DADEE: Unsupervised Domain Adaptation in Early Exit PLMs

Divya Jyoti Bajpai and Manjesh Kumar Hanawal
Department of IEOR, IIT Bombay
{divyajyoti.bajpai, mhanawal}@iitb.ac.in

Abstract

Pre-trained Language Models (PLMs) exhibit good accuracy and generalization ability across various tasks using self-supervision, but their large size results in high inference latency. Early Exit (EE) strategies handle the issue by allowing the samples to exit from classifiers attached to the intermediary layers, but they do not generalize well, as exit classifiers can be sensitive to domain changes. To address this, we propose Unsupervised Domain Adaptation in EE framework (DADEE) that employs multi-level adaptation using *knowledge distillation*. DADEE utilizes GAN-based adversarial adaptation at each layer to achieve domain-invariant representations, reducing the domain gap between the source and target domain across all layers. The attached exits not only speed up inference but also enhance domain adaptation by reducing catastrophic forgetting and mode collapse, making it more suitable for real-world scenarios. Experiments on tasks such as sentiment analysis, entailment classification, and natural language inference demonstrate that DADEE consistently outperforms not only early exit methods but also various domain adaptation methods under domain shift scenarios. The anonymized source code is available at https://github.com/Div290/DAdEE.

1 Introduction

Pre-trained Language Models (PLMs) such as BERT (Devlin et al., 2018), GPT (Radford et al., 2019), XLNet (Yang et al., 2019), and RoBERTa (Liu et al., 2019) have experienced substantial growth in size to attain state-of-the-art performances across a wide range of natural language processing tasks. Despite their remarkable efficacy, PLMs suffer from inference latencies, limiting their utility in industrial applications requiring faster inference. Prior research (Zhou et al., 2020; Zhu, 2021) has also highlighted overthinking issues in PLMs. More specifically, shallow representations in the initial layers may suffice for correct inference of 'easy' samples, while final layer representations may overfit or be influenced by irrelevant features, resulting in poor generalization and computational inefficiency.

To circumvent this, several methods such as pruning (Michel et al., 2019), quantization (Kim et al., 2021) and knowledge distillation (Jiao et al., 2019) have been proposed. Among them, Early Exit methods (Zhou et al., 2020; Zhu, 2021) have gained significant attention, where inference can be made at classifiers attached at intermediary layer based on the hardness of the input samples. They perform adaptive inference strategies to address both efficiency and overthinking concerns.

Nevertheless, all these methods do not generalize well to new domains. Also, as EE introduces more parameters in the exit classifers, it requires high-quality labeled training data to learn the exit weights. The cost of creating labeled training data is often prohibitively expensive. The question then arises: How can we adapt early exit PLMs to diverse domains in an unsupervised setup?

Domain adaptation is a vital technique in machine learning to ensure models perform well on data from a target domain, even if trained on a different source domain. Unsupervised domain adaptation methods primarily focus on aligning source and target data in a shared feature space. This alignment is achieved by optimizing the model's representation to minimize domain discrepancies, such as maximum mean discrepancy (Tzeng et al., 2014) or correlation instances (Sun and Saenko, 2016). Previous approaches, like adversarial training methods (Ajakan et al., 2014; Tzeng et al., 2017; Ryu et al., 2022), have utilized adversarial objectives to bridge domain gaps by producing domain-invariant features at the final layer. However, since the Early Exit PLMs (EEPLMs) have

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 6389–6400

added exits, domain adaptation methods cannot be directly used as they only focus on adapting to the final layer performance. This necessitates domain adaptation at every layer such that domain invariant features are not only available at the final layer but across all the layers of PLM.

To tackle this problem, we present a novel strategy that integrates adversarial domain adaptation techniques in early exits that adapt EEPLMs to diverse domains. Our approach, named Unsupervised Domain Adaptation in EE PLMs (DADEE), emphasizes achieving domain invariant representations across all the layers such that exit classifiers trained on a source domain can be directly utilized for the target domain. This allows for the exit classifiers trained on the source domain to be directly utilized in the target domain without requiring labels.

Our method not only speeds up inference but also allows for fast and robust bridging of domain gaps compared to traditional methods that rely solely on final layer representations in adversarial setups, which is insufficient for such large and complex models. During adversarial training, the exits aid the adaptive process by mitigating the risk of catastrophic forgetting and mode collapse using knowledge distillation between source and target representations across all the layers.

