Min-Yen Kan[β], **Soujanya Poria**[♣]
[α] The University of Edinburgh, Edinburgh
[β] National University of Singapore, Singapore
[γ] ASUS Intelligent Cloud Services (AICS), Singapore
[♣] DeCLaRe Lab, Singapore University of Technology and Design, Singapore
b.malik-1@sms.ed.ac.uk, abhinav_kashyap@asus.com, kanmy@comp.nus.edu.sg,
sporia@sutd.edu.sg

Abstract

We propose two methods to make unsupervised domain adaptation (UDA) more parameter efficient using adapters, small bottleneck layers interspersed with every layer of the large-scale pre-trained language model (PLM). The first method deconstructs UDA into a two-step process: first by adding a *domain adapter* to learn domain-invariant information and then by adding a *task adapter* that uses domain-invariant information to learn task representations in the source domain. The second method jointly learns a supervised classifier while reducing the divergence measure. Compared to strong baselines, our simple methods perform well in natural language inference (MNLI) and the cross-domain sentiment classification task. We even outperform unsupervised domain adaptation methods such as DANN (Ganin et al., 2016) and DSN (Bousmalis et al., 2016) in sentiment classification, and we are within 0.85% F1 for natural language inference task, by fine-tuning only a fraction of the full model parameters. We release our code at *https://github.com/declare-lab/domadapter*.

1 Introduction

Fine-tuning pretrained language models (PLM) is the predominant method for improving NLP tasks such as sentiment analysis, natural language inference, and other language understanding tasks (Wang et al., 2018). However, fine-tuning forces us to modify all the parameters of the model and store one copy of the model for one task. Given the large size of current PLMs, this can be expensive. Furthermore, fine-tuning needs large-scale data to be effective and is unstable when using different seeds (Han et al., 2021).

A new approach to alleviate this is parameter-efficient fine-tuning – freezing the PLM parameters and fine-tuning only a small fraction of the parameters. Fine-tuning with adapters (Houlsby et al., 2019) is one of these methods in which small additional layers are tuned within each PLM layer. Fine-tuning with adapters has many advantages: performance comparable to full fine-tuning (He et al., 2021a), and robustness to different seeds and adversarial examples (Han et al., 2021).

Unsupervised domain adaptation (UDA) aims to adapt models to new domains and considers situations where labeled data are available only in the source domain and unlabeled data are available in the target domain. UDA methods in general have two components: The first reduces the divergence between the source and target domains, and the second reduces the loss corresponding to a particular task (Ramesh Kashyap et al., 2021a). However, they fine-tune a large number of parameters and are susceptible to catastrophic forgetting. Adapters (Houlsby et al., 2019) can help solve these problems. However, the benefits of using adapters fine-tuning for domain adaptation have been mostly overlooked. *How well can adapter fine-tuning perform across different domains and can we make domain adaptation more efficient?* In this work, we answer these questions and propose models to perform domain adaptation using adapters.

Adapters are known to perform well in low-resource scenarios where a small amount of supervised data is available in a new domain or language (He et al., 2021b; Pfeiffer et al., 2020b). In this work, using the principles of UDA, we propose to make domain adaptation more effective using unsupervised data from the target domain. We introduce two methods that we collectively call the **U** nsupervised **D** omain **A** daptation method using ada**pters** (UDAPTER). The first method is a two-step process: First, we learn *domain adapters* – where we use a divergence measure to bring two probabilistic distributions closer together. This helps us to learn representations that are independent of the

[*]The first two authors contributed equally.

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 2249–2263
May 2-6, 2023

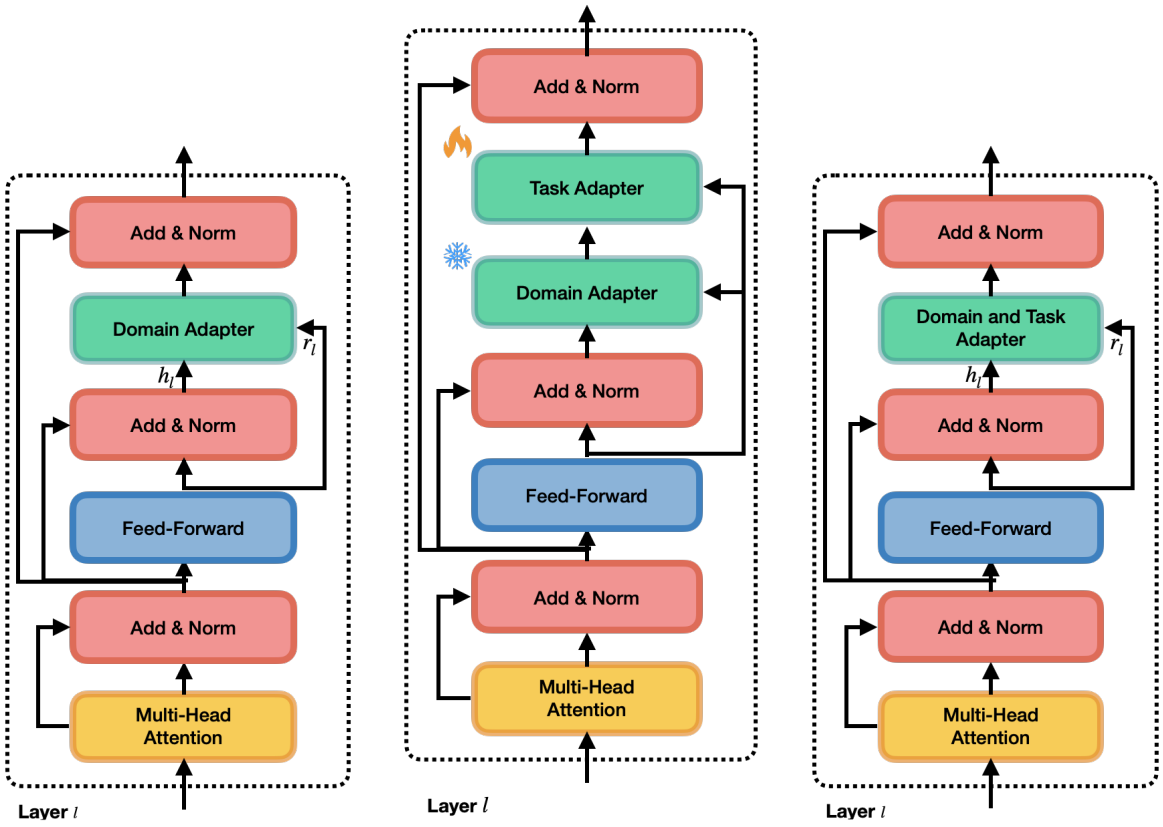


Figure 1: UDAPTER for a transformer layer l uses principles from unsupervised domain adaptation to make domain adaptation more parameter efficient. (a) The first method TS-DT- 🔌 trains a Domain Adapter that reduces the marginal distribution between the domains (b) The task adapter is stacked on top of the domain adapter. and trained on an end task like sentiment analysis or natural language inference. The domain adapter is frozen during training. (c) The second method JOINT-DT- 🔌 reduces the domain divergence and the task loss jointly.

domain from which they come. Second, we use the domain-invariant information learned as input to another task adapter that learns to perform an NLP task using labeled data from the source domain. We combine the two adapters by stacking them. The second method adds a single adapter without stacking, where we simultaneously reduce the divergence between domains and learn the task in the source domain.

Domain Adversarial Neural Networks (DANN) and Domain Separation Networks (DSN) are the most common methods for unsupervised domain adaptation in NLP (Ramesh Kashyap et al., 2021a). We compare our proposed methods with these strong baselines that fine-tune all model parameters, on Amazon (Blitzer et al., 2007) and the MNLI dataset (Williams et al., 2018) consisting of five domains each. UDAPTER performs better than all baselines. It achieves competitive performance compared to UDA methods by fine-tuning only a fraction of the parameters. In an era where large resources are spent to further pretrain language models on large amounts of unsupervised data to achieve domain adaptation (Gururangan et al., 2020), it is necessary to provide cheaper, faster solutions.

2 Method

Setup. We consider an NLP task (sentiment analysis) consisting of data $\mathcal{X}$ and labels $\mathcal{Y}$ (positive, negative). There exist two different distributions, called the source domain $\mathcal{D}_\mathcal{S}$ and the target domain $\mathcal{D}_\mathcal{T}$ over $\mathcal{X} \times \mathcal{Y}$. Unsupervised domain adaptation (UDA) consists of a model $\mathcal{C}$ that receives labeled input samples $\mathcal{X}_\mathcal{S} : (x_s, y_s)_{s=1}^{n_s} \sim \mathcal{D}_\mathcal{S}$ and unlabeled input $\mathcal{X}_\mathcal{T} : (x_t)_{t=1}^{n_t} \sim \mathcal{D}_\mathcal{T}$. The goal of UDA is to learn a model $\mathcal{C}$ such that we perform well in the NLP task for the target domain $\mathcal{D}_\mathcal{T}$.

The popular method in UDA is to learn representations that are invariant in the input domain and still have sufficient power to perform well in the source domain (Ganin et al., 2016; Bousmalis

et al., 2016). Then according to the theory of domain divergence (Ben-David et al., 2010) shows that the error in the target domain is bounded by the error in the source domain and the divergence. The unsupervised domain adaptation method thus consists of two components: the reduction of the divergence measure and a classifier for the source domain. A new classifier must be learned for every pair of source-target domains, and the method fine-tunes a large number of parameters.

UDAPTER makes unsupervised domain adaptation more parameter efficient (cf. § 2.1, § 2.2) using adapters. We follow the framework proposed by Houlsby et al. (2019) where small bottleneck layers are added to the transformer layers, fine-tuning only the adapter parameters while keeping the other parameters frozen, and propose the following.

2.1 Two-Step Domain and Task Adapters

Domain Adapters. To learn domain-invariant representations, we first train a domain adapter. The adapter architecture follows the work of Pfeiffer et al. (2021), which consists of a simple down-projection followed by an up-projection. In a transformer layer l, let h_l be the hidden representation of the layer **Add & Norm** and let r_l be the representation of the layer **Feed-Forward** (Figure 1a), then the adapter makes the following transformation and calculates a new hidden representation.

$$dom_l = W_{up} \cdot f(W_{down} \cdot h_l) + r_l \quad (1)$$

where f is a nonlinear function such as RELU, $W_{down} \in \mathbb{R}^{h \times d}$ projects the hidden representations down to a lower dimension, $W_{up} \in \mathbb{R}^{d \times h}$ projects them back to a higher dimension, and $d \ll h$. We pass a sample from the source domain $(x_s^{src}) \sim \mathcal{D}_S$ and a sample from the target domain $(x_t^{trg}) \sim \mathcal{D}_T$ through the adapters in layer l and obtain their representations h_l^{src} and h_l^{trg}, respectively. We then reduce the divergence between these representations.

$$\Delta_l = div(dom_l^{src}, dom_l^{trg}) \quad (2)$$

Here $div(\cdot)$ is the divergence function such as the correlation alignment (CORAL) (Sun et al., 2016), the central moment discrepancy (CMD) (Zellinger et al., 2017) or the multi-kernel maximum mean discrepancy (MK-MMD) (Gretton et al., 2012; Bousmalis et al., 2016). In this work, we use MK-MMD for all of our experiments, since it performed the best[1]. Similar ideas are used to adapt representations in computer vision models (Long et al., 2019; Sun and Saenko, 2016). The final divergence loss considers all L layers.

$$\mathcal{L}_{div} = \sum_{l=1}^{L} \Delta_l \quad (3)$$

Task Adapters. Task adapters are stacked with frozen domain adapters. We pass the representations dom_l from the previous step and the supervised data from the source domain $(x_s^{src}, y_s^{src}) \sim \mathcal{D}_S$. Task adapters have the same architecture as domain adapters and perform the following.

$$task_l = W_{up} \cdot f(W_{down} \cdot dom_l^{src}) + r_l \quad (4)$$

The goal of these task adapters is to learn representations that are task-specific. Only task adapters are updated when training on the end task (sentiment classification, natural language inference) and all other parameters, including domain adapters, are frozen. Regular cross-entropy loss is reduced during training of task adapters.

$$\mathcal{L}_{task} = softmax\_ce(W_{task} \cdot h_L) \quad (5)$$

h_L is the hidden representations of the last layer of the transformer, $W_{task} \in \mathbb{R}^{h * |\mathcal{Y}|}$ where $|\mathcal{Y}|$ is the number of classes, and $softmax\_ce$ is the softmax followed by cross-entropy. This two-step process deconstructs UDA methods with a domain adapter and a task adapter. This affords composability, where task adapters can be reused for different pairs of domains (§ 3.4). However, domain and task representations can be learned jointly, as explored in the next section.

Training Process. Given a source-target domain adaptation scenario, we first train the domain adapter and save their weights. We then stack the task adapter with the domain adapter, which is trained using the supervised data from the source domain. When training the task adapter, the domain adapter is frozen. During inference, we stack the domain and task adapter.

2.2 Joint Domain Task Adapters

This method adds a single adapter that performs the reduction of the divergence measure and learns

[1]We also tried using CMD and CORAL and our systems performed similarly to MK-MMD

Dataset	Train	Dev	Test
MNLI	69,600	7,730	1,940
AMAZON	1,440	160	400

Table 1: Dataset statistics, showing number of train, dev, and test instances per domain.

task representations jointly. For a given supervised sample from the source domain $(x_s^{src}, y_s^{src}) \sim \mathcal{D}_\mathcal{S}$ and an unsupervised sample $(x_t^{trg}) \sim \mathcal{D}_\mathcal{T}$, let h_l^{src}, h_l^{trg} be the hidden representations of the adapters for x_s^{src} and x_t^{trg} for layer l. We reduce the following joint loss:

$$\mathcal{L} = \lambda \cdot \mathcal{L}_{task} + (1 - \lambda) \cdot \mathcal{L}_{div} \quad (6)$$

Here $\mathcal{L}_{task}$ is the task loss on the source domain supervised samples, λ is the adaptation factor.

Reducing divergence along with cross-entropy loss beyond a certain point makes training unstable and does not contribute to increased performance. Following (Ganin et al., 2016) we suppress the noisy signal from the divergence function as training progresses and gradually change λ from 0 to 1 to reduce the contribution of divergence loss using the following schedule ($\gamma = 10$ for all of our experiments):

$$\lambda = \frac{2}{1 + \exp\left(-\gamma \cdot p\right)} - 1 \quad (7)$$

Similar methods have been proposed to adapt models to other domains by Long et al. (2019) and Wu et al. (2022). Compared to the two-step process introduced earlier (§ 2.2), we need to properly control the losses to obtain optimal results and also this method does not offer composability (§ 3.4).

3 Experiments

3.1 Datasets

We evaluate our approach on two representative datasets with different tasks, both in English. Table 1 shows the details of the datasets. Every dataset has 5 domains, and we consider each domain with every other domain which results in 20 domain adaptation scenarios for every dataset, 120 experiments per method, and 1900+ experiments.

AMAZON: Multi Domain Sentiment Analysis Dataset (Blitzer et al., 2007) that contains Amazon product reviews for five different types of products (domains): Apparel (A), Baby (BA), Books (BO), Camera_Photo (C), and Movie Reviews (MR). Each review is labeled as positive or negative. We follow the setup in (Ramesh Kashyap et al., 2021a)

MNLI: The Multigenre Natural Language Inference (MNLI) corpus (Williams et al., 2018) contains hypothesis–premise pairs covering a variety of genres: Travel (TR), fiction (F), telephone (TE), government (G), and slate (S). Each pair of sentences is labeled Entailment, Neutral, or Contradiction. The train and validation data set are taken from the train set by sampling 90% and 10% samples, respectively. We use the MNLI-matched validation set as our test set.