By leveraging multi-level domain feature adaptation, DADEE enhances overall effectiveness across real-world applications, addressing the challenges of inference speed and domain adaptation in PLMs. Also, in scenarios where the size of the source dataset is limited, early exit models can adapt more readily as these models do not depend on the prediction of just the final layer but multiple layers, improving generalization. Our method achieves improved performance metrics on sentiment analysis, entailment classification, and natural language inference (NLI) tasks.

Our main contributions are as follows:

- We propose DADEE for unsupervised domain adaptation to bridge the gaps between the source and target domains in EEPLMs.
- DADEE uses a GAN-based multi-level adaptation to bridge the domain gaps, i.e., we perform layer-by-layer adaptation.
- We utilize EE strategies not only for faster inference but also for the adaptation process. Our method gets the best of both early exit and domain adaptation methods to simultaneously increase both performance and speed.
- Through extensive experiments on sentiment analysis, entailment classification and natural language inference (NLI) tasks, we show that DADEE achieved an average improvement of 2.9% in accuracy and $1.61\times$ average inference speed up as compared to previous vanilla PLM inference.

2 Related works

In this section, we discuss studies relevant to our work in domain adaptation and early exiting.

Domain Adaptation: The aim is to learn domain-invariant representations for labeled source and unlabeled target domains. Key methods include Deep Domain Classification (Tzeng et al., 2014), which minimizes maximum mean discrepancy with classification loss, and Deep Adaptation Network (Long et al., 2015), employing multiple kernels across layers. DCA (Sun and Saenko, 2016) reduces the disparity in second-order statistics. Adversarial methods like DANN (Ajakan et al., 2014) use a domain classifier with a gradient reversal layer for domain confusion. Similarly Domain Separation Networks (DSN) (Bousmalis et al., 2016) have a notion of private subspaces for every domain and separate the information for each domain.

Generative approaches, such as CyCADA (Hoffman et al., 2018), enforce cycle and semantic consistency. DAAT (Du et al., 2020) enhances domain awareness through post-training, while ADDA (Tzeng et al., 2017) introduces GAN-based loss, further improved by AAD (Ryu et al., 2022) with knowledge distillation. Pivot-based methods (Blitzer et al., 2007; Yu and Jiang, 2016; Ziser and Reichart, 2018; Peng et al., 2018; Zhang et al., 2019) induce shared low-dimensional features based on pivot co-occurrence. Multi-level domain adaptation methods (Malik et al., 2023) utilize all layers to bridge domain gaps.

Early Exits: are input adaptive inference methods. For image classification, BranchyNet (Teerapittayanon et al., 2016) uses classification entropy for early inference, while MSDNet (Huang et al., 2017) selects thresholds based on confidence distribution. DeeCAP (Fei et al., 2022) and MuE (Tang et al., 2023) extend it to image captioning.

Early exiting has also been applied to PLMs for various NLP tasks (Bapna et al., 2020; Elbayad