3.2 Baseline Methods

Fully supervised. *Fine-tune (* 🔥 *)*: Fine-tunes a language model using labeled data from the target domain. Serves as an upper bound of performance.

Unsupervised Domain Adaptation (UDA). *Domain Adversarial Neural Networks* (DANN): An unsupervised domain adaptation method (Ganin et al., 2016) that learns domain-invariant information by minimizing task loss and maximizing domain confusion loss with the help of gradient reversal layers. *Domain Separation Networks:* (DSN) (Bousmalis et al., 2016) improves DANN, with additional losses to preserve domain-specific information along with the extraction of domain-invariant information. `bert-base-uncased` serves as a feature extractor for both methods.

Adapter Based. DANN *Adapter* (DANN- 🔌): Similar to DANN, but we insert trainable adapter modules into every layer of a PLM. DANN *Adapter with Multiple Classifiers* (DANN- 🔌 -MC): Unlike DANN- 🔌 which involves a single task and domain classifier, here a task and domain classifier are added to each of the last 3 layers of a PLM. The representation of the last layers of a PLM is domain variant (Ramesh Kashyap et al., 2021b), and this model obtains domain-invariant information[2] (vi) Task adapter (TASK- 🔌): Adapter fine-tuning (Pfeiffer et al., 2020a) where adapters are fine-tuned in the labeled source domain and tested in the target domain. (vii) Two-step Domain and Task Adapter (TS-DT- 🔌): This work, where we first train a domain adapter that reduces the probabilistic divergence between two domains and then fine-tunes a task adapter by stacking. (viii) Joint

[2]We tried adding classifiers incrementally to the last few layers. Adding it to the last 3 layers performed the best.

Src →Trg	Fully Supervised	Unsupervised Domain Adaptation		Adapter Based				
		DANN	DSN	DANN-	DANN- -MC	TASK-	TS-DT-	JOINT-DT-
A → BA	87.52 (1.96)	85.57 (3.72)	89.90 (0.26)	86.46 (0.26)	**88.74 (0.64)**	87.03 (0.26)	88.24 (0.76)	**88.74 (0.13)**
A → BO	86.67 (1.06)	36.48 (0.45)	84.47 (0.99)	78.41 (1.14)	83.36 (0.43)	84.15 (1.10)	84.22 (0.76)	**84.96 (0.28)**
A → C	91.62 (0.37)	57.51 (13.32)	88.56 (0.81)	87.31 (0.39)	88.75 (0.69)	**89.67 (0.32)**	88.76 (1.32)	89.39 (0.23)
A → MR	82.08 (0.78)	35.23 (1.99)	78.08 (0.46)	75.54 (0.63)	76.60 (1.06)	76.63 (0.92)	77.39 (0.13)	**77.63 (0.71)**
BA → A	89.12 (0.38)	77.52 (11.25)	87.46 (1.83)	87.72 (1.85)	88.47 (0.72)	88.33 (1.10)	89.55 (0.10)	**89.70 (0.23)**
BA → BO	86.67 (1.06)	43.45 (8.96)	82.19 (3.70)	82.89 (3.08)	83.86 (0.41)	84.61 (0.39)	84.38 (0.61)	**85.01 (0.60**
BA → C	91.62 (0.37)	47.58 (7.65)	89.68 (0.71)	86.63 (0.53)	88.73 (0.42)	**90.63 (0.33)**	87.46 (0.88)	88.64 (0.30)
BA → MR	82.08 (0.78)	50.63 (7.43)	77.88 (0.38)	74.48 (1.79)	78.07 (0.34)	78.74 (0.35)	**79.42 (0.44)**	78.44 (0.70)
BO → A	89.12 (0.38)	37.40 (1.90)	88.20 (0.51)	85.90 (0.12)	85.91 (0.25)	85.03 (0.36)	84.79 (0.75)	**87.46 (0.27)**
BO → BA	87.52 (1.96)	54.33 (12.49)	88.56 (0.44)	82.06 (1.15)	84.27 (0.11)	86.50 (0.39)	**86.84 (0.48)**	86.41 (0.79)
BO → C	91.62 (0.37)	39.43 (0.49)	88.58 (1.01)	86.94 (0.83)	87.40 (0.44)	88.44 (0.53)	87.86 (0.61)	**88.53 (0.43)**
BO → MR	82.08 (0.78)	54.23 (13.94)	79.07 (1.01)	76.19 (0.89)	79.44 (0.86)	79.44 (0.95)	**80.52 (0.61)**	78.91 (0.38)
C → A	89.12 (0.38)	60.93 (3.78)	89.76 (0.76)	87.02 (1.86)	86.63 (0.29)	87.74 (1.18)	88.53 (0.42)	**88.92 (0.44)**
C → BA	87.52 (1.96)	77.29 (3.61)	89.42 (0.70)	88.10 (1.13)	89.14 (0.30)	81.71 (2.72)	**89.72 (0.43)**	89.32 (0.42)
C → BO	86.67 (1.06)	38.21 (1.40)	85.56 (0.62)	81.18 (2.07)	83.61 (0.67)	80.55 (0.81)	84.14 (0.52)	**85.42 (0.70)**
C → MR	82.08 (0.78)	35.08 (1.94)	76.13 (0.54)	64.99 (5.91)	**74.22 (0.31)**	69.53 (1.24)	73.22 (0.48)	73.50 (0.84)
MR → A	89.12 (0.38)	37.07 (4.16)	82.64 (2.17)	81.05 (1.15)	79.56 (0.53)	82.45 (1.43)	81.93 (0.47)	**84.41 (0.43)**
MR → BA	87.52 (1.96)	38.76 (4.17)	80.59 (2.18)	77.95 (1.46)	79.33 (0.43)	81.70 (1.22)	84.28 (0.41)	**84.91 (0.36)**
MR → BO	86.67 (1.06)	42.07 (4.86)	85.13 (0.83)	82.83 (0.62)	**84.90 (1.29)**	**84.90 (0.23)**	84.47 (0.80)	84.45 (0.31)
MR → C	91.62 (0.37)	36.92 (1.86)	86.56 (0.63)	84.58 (0.46)	82.53 (0.92)	86.68 (0.65)	86.25 (0.38)	**88.37 (0.11)**
Avg	87.40 (0.91)	49.28 (5.47)	84.92 (1.03)	81.91 (1.37)	83.68 (0.50)	83.72 (0.88)	84.60 (0.57)	**85.16 (0.43)**

Table 2: F1 scores for AMAZON dataset. We report mean and standard deviation of 3 runs. The five domains are Apparel (A), Baby (BA), Books (BO), Camera_Photo (C) and Movie Reviews (MR). On average, our method outperforms all baselines. Our methods are competitive with fully unsupervised domain adaptation methods.

Domain Task Adapter (JOINT-DT-) - We train a single adapter that reduces the domain and task loss jointly. For all adapter-based experiments, the PLM is frozen, and only adapter modules are trained.

Since we use adapters, we only consider other adapter based baselines and omit other methods such as Prefix-tuning (Lester et al., 2021). Also, (Zhang et al., 2021) target multidomain adaptation and use data from all the domains during training unlike our method and is not a fair comparison.

Implementation Details and Evaluation. For our experiments, we use `bert-base-uncased` (Devlin et al., 2019) available in the HuggingFace Transformers library (Wolf et al., 2020) as our backbone. Adapter implementations are from AdapterHub (Pfeiffer et al., 2020a). We follow (Pfeiffer et al., 2021) and add only one bottleneck layer after the feedforward layer.

We use the AdamW optimizer and a learning rate of $1e-4$ for all our adapter-based training and $2e-5$ otherwise. Only for the smaller AMAZON dataset, we used an adapter bottleneck size (reduction factor) of 32. For all other adapter-based experiments and datasets, we use the default adapter bottleneck size of 16. We performed experiments on three different seeds. We report the mean and standard deviation of the F1 scores . For DANN we use 0.04 as our λ and for DSN we use 0.1, 0.1, and 0.3 as our weights for three losses: reconstruction, similarity, and difference respectively. We avoid extensive hyperparameter tuning per domain adaptation scenario for efficiency.

3.3 Results

From Table 2 and Table 3 our methods TS-DT- and JOINT-DT- perform well in both AMAZON and MNLI. We find that fine-tuning the task adapter (TASK-) is a strong baseline and, compared to it, we perform well in 17/20 domain adaptation scenarios in AMAZON (largest increase of 8 points for C → BA) and 19/20 domain adaptation scenarios in MNLI (largest increase of 2.2 for F → TE). One possible explanation of scenarios where our method finds the largest increase is the proximity of the two domains. The overlap in vocabularies (Figure 9 in Appendix) between C →BA in AMAZON and F → TE in MNLI is high, and our method takes advantage of learning domain-invariant information that can be used for efficient domain transfer. Our methods for learning domain-invariant information are necessary to achieve good domain adaptation.

UDAPTER is comparable to UDA methods. Compared to UDA methods where all parameters of the backbone model are fine-tuned, we perform close to them on average. JOINT-DT- performs better than DSN by 0.2% in AMAZON. We are within 0.85% in MNLI compared to DSN. Training DANN is highly unstable and produces varied re-

	Fully Supervised	Unsupervised Domain Adaptation		Adapter Based				
Src →Trg	🔥	DANN	DSN	DANN-🔌	DANN-🔌-MC	TASK-🔌	TS-DT-🔌	JOINT-DT-🔌
F →S	74.09 (0.40)	73.68 (0.21)	72.36 (0.17)	70.96 (0.03)	62.40 (4.79)	72.36 (0.36)	**73.46 (0.34)**	72.30 (0.26)
F →G	82.19 (0.12)	79.17 (0.25)	79.79 (0.21)	78.73 (0.43)	77.23 (0.33)	79.00 (0.46)	78.65 (0.25)	**79.79 (0.22)**
F →TE	78.41 (0.66)	73.72 (0.81)	75.07 (0.32)	70.89 (0.74)	71.68 (0.59)	70.83 (0.54)	**73.05 (0.70)**	71.59 (0.78)
F→TR	81.81 (0.20)	76.99 (0.19)	76.82 (0.50)	74.42 (0.18)	75.09 (0.05)	75.85 (0.19)	76.75 (0.80)	**77.07 (0.26)**
S →F	78.59 (0.34)	75.91 (0.23)	76.62 (0.38)	73.89 (0.61)	73.47 (0.28)	75.25 (0.19)	**75.52 (0.89)**	75.35 (0.56)
S →G	82.19 (0.12)	80.91 (0.46)	81.27 (0.23)	79.99 (0.36)	79.16 (0.10)	80.76 (0.40)	**81.65 (0.11)**	80.94 (0.30)
S →TE	78.41 (0.66)	74.32 (0.57)	74.27 (0.48)	72.29 (0.57)	71.89 (0.07)	72.66 (0.79)	**74.09 (0.30)**	73.38 (0.63)
S →TR	81.81 (0.20)	76.81 (0.35)	78.17 (0.20)	75.58 (0.54)	75.77 (0.39)	76.16 (0.22)	**77.31 (0.60)**	77.16 (0.18)
G →F	78.59 (0.34)	73.41 (0.73)	72.62 (0.37)	71.57 (0.68)	70.34 (0.73)	72.66 (0.31)	72.66 (0.56)	**73.56 (0.23)**
G →S	74.09 (0.40)	72.51 (0.10)	71.93 (0.25)	70.17 (0.64)	69.49 (0.40)	71.11 (0.38)	71.14 (0.21)	**71.36 (0.04)**
G →TE	78.41 (0.66)	71.52 (0.13)	72.90 (0.39)	69.45 (0.96)	68.67 (0.17)	71.40 (0.30)	71.53 (1.04)	**71.99 (0.67)**
G →TR	81.81 (0.20)	77.42 (0.54)	77.80 (0.42)	74.35 (0.22)	74.04 (0.51)	76.29 (0.10)	76.16 (0.34)	**76.79 (0.59)**
TE →F	78.59 (0.34)	75.07 (0.08)	75.17 (0.35)	72.24 (0.59)	71.49 (0.45)	**74.48 (0.33)**	73.34 (0.41)	73.89 (0.12)
TE →S	74.09 (0.40)	71.65 (0.50)	72.16 (0.23)	69.09 (1.79)	69.25 (0.31)	70.94 (0.16)	70.94 (0.55)	**71.41 (0.19)**
TE →G	82.19 (0.12)	78.57 (0.60)	79.24 (0.31)	77.80 (0.27)	76.65 (0.20)	79.24 (0.35)	79.65 (0.60)	**79.78 (0.64)**
TE →TR	81.81 (0.20)	75.72 (0.37)	77.29 (0.61)	74.67 (0.50)	74.08 (0.25)	75.27 (0.83)	**76.11 (0.91)**	75.95 (0.50)
TR →F	78.59 (0.34)	73.22 (0.92)	72.44 (0.50)	70.27 (0.45)	69.08 (0.64)	72.20 (0.49)	73.12 (0.08)	**73.13 (0.22)**
TR →S	74.09 (0.40)	70.76 (0.72)	70.97 (0.26)	68.35 (0.62)	67.23 (0.39)	70.28 (0.37)	70.67 (0.50)	**71.28 (0.38)**
TR →G	82.19 (0.12)	80.91 (0.28)	81.67 (0.37)	79.25 (0.34)	78.77 (0.32)	81.26 (0.37)	81.11 (0.42)	**81.55 (0.16)**
TR →TE	78.41 (0.66)	70.41 (1.63)	71.98 (0.50)	69.33 (0.41)	69.45 (0.39)	70.98 (0.11)	70.95 (0.19)	**71.42 (0.12)**
Avg	79.02 (0.34)	75.13 (0.48)	75.53 (0.35)	73.16 (0.55)	72.26 (0.57)	74.45 (0.40)	**74.89 (0.49)**	**74.98 (0.35)**

Table 3: F1 scores for MNLI dataset. We report mean and standard deviation of 3 runs. The five domains are Fiction (F), Slate (S), Government (G), Telephone (TE), and Travel (TR). On average, our method performs better than all baselines.

sults, especially for AMAZON with a small number of examples in each domain. Our adapter method achieves better results compared to DANN with a minimal modification of the hyperparameters.

Replacing UDA Feature Extractors with Adapter Versions is insufficient. *Given that fully fine-tuned* UDA *methods perform well, can we freeze the feature extractors* UDA *methods and fine-tune only adapters and perform effective domain adaptation?* We compare our methods with DANN-🔌 and DANN-🔌-MC and outperform them both in AMAZON and MNLI. This is in line with Karouzos et al. (2021) that although domain adversarial training brings domain representations closer, it introduces distortion in the semantic space, reducing model performance. This shows that simply replacing feature extractors with their adapter versions in existing UDA methods is not an effective strategy.

Gap to Full Fine-Tuning. Fine-tuning a PLM with supervised data in the target domain is the upper bound performance for domain adaptation. The gap from full fine-tuning is greater when more data are available (3.15 in AMAZON and 4.13 in MNLI). This is not surprising, as the supervised fine-tuning works better with more data. However, while adapters perform closely to complete fine-tuning in supervised scenarios (He et al., 2021a), there is still a large gap between domain adaptation and complete fine-tuning and would require significant future work.