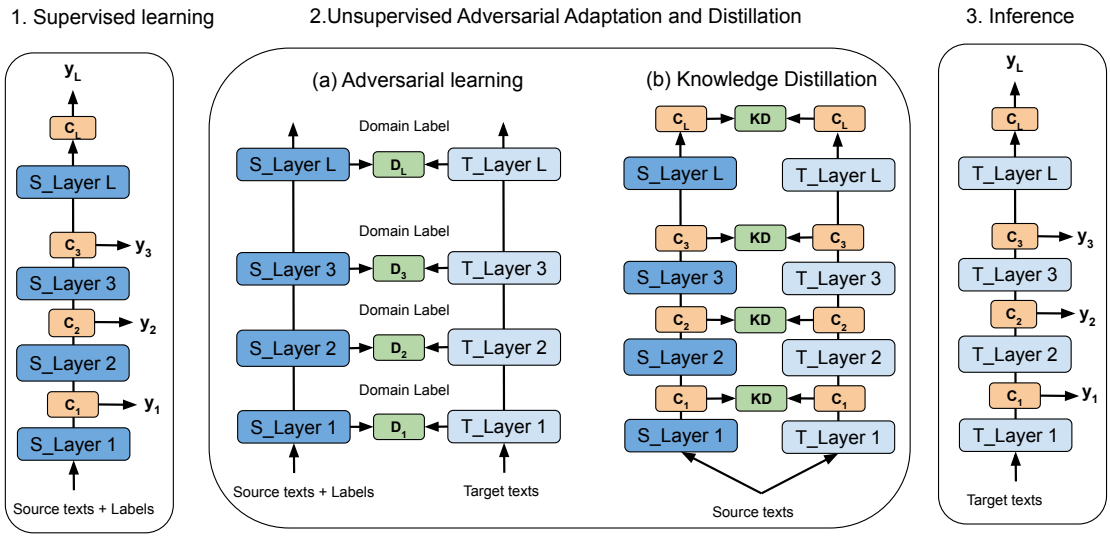


Figure 1: This figure outlines our model's workflow with three main steps: 1) Supervised learning with attached exits during backbone training. 2) Adversarial adaptation and distillation, where a) a discriminator predicts domain labels at each layer, and b) the target backbone is trained for domain-invariant features. 3) The target backbone produces domain-invariant features across all layers, ready for target domain inference.

et al., 2020; Liu et al., 2021; Xin et al., 2021; Zhou et al., 2020; He et al., 2021; Banino et al., 2021; Ji et al., 2023). DeeBERT (Xin et al., 2020) and ElasticBERT (Liu et al., 2021) propose different fine-tuning strategies. PABEE (Zhou et al., 2020) bases early exit decisions on prediction consistency. DDA (Li et al., 2021) uses exits for dynamic domain adaptation. It considers a feature classifier to bridge the domain gap which is not sufficient when there is a larger domain gap. CeeBERT (Bajpai and Hanawal, 2024) adapts thresholds for target domains using multi-armed bandits.

The key differences in our work are: 1) We are the first to perform domain adaptation in the early exit PLMs. 2) We utilize a GAN-based framework for adversarial training across all exits for multi-level domain adaptation. 3) Our exits employ knowledge distillation between the source and target domains to mitigate catastrophic forgetting, thus outperforming both the domain adaptation and early exit methods.

3 Methodology

In this section, we detail our method. We start with a PLM such as BERT/RoBERTa with L layers. We attach exit classifiers to each layer and train them using the source dataset. We then perform multi-level adversarial training on target data.

3.1 Training the source backbone

For any source sample (x_s, y_s), the loss at exit classifier i is computed as:

$$\mathcal{L}_i(\theta) = \mathcal{L}_{CE}(f_i(x_s, \theta), y_s), \quad (1)$$

where $f_i(x_s, \theta)$ is the output of the classifier attached at the ith layer, θ is the set of learnable parameters of the source backbone, and $\mathcal{L}_{CE}$ is the cross-entropy loss. We learn the parameters for all the exit-classifiers simultaneously following the approach of Shallow-Deep (Kaya et al., 2019), with the loss function defined as $\mathcal{L} = \frac{\sum_{i=1}^{L} i \cdot \mathcal{L}i}{\sum_{i=1}^{L} i}$, over all the L layers. This weighted average considers the relative inference cost of each internal classifier. This is shown as the step Supervised learning in Fig 1. After this training, the weights of the source encoder and exit classifiers at each layer are frozen. We note that as a part of the training, a function C_i that maps the source encoder's output to class probabilities at ith layer is also learned and frozen.

3.2 Adversarial adaptation on target

We initialize the target encoder weights with what is learned for the source encoder. We assume that the label space of the target dataset is the same as the source label space $\mathcal{C}$. We denote outputs of the ith layer of the source and target encoder as $E_i^s(x)$ and $E_i^t(x)$, respectively. Let D_i denote a

discriminator function that maps the output of ith layer of either source encoder or target encoder to domain probabilities, i.e., target or source domain.

We train the target encoder and the discriminator alternately as the ADDA framework (Tzeng et al., 2017). This can be formulated as an unconstrained optimization problem and is represented as step 2(a): Adversarial Framework in Fig. 1. Let D_s and D_t denote the target distributions. For $x_s \sim D_s$ and $x_t \sim D_t$, loss for ith discriminator can be formulated as

$$\mathcal{L}_i^{dis}(x_s, x_t) = -\log D_i(E_i^s(x_s)) - \log(1 - D_i(E_i^t(x_t))), \quad (2)$$

and the overall discriminator loss across all the layers can be as $\mathcal{L}^{dis} = \frac{\sum_{i=1}^{L} i \cdot \mathcal{L}_i^{dis}}{\sum_{i=1}^{L} i}$. The generator loss for ith layer is:

$$\mathcal{L}_i^{gen}(x_t) = -\log D_i(E_i^t(x_t)) \quad (3)$$

Given that the weights of the target encoder are untied from those of the source encoder, the target encoder has more flexibility in learning the specific domain features. However, this formulation is prone to catastrophic forgetting (Ryu et al., 2022), leading to erratic classification performance, as it lacks access to class labels and can diverge from the original task.