3.4 Further Analysis

Adapter Reduction Factor. The bottleneck size (d) of the adapters plays an important role in the final performance of the model. We show the performance of the models at various reduction factors in Figure 2. For JOINT-DT-🔌, smaller reduction factors generally perform well in both AMAZON and MNLI, with performance reducing for larger reduction factors. This shows that the JOINT-DT-🔌 method requires a greater number of parameters to reduce divergence and learn task representations together. Since TS-DT-🔌 adds two adapters, this increases the number of parameters added for the same reduction factor compared to JOINT-DT-🔌. As a result, we find that as the data scale up, relatively low reduction factors work well.

The removal of adapters from continuous layer spans. All adapters are not equal. Removing adapters from the first few layers still preserves performance (Figure 3). For JOINT-DT-🔌 and TS-DT-🔌, the F1 slowly decreases as we continually remove the adapters. However, we obtained a comparable performance after removing the adapters from layers 1-6. This suggests that adapters are effective when added to higher layers, where the divergence between domains is greater at higher layers compared to lower layers (Ramesh Kashyap

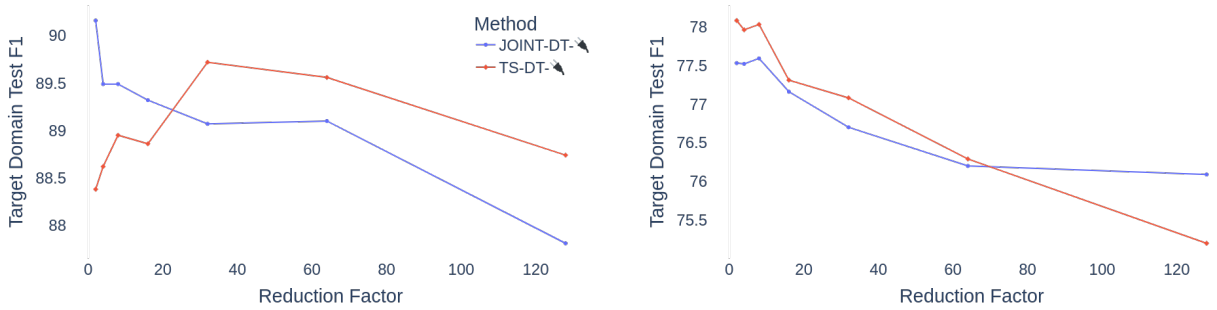


Figure 2: (a) Performance for AMAZON on the C → BA domain adaptation scenario for different reduction factors. (b) Performance for MNLI on the S → TR scenario for different reduction factors.

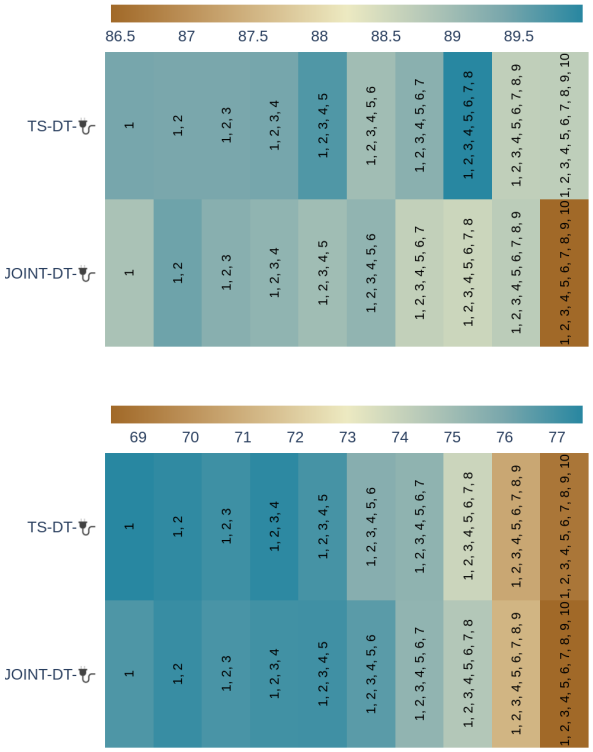


Figure 3: Shows the difference in performance when adapters are removed from certain layers (mentioned inside the cells) for the AMAZON dataset (top) and for MNLI dataset (bottom). The performance reduces if adapters are removed from certain layers

et al., 2021b). Thus we can further reduce the number of parameters for domain adaptation.

t-SNE plots. The t-SNE (van der Maaten and Hinton, 2008) plots from domain adapters are shown in Figure 4 for the data set MNLI. The lower layers have low divergence and the data from the two domains are interspersed, whereas the higher layers have high divergence. Our method effectively reduces the divergence in higher layers.

Composability. We test the composability of our two-step method TS-DT- . We reuse the task adapter trained for C → BA and replace the domain adapter with the domain adapter of C → MR and perform inference on C → MR dataset. The initial F1 of the C → MR dataset was 73.22 and after composing it with a different task adapter, the F1 score is 72.66 – a minimal performance loss. This shows the composability of TS-DT- .

4 Literature Review

Parameter Efficient Fine-tuning Methods. Adapters (Houlsby et al., 2019) are task-specific modules added to frozen transformer layers, with only the adapter parameters updated. Their plug-and-play characteristics and the avoidance of catastrophic forgetting have resulted in their use for NLP tasks: machine translation (Bapna and Firat, 2019), named entity recognition (Pfeiffer et al., 2020b), etc. Recently, (He et al., 2021b) have shown that they are efficient in scenarios where there is minimal supervised data. However, they neither test their performance under domain shift nor propose methods to improve adapter fine-tuning. Closely related to our method is the work of Ngo Trung et al. (2021), who learns a shared-private representation per layer, similar to DSN (Bousmalis et al., 2016). Their method requires balancing multiple loss functions, compared to our simpler two-step domain adaptation method. The stacking of adapters has been followed before by (Pfeiffer et al., 2020b) for cross-lingual tasks: learning a language adapter first and stacking

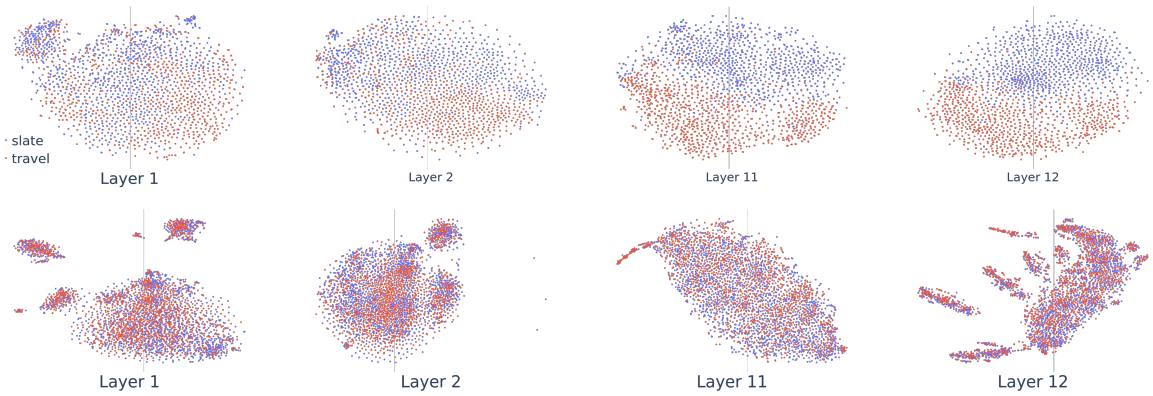


Figure 4: (top)t-SNE plots for the representations from `bert-base-uncased`. The lower layers are domain invariant while the higher layers are domain variant (bottom) tSNE plots from the domain adapter trained on the S → TR domain. We reduce the divergence using domain adapters where even higher layers are domain invariant.

a task adapter. However, one language adapter is learned per language, assumes large amounts of unsupervised data to be available in all the languages, and requires supervised data to be available to learn a task, which is not applicable for domain adaptation. Compared to other methods, we make domain adaptation more efficient using principles of unsupervised domain adaptation.