To address this challenge, we employ knowledge distillation. This involves introducing distillation loss alongside generator loss after each layer to mitigate the risk of catastrophic forgetting and mode collapse across all layers. We ensure robustness throughout the model by leveraging classifiers (exits) appended after each layer to distil the knowledge between source and target. The formulation for knowledge distillation loss is expressed as follows:

$$\mathcal{L}_i^{KD} = KL(C_i(E_i^s(x_s)), C_i(E_i^t(x_s))) \quad (4)$$

where KL is the KL-divergence loss. The total loss of generator of the ith classifier then becomes $\mathcal{L}_i = \mathcal{L}_i^{gen} + \mathcal{L}_i^{KD}$, and the overall generator loss is taken as $\mathcal{L} = \frac{\sum_{i=1}^{L} i \cdot \mathcal{L}_i}{\sum_{i=1}^{L} i}$. We provide lower weights to the discriminator and generator at initial layers since these layers are responsible for learning the general features which should not be changed much while deeper domain-specific rich representations lie in deeper layers justifying higher weights to deeper layers.

After this step, the target backbone is ready for inference as discussed next.

3.3 Inference on target dataset

For any $x_t \sim \mathcal{D}_t$, let $p_i^t(c)$ denote the probability assigned at layer i that a sample belongs to class $c \in \mathcal{C}$. Let $S_i := \max_{c \in \mathcal{C}} p_i^t(c)$. It denotes the confidence in prediction at the i layer. The decision to exit early is made based on this confidence score exceeding a fixed threshold α, i.e., a sample exit from layer i if $S_i \geq \alpha$, else it is processed to the next layer. When the sample exits at layer i, it is assigned a label $\arg\max_{c \in \mathcal{C}} p_i^t(c)$. If the sample's confidence is below α for all the intermediary layers, the sample is inferred at the final layer.

We set the value of α using the validation split of the source dataset. We use the same threshold for the target dataset as after adversarial training as all the layers provide domain invariant feature representation resulting in a similar accuracy-efficiency trade-off in the target domain as learned on the source domain.

3.4 Analysis

This section provides a theoretical justification of DADEE. Initially, we delve into the existing theoretical framework, subsequently elucidating the advantageous aspects of early exits and adversarial training from a theoretical standpoint. We follow the method pioneered in (Ben-David et al., 2010) to upper bound expected error of a hypothesis on the target domain.

Let h_s^* and h_t^* denote the hypotheses that assign ground-truth labels for the source and target domain, respectively. We define the disagreement function for any hypothesis h_1 and h_2 as :

$$\epsilon(h_1, h_2) = \mathbb{E}[|h_1(x) - h_2(x)|]. \quad (5)$$

For any hypothesis h, define $\epsilon_s(h) = \epsilon_s(h, h_t^*)$ and $\epsilon_t(h) = \epsilon_t(h, h_s^*)$ as the expected error on the source and target domain respectively. The error $\epsilon_t(h)$ of a hypothesis h on the target domain can be bounded using three terms: (a) expected error of h on the source domain, $\epsilon_s(h)$; (b) $\mathcal{H}\Delta\mathcal{H}$-distance $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_s, \mathcal{D}_t)$ measuring domain shift as the discrepancy between the disagreement of the two hypotheses $h, h^{'} \in \mathcal{H}$ which is defined as

$$d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_s, \mathcal{D}_t) = 2 \sup_{h, h^{'} \in \mathcal{H}} |\epsilon_s(h, h^{'}) - \epsilon_t(h, h^{'})| \quad (6)$$

and (c) the error λ of the ideal joint hypothesis h^* on both source and target domains. The upper bound is:

$$\epsilon_t(h) \leq \epsilon_s(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}} + \lambda \quad (7)$$

Usually, λ is considered negligible and discarded. Therefore, we can focus on making the first and second terms small to keep the target error small.

For the first term, the error rate of the source domain is minimized by training it on the labeled training data. For the second term, it is required that the PLM generate similar features for the source as well as the target dataset. By imposing multi-level adaptation, we reduce the value of the second term in the upper bound. In our method, all the exits simultaneously bridge the domain gap across all the layers instead of just the final one. This helps in layer-by-layer adaptation to the target domain, producing similar feature representations for the target and source datasets across all the layers instead of just the final layer. This helps significantly reduce the second term. We demonstrate it in figure 2a by calculating the $\mathcal{A}$-distance. Note that if knowledge distillation is not applied at every layer, it could increase the value of the first term because of mode collapse or catastrophic forgetting. This aspect was not considered in previous methods like (Ryu et al., 2022), where knowledge distillation is applied only at the last layer.

4 Experiments

In this section, we elaborate on all the experimental details of our work.

4.1 Datasets

We conduct experiments on well-established benchmark datasets sourced from Amazon reviews (Blitzer et al., 2007). These datasets cover reviews from four distinct domains: Books (B), DVDs (D), Electronics (E), and Kitchen appliances (K). Additionally, we utilize the Airline review dataset (A) (Nguyen, 2015) and the IMDB dataset (I) (Maas et al., 2011). In total, our study encompasses 30 domain adaptation tasks for sentiment analysis. For NLI and entailment classification, we use the datasets available in GLUE (Wang et al., 2019) and ELUE (Liu et al., 2021) tasks.

For each domain, we use the train split of the source (with labels) and the target domain (without labels) to train and adapt the backbone. The validation split of the source dataset is used for development and then finally the model is tested on the target test set.

4.2 Implementation details

Training: Initially, we train the backbone on the source dataset. We add a linear classifier layer after each intermediate layer of the pre-trained BERT/RoBERTa model, running the model for three epochs. The training uses a batch size of 16 and a learning rate of 1e-5 with the ADAM optimizer (Kingma and Ba, 2014). We apply early stopping and select the best-performing model on the development set. Subsequently, we freeze the source encoder and initialize the target encoder with the source encoder's weights. The discriminator is an MLP with two hidden layers (hidden size 3072) and LeakyReLU activation. The domain adaptation step runs for five epochs with the same hyperparameters. The experiments are conducted on a single NVIDIA RTX 2070 GPU with an average runtime of less than 10 minutes.

4.3 Inference

During inference, we use a batch size of 1 and α is chosen as the best-performing threshold on the source dataset's validation split based on accuracy. The search space for α is $S_\alpha = \{0.8, 0.85, 0.9, 0.95, 1.0\}$. We apply the same threshold as learned on the source dataset since all layers produce domain-invariant features, allowing the threshold to work effectively for the target domain as well.

Speedup metric: To maintain consistency with previous methods, we use the speedup ratio as the metric to assess our model which could be written as: $\frac{\sum_{i=1}^{L} L \times n_i}{\sum_{i=1}^{L} i \times n_i}$ where n_i are the number of samples exiting from the ith layer and L represents the number of layers. This metric could be interpreted as the increase in speed of the model as compared to the naive BERT/RoBERTa models.

4.4 Baselines

We compare our method against state-of-the-art early exiting and domain adaptation techniques, categorizing the baselines into four groups:

1) Backbone: We use vanilla BERT/RoBERTa as the primary baseline, in this we fine-tune the model on source domain data and infer on complete target data without adaptation.

2) Domain Adaptation: This category includes methods focused solely on domain adaptation. DANN (Ajakan et al., 2014) employs adversarial training on deep neural networks, adapting the representation encoded in a 5000-dimensional feature vector of frequent unigrams and bigrams. IATN (Zhang et al., 2019) features two attention networks: one identifies common features be-

tween domains through domain classification, and the other extracts information from aspects using these common features. DAAT (Du et al., 2020) initializes with BERT's pre-trained weights and adapts it using novel self-supervised pre-training tasks followed by adversarial training. This method is specific to BERT and not easily extendable to other PLMs. AAD (Ryu et al., 2022) uses adversarial training with distillation from the source to reduce overfitting. UDApter (Malik et al., 2023) performs unsupervised multi-level domain adaptation.

3) Early Exit Methods: PABEE (Zhou et al., 2020) is a state-of-the-art early exit method that exits based on prediction consistency. This is chosen as a baseline to give an interpretation of how vanilla early exiting performs without adaptation.

4) Domain Adaptation with Early Exit: DDA (Li et al., 2021) performs multi-level adaptation for image datasets to enable faster inference, though it can suffer from catastrophic forgetting. CeeBERT (Bajpai and Hanawal, 2024) uses multi-armed bandits to adapt thresholds, focusing on inference efficiency without directly addressing the domain gap. In the result tables, 'Final' represents the performance of our method when inference is conducted solely at the final layer after adaptation, without utilizing early exits.