Unsupervised Domain Adaptation (UDA). Existing UDA approaches can be categorized into model-centric, data-centric, and hybrid. *Model-centric* approaches involve augmenting feature space or altering the loss function, architecture, or model parameters (Blitzer et al., 2006; Pan et al., 2010; Ganin et al., 2016) have been popular. A popular *model-centric* approach is to use adversarial training between the domain and the task classifier (Ganin et al., 2016) to extract domain-invariant information. (Bousmalis et al., 2016) in addition preserves domain-specific information. These works involve training a large number of parameters and require careful balancing of multiple loss functions. Our methods build on top of these works and make it more parameter-efficient.

Large-scale transformers pretrained on domain-specific corpora have been a norm: biomedical (Lee et al., 2019), scientific publications (Beltagy et al., 2019), among others. Another alternative is to continue pretraining generic models on domain-specific data: domain adaptive pretraining (Gururangan et al., 2020). Both solutions are expensive since a huge model has to be stored for every domain while using adapters affords storing a small number of parameters for every domain pair and can be quickly adapted to new domains.

5 Discussion

This work shows that domain adaptation in NLP can be made more efficient using adapters. We use adapters fine-tuning (Houlsby et al., 2019) proposed before and stacking of adapters that have been proposed before for a cross-lingual setting (Pfeiffer et al., 2020b) for the unsupervised domain adaptation. The approach we have discussed will make domain adaptation more practical for real-world use cases, making adaptation faster and cheaper. However, in this work, we have used `bert-base-uncased` for all of our methods. Using other backbone transformer models is part of our future work. We deal only with a classification and natural language inference task. Adapters have previously been used for machine translation (Bapna and Firat, 2019) and other generation tasks (Zhang et al., 2022). We need to explore our domain adaptation methods for other generation tasks.

In this work, we reduce the marginal distribution of the two distributions. Previous works such as Kumar et al. (2018) show that reducing only the marginal distribution is not sufficient and aligning the label distributions is necessary. However, NLP works do not consider this and would require further investigation by the community.

6 Conclusion

In this work, we propose UDAPTER, to make unsupervised domain adaptation more parameter-efficient. Our methods outperform other strong baselines, and we show that we can perform better than just training a task adapter on supervised data. We perform competitively to other UDA methods at a fraction of the parameters and outperform

them when there is limited data – a more practical scenario. Future work should explore other parameter-efficient methods such as prefix-tuning (Li and Liang, 2021) for domain adaptation. NLP should also consider other avenues, such as continuous adaptation to new domains and adaptation to new domains when there are no data available.

7 Acknowledgments

This research is supported by the SRG grant id: T1SRIS19149 and the Ministry of Education, Singapore, under its AcRF Tier-2 grant (Project no. T2MOE2008, and Grantor reference no. MOET2EP20220-0017). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore.

8 Limitations

We have several limitations to our work. We have experimented with only one type of parameter-efficient method, which is the adapter fine-tuning method. Several other alternative parameter-efficient methods, such as LoRA (Hu et al., 2021), Bitfit (Ben Zaken et al., 2022), and other unifying paradigms (He et al., 2021a), have been proposed in recent times. These methods are modular and can be easily substituted for adapters.

Another major limitation of our work is that we cannot explore whether we can learn different tasks over a given pair of domains. For example, for a given pair of domains such as NEWS and TWITTER, it would be ideal if we learned a domain adapter and reused it for different applications such as sentiment analysis, named entity recognition, among others. We are limited by the availability of data for such scenarios and this would be a potential future work.

References

Ankur Bapna and Orhan Firat. 2019. Simple, scalable adaptation for neural machine translation. 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 1538–1548, Hong Kong, China. Association for Computational Linguistics.

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 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447, Prague, Czech Republic. Association for Computational Linguistics.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128, Sydney, Australia. Association for Computational Linguistics.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 343–351.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander J. Smola. 2012. A kernel two-sample test. *J. Mach. Learn. Res.*, 13:723–773.

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, Online. Association for Computational Linguistics.

Wenjuan Han, Bo Pang, and Ying Nian Wu. 2021. Robust transfer learning with pretrained language models through adapters. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 854–861, Online. Association for Computational Linguistics.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021a. Towards a unified view of parameter-efficient transfer learning. *CoRR*, abs/2110.04366.

Ruidan He, Linlin Liu, Hai Ye, Qingyu Tan, Bosheng Ding, Liying Cheng, Jiawei Low, Lidong Bing, and Luo Si. 2021b. On the effectiveness of adapter-based tuning for pretrained language model 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 2208–2222, Online. Association for Computational Linguistics.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685.

Constantinos Karouzos, Georgios Paraskevopoulos, and Alexandros Potamianos. 2021. UDALM: Unsupervised domain adaptation through language modeling. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2579–2590, Online. Association for Computational Linguistics.

Abhishek Kumar, Prasanna Sattigeri, Kahini Wadhawan, Leonid Karlinsky, Rogerio Feris, William T. Freeman, and Gregory Wornell. 2018. Co-regularized alignment for unsupervised domain adaptation. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9367–9378, Red Hook, NY, USA. Curran Associates Inc.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Mingsheng Long, Yue Cao, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. 2019. Transferable representation learning with deep adaptation networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(12):3071–3085.

Nghia Ngo Trung, Duy Phung, and Thien Huu Nguyen. 2021. Unsupervised domain adaptation for event detection using domain-specific adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4015–4025, Online. Association for Computational Linguistics.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 751–760, New York, NY, USA. Association for Computing Machinery.

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, Online. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

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.

Abhinav Ramesh Kashyap, Devamanyu Hazarika, Min-Yen Kan, and Roger Zimmermann. 2021a. Domain divergences: A survey and empirical analysis. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1830–1849, Online. Association for Computational Linguistics.

Abhinav Ramesh Kashyap, Laiba Mehnaz, Bhavitvya Malik, Abdul Waheed, Devamanyu Hazarika, Min-Yen Kan, and Rajiv Ratn Shah. 2021b. Analyzing the domain robustness of pretrained language models, layer by layer. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 222–244, Kyiv, Ukraine. Association for Computational Linguistics.

Baochen Sun, Jiashi Feng, and Kate Saenko. 2016. Correlation alignment for unsupervised domain adaptation. *CoRR*, abs/1612.01939.

Baochen Sun and Kate Saenko. 2016. Deep CORAL: correlation alignment for deep domain adaptation. In *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III*, volume 9915 of *Lecture Notes in Computer Science*, pages 443–450.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

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, Brussels, Belgium. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

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

Linjuan Wu, Shaojuan Wu, Xiaowang Zhang, Deyi Xiong, Shizhan Chen, Zhiqiang Zhuang, and Zhiyong Feng. 2022. Learning disentangled semantic representations for zero-shot cross-lingual transfer in multilingual machine reading comprehension. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 991–1000, Dublin, Ireland. Association for Computational Linguistics.

Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. 2017. Central moment discrepancy (CMD) for domain-invariant representation learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Rongsheng Zhang, Yinhe Zheng, Xiao-Xi Mao, and Minlie Huang. 2021. Unsupervised domain adaptation with adapter. *ArXiv*, abs/2111.00667.

Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. 2022. Continual sequence generation with adaptive compositional modules. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3653–3667, Dublin, Ireland. Association for Computational Linguistics.