4.5 Experimental Results

In Tables 1 and 2, we present comprehensive results using the BERT/RoBERTa backbone across various domain pairs for sentiment analysis, while Table 3 showcases results for NLI and entailment classification tasks on BERT. Each experiment is performed five times with different seeds, and the average results are reported. Our method consistently outperforms existing baselines.

While PABEE, an early exiting method without adaptation, shows comparable performance to BERT, both lack domain adaptation techniques. Previous methods like DANN and IATN rely on word2vec or GloVe embeddings, which fail to capture the nuanced word characteristics that BERT's contextual embeddings can. By leveraging BERT embeddings, we enhance these methods to achieve comparable performance to more advanced baselines. AAD, which uses BERT but focuses only on the final layer for adaptation, fails to enforce domain-invariant representations across all layers, resulting in suboptimal performance. DAAT improves on prior baselines by incorporating an additional post-training step that makes it domain-aware before performing adversarial training. However, DANN's method is specific to BERT and not easily extensible.

The performance of DDA suffers from overfitting to the source domain data and does not achieve better speedup or performance compared to our approach due to poor generalization as explained in (Ryu et al., 2022). Additionally, DDA only has a feature classifier to reduce the domain gap which might not be sufficient to effectively minimize the domain discrepancy.

Our method surpasses previous baselines by adapting to the target domain across all network layers, creating domain-invariant hidden representations throughout rather than relying solely on the final layer. DADEE effectively reduces the issues of catastrophic forgetting using knowledge distillation which further makes adaptation robust. As shown in the last two columns of Table 1, we observe performance improvements even with early predictions, mitigating the issue of overthinking.

Moreover, we achieve an average speedup of 1.61x, with the highest at 2.11x and the lowest at 1.00x, demonstrating our method's efficiency in both domain adaptation and faster inference, making the model both quick and robust.

4.6 $\mathcal{A}$-distance

From Equation 6, we note the significance of domain divergence, a pivotal metric in assessing the efficacy of domain adaptation methods. To quantify this, we calculate the $\mathcal{A}$-distance, commonly utilized for measuring domain dissimilarity. Defined as $d_{\mathcal{A}} = 2(1 - 2\epsilon)$, where ϵ denotes the generalization error of a classifier trained to discern source from target domain samples. Following existing methods, we divide the data into equal subsets (both source and target), train a linear SVM on one subset to classify domain origin, and evaluate error rates on the other subset, deriving the $\mathcal{A}$-distance accordingly.

Comparing the $\mathcal{A}$-distance across BERT, DAAT, DDA, and our method as these are the top best-performing methods, BERT consistently exhibits the highest $\mathcal{A}$-distance across dataset pairs. DDA and DAAT demonstrate lower $\mathcal{A}$-distance, attributed to their application of adversarial training, which mitigates domain dissimilarity. Our method achieves the lowest $\mathcal{A}$-distance, owing to adversarial training and reduction of catastrophic forgetting across all layers.