Src →Trg	Fully Supervised	Adapter Based				
		DANN-	DSN-	TASK-	TS-DT-	JOINT-DT-
A → BA	87.68 (1.92)	86.46 (0.26)	87.13 (0.23)	87.03 (0.26)	88.24 (0.76)	**88.74 (0.13)**
A → BO	83.73 (1.61)	78.41 (1.14)	80.23 (0.81)	84.15 (1.10)	84.22 (0.76)	**84.96 (0.28)**
A → C	90.00 (1.17)	87.31 (0.39)	87.58 (0.48)	**89.67 (0.32)**	88.76 (1.32)	89.39 (0.23)
A → MR	76.57 (0.36)	75.54 (0.63)	75.96 (0.27)	76.63 (0.92)	77.39 (0.13)	**77.63 (0.71)**
BA → A	88.56 (1.04)	87.72 (1.85)	87.62 (0.86)	88.33 (1.10)	89.55 (0.10)	**89.70 (0.23)**
BA → BO	85.52 (0.59)	82.89 (3.08)	84.26 (0.85)	84.61 (0.39)	84.38 (0.61)	**85.01 (0.60**
BA → C	89.58 (0.32)	86.63 (0.53)	88.44 (0.90)	**90.63 (0.33)**	87.46 (0.88)	88.64 (0.30)
BA → MR	77.26 (0.71)	74.48 (1.79)	48.67 (15.98)	78.74 (0.35)	**79.42 (0.44)**	78.44 (0.70)
BO → A	87.38 (1.08)	85.90 (0.12)	86.62 (0.41)	85.03 (0.36)	84.79 (0.75)	**87.46 (0.27)**
BO → BA	84.72 (1.15)	82.06 (1.15)	82.75 (1.51)	86.50 (0.39)	**86.84 (0.48)**	86.41 (0.79)
BO → C	87.58 (0.67)	86.94 (0.83)	86.61 (1.03)	88.44 (0.53)	87.86 (0.61)	**88.53 (0.43)**
BO → MR	80.14 (0.52)	76.19 (0.89)	72.08 (7.29)	79.44 (0.95)	**80.52 (0.61)**	78.91 (0.38)
C → A	89.46 (0.49)	87.02 (1.86)	85.50 (1.30)	87.74 (1.18)	88.53 (0.42)	**88.92 (0.44)**
C → BA	90.15 (0.46)	88.10 (1.13)	88.56 (0.25)	81.71 (2.72)	**89.72 (0.43)**	89.32 (0.42)
C → BO	85.08 (0.97)	81.18 (2.07)	83.81 (1.68)	80.55 (0.81)	84.14 (0.52)	**85.42 (0.70)**
C → MR	76.03 (1.15)	64.99 (5.91)	63.59 (11.98)	69.53 (1.24)	73.22 (0.48)	**73.50 (0.84)**
MR → A	79.55 (1.38)	81.05 (1.15)	66.28 (19.68)	82.45 (1.43)	81.93 (0.47)	**84.41 (0.43)**
MR → BA	74.63 (9.8)	77.95 (1.46)	54.64 (17.71)	81.70 (1.22)	84.28 (0.41)	**84.91 (0.36)**
MR → BO	86.09 (1.0)	82.83 (0.62)	49.92 (24.06)	**84.90 (0.23)**	84.47 (0.80)	84.45 (0.31)
MR → C	76.54 (1.78)	84.58 (0.46)	69.47 (12.49)	86.68 (0.65)	86.25 (0.38)	**88.37 (0.11)**
Avg	83.81 (1.41)	81.91 (1.37)	76.49 (5.98)	83.72 (0.88)	84.60 (0.57)	**85.16 (0.43)**

Table 4: F1 scores for AMAZON dataset. We report the mean and standard deviation of 3 runs. The five domains are Apparel (A), Baby (BA), Books (BO), Camera_Photo (C) and Movie Reviews (MR). The difference between this table and Table 2 is we experiment with DSN- . fine-tunes a language model using labeled data from the source domain and tests it on the target domain. This shows that just using the supervised data from the source domain is not enough

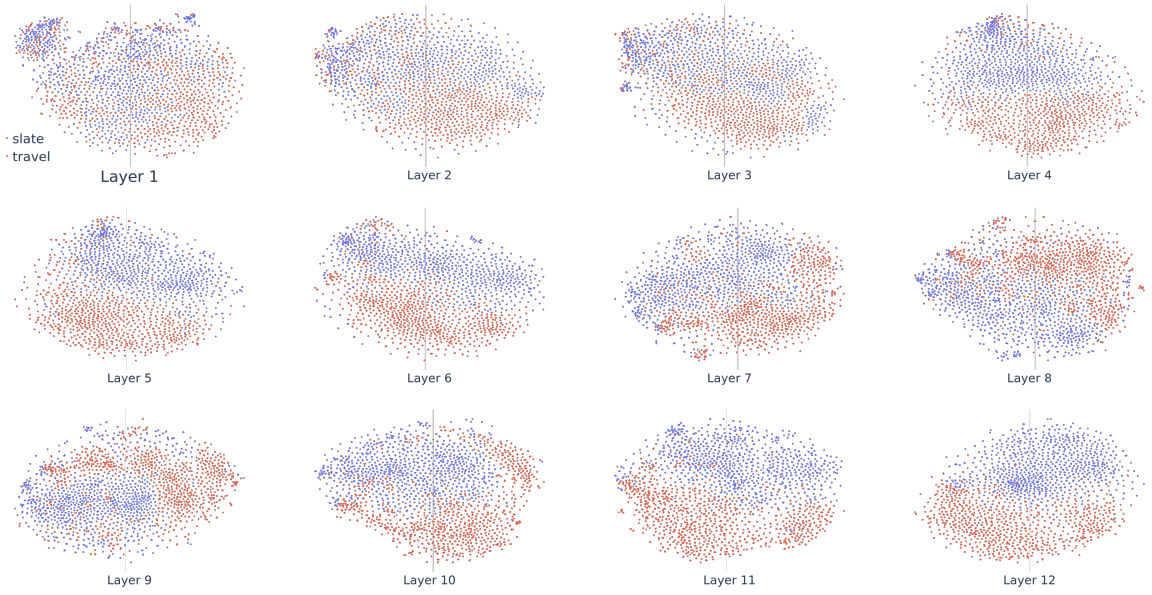


Figure 5: t-SNE plots for the pretrained representations from `bert-base-uncased` for MNLI. Lower layers are domain-invariant whereas higher layers are domain variant.

Src →Trg	Fully Supervised	Adapter Based				
		DANN-	DSN-	TASK-	TS-DT-	JOINT-DT-
F →S	71.58 (0.31)	70.96 (0.03)	70.16 (0.25)	72.36 (0.36)	**73.46 (0.34)**	72.30 (0.26)
F →G	79.05 (0.94)	78.73 (0.43)	77.01 (0.31)	79.00 (0.46)	78.65 (0.25)	**79.79 (0.22)**
F →TE	74.73 (0.41)	70.89 (0.74)	69.89 (0.04)	70.83 (0.54)	**73.05 (0.70)**	71.59 (0.78)
F→TR	75.84 (0.48)	74.42 (0.18)	73.98 (0.70)	75.85 (0.19)	76.75 (0.80)	**77.07 (0.26)**
S →F	76.27 (0.30)	73.89 (0.61)	73.79 (0.06)	75.25 (0.19)	**75.52 (0.89)**	75.35 (0.56)
S →G	81.00 (0.37)	79.99 (0.36)	79.39 (0.16)	80.76 (0.40)	**81.65 (0.11)**	80.94 (0.30)
S →TE	74.32 (0.71)	72.29 (0.57)	71.69 (0.16)	72.66 (0.79)	**74.09 (0.30)**	73.38 (0.63)
S →TR	77.85 (0.40)	75.58 (0.54)	75.24 (0.42)	76.16 (0.22)	**77.31 (0.60)**	77.16 (0.18)
G →F	73.12 (0.39)	71.57 (0.68)	70.67 (0.29)	72.66 (0.31)	72.66 (0.56)	**73.56 (0.23)**
G →S	72.10 (1.01)	70.17 (0.64)	70.31 (0.44)	71.11 (0.38)	71.14 (0.21)	**71.36 (0.04)**
G →TE	72.80 (0.32)	69.45 (0.96)	69.47 (0.25)	71.40 (0.30)	71.53 (1.04)	**71.99 (0.67)**
G →TR	76.76 (0.08)	74.35 (0.22)	74.00 (0.32)	76.29 (0.10)	76.16 (0.34)	**76.79 (0.59)**
TE →F	73.25 (0.36)	72.24 (0.59)	73.04 (0.28)	**74.48 (0.33)**	73.34 (0.41)	73.89 (0.12)
TE →S	69.52 (1.17)	69.09 (1.79)	69.40 (0.42)	70.94 (0.16)	70.94 (0.55)	**71.41 (0.19)**
TE →G	77.59 (1.38)	77.80 (0.27)	77.56 (0.46)	79.24 (0.35)	79.65 (0.60)	**79.78 (0.64)**
TE →TR	72.45 (2.44)	74.67 (0.50)	74.14 (0.21)	75.27 (0.83)	**76.11 (0.91)**	75.95 (0.50)
TR →F	72.78 (0.37)	70.27 (0.45)	71.10 (0.21)	72.20 (0.49)	73.12 (0.08)	**73.13 (0.22)**
TR →S	70.40 (0.10)	68.35 (0.62)	69.92 (0.50)	70.28 (0.37)	70.67 (0.50)	**71.28 (0.38)**
TR →G	79.75 (0.42)	79.25 (0.34)	79.75 (0.24)	81.26 (0.37)	81.11 (0.42)	**81.55 (0.16)**
TR →TE	72.02 (0.49)	69.33 (0.41)	70.10 (0.52)	70.98 (0.11)	70.95 (0.19)	**71.42 (0.12)**
Avg	74.66 (0.62)	73.16 (0.55)	73.03 (0.32)	74.45 (0.40)	74.89 (0.49)	**74.98 (0.35)**

Table 5: F1 scores for MNLI. We report mean and standard deviation of 3 runs. The five domains are Fiction (F), Slate (S), Government (G), Telephone (TE), and Travel (TR). The difference between this table and Table 3 is we experiment with DSN- . fine-tunes a language model using labeled data from the source domain and tests it on the target domain. This shows that just using the supervised data from the source domain is not enough.

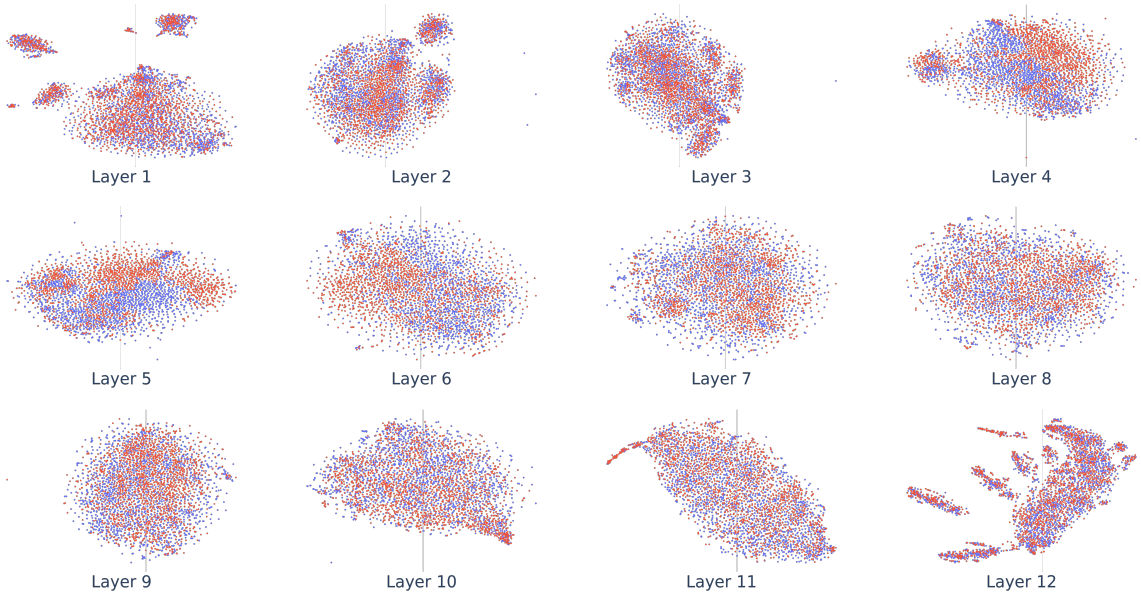


Figure 6: t-SNE plots for the representations from domain adapter trained on S → TR domain for MNLI. We reduce divergence between domains for all layers.

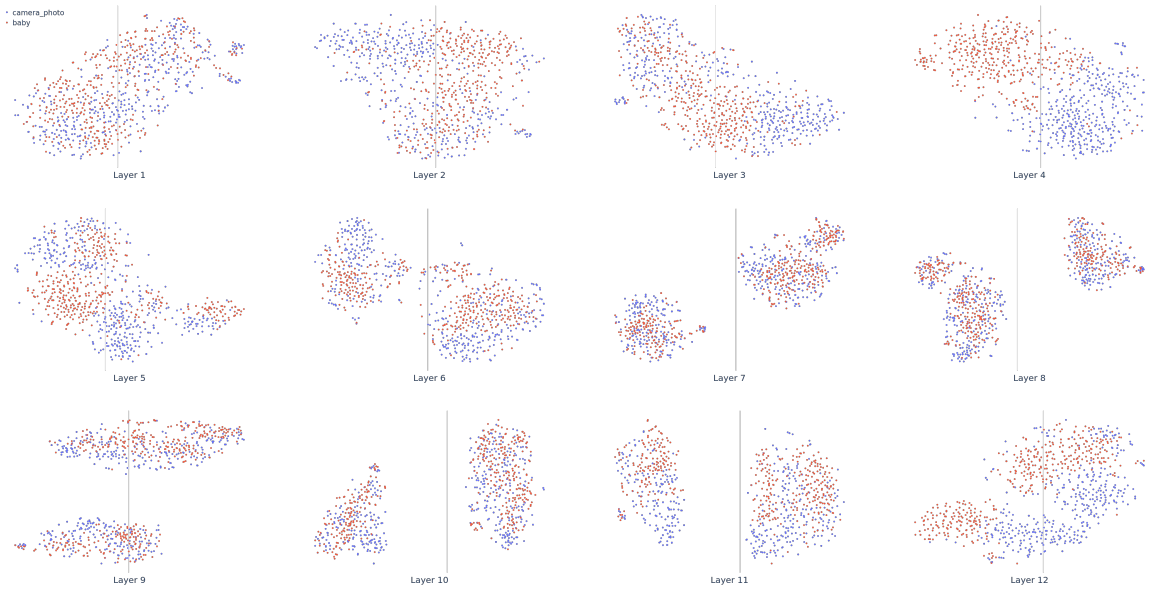
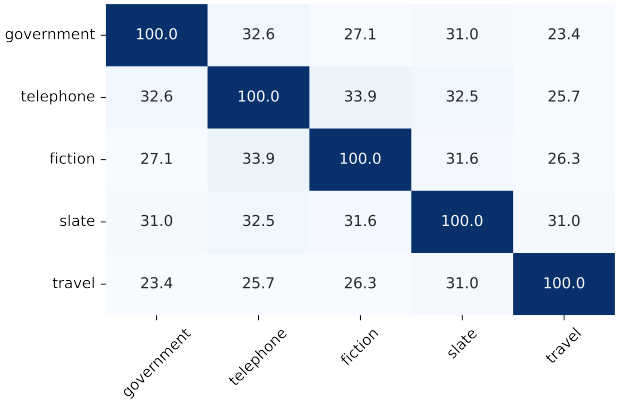


Figure 7: t-SNE plots for the pretrained representations from `bert-base-uncased` for AMAZON. Lower layers are domain-invariant whereas higher layers are domain variant.

Figure 8: t-SNE plots for the representations from domain adapter trained on C→ BO domain for AMAZON. We reduce divergence between domains for all layers.

(a)



(b)

Figure 9: (a) Vocabulary overlap (%) between domains in AMAZON. (b) Vocabulary overlap (%) between domains in MNLI.