		Domain Adaptation methods					*Early Exit models*				
S → T	**BERT**	**DANN**	**IATN**	**DAAT**	**AAD**	**UDA**	**PABEE**	**CBRT**	**DDA**	**Final**	**Our**
B → D	85.9	85.5	86.3	87.2	86.6	87.5	86.0	86.5	86.9	88.5	**88.7**
B → E	85.7	84.3	86.5	87.5	86.7	87.3	85.3	86.3	87.1	87.4	**87.8**
B → K	87.4	86.7	88.1	89.2	88.3	88.9	87.1	89.4	89.6	89.8	**90.6**
B → A	84.5	83.5	85.3	86.5	85.9	86.8	84.9	86.0	86.3	86.7	**87.1**
B → I	83.1	82.9	83.6	83.9	82.7	84.1	83.3	83.5	83.8	83.6	**84.0**
D → B	85.0	83.7	86.6	87.4	87.0	87.2	84.5	86.9	88.1	87.9	**88.4**
D → E	83.9	81.9	84.2	86.0	85.6	85.9	84.2	85.8	86.4	86.3	**86.5**
D → K	85.3	85.5	85.8	87.8	87.4	88.0	84.7	86.1	87.8	88.1	**88.5**
D → A	80.7	81.9	82.6	**86.7**	84.1	86.4	80.3	85.2	86.3	86.5	86.6
D → I	82.5	81.6	82.5	84.7	83.0	84.1	82.6	83.9	84.6	84.9	**84.9**
E → B	84.7	83.1	85.0	85.8	85.4	85.5	84.4	85.1	85.7	85.8	**86.1**
E → D	84.4	81.8	85.4	86.1	85.2	86.4	83.8	84.7	86.6	86.9	**87.3**
E → K	90.2	88.4	90.8	90.9	90.7	90.8	89.7	90.3	90.1	90.8	**91.1**
E → A	83.8	85.2	86.1	87.1	86.6	87.3	83.5	85.6	86.8	87.2	**87.5**
E → I	79.5	79.7	80.4	82.5	81.1	82.7	79.6	81.9	82.4	82.8	**82.8**
K → B	85.1	82.6	85.3	86.1	84.9	86.5	84.8	86.1	86.3	86.5	**86.8**
K → D	83.0	82.3	84.0	84.4	83.5	84.2	83.2	84.2	83.9	84.1	**84.5**
K → E	88.1	86.9	88.1	88.6	88.1	88.9	87.9	88.5	89.0	89.7	**89.7**
K → A	80.3	83.0	83.9	85.7	85.8	85.6	80.2	83.2	85.8	86.0	**86.2**
K → I	80.7	77.5	80.7	80.6	80.2	80.8	80.4	80.6	80.9	81.2	**81.3**
A → B	76.9	77.2	78.2	78.9	78.5	78.4	77.1	78.8	79.1	79.4	**79.6**
A → D	77.6	77.9	79.4	80.2	79.8	80.0	77.4	79.2	80.0	80.1	**80.4**
A → E	84.4	84.1	84.8	85.8	85.1	85.9	84.5	86.0	86.1	85.8	**86.2**
A → K	85.3	82.3	85.5	87.7	85.8	87.4	84.9	86.3	87.9	88.3	**88.3**
A → I	72.1	74.9	75.4	76.4	75.2	76.1	71.8	75.7	76.2	76.7	**77.3**
I → B	84.1	82.5	84.1	87.3	86.6	87.8	84.3	86.4	86.9	87.9	**88.4**
I → D	84.8	83.8	84.6	**86.5**	85.7	86.2	84.6	85.8	86.2	86.3	**86.5**
I → E	81.9	84.0	84.9	87.6	87.1	87.5	82.1	85.3	87.8	88.2	**88.2**
I → K	85.2	84.7	85.5	85.4	85.0	84.9	85.2	85.7	85.5	85.7	**86.0**
I → A	82.4	83.5	84.2	85.1	84.5	85.5	82.6	84.9	85.4	85.4	**85.9**
Average	83.3	82.7	84.3	85.5	84.7	85.4	83.1	84.8	85.5	85.8	**86.2**
Avg. Spd	1.00×	1.00×	1.00×	1.00×	1.00×	1.00×	1.33×	1.53×	1.29×	1.00×	**1.61**×

Table 1: Main results: This table shows the results of our method compared with other baselines. UDA is the UDApter baseline and CBRT is CeeBERT. Avg. Spd is the average speedup.

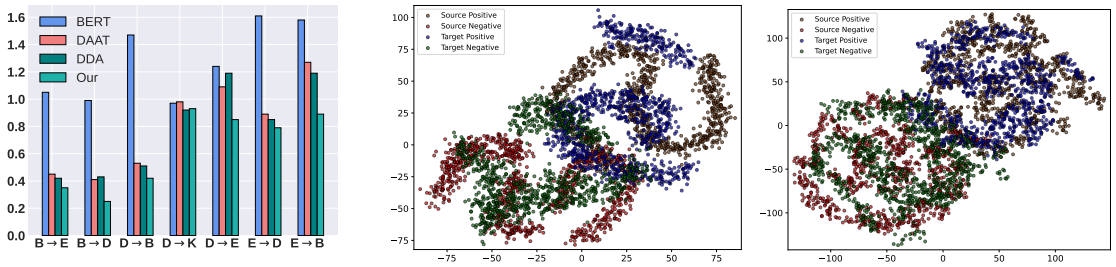
S → T	**RBERT**	**AAD**	**DAAT**	**final**	**Our**
B → D	87.9	88.1	88.5	88.7	**89.0**
B → E	89.5	89.6	89.2	90.4	**90.4**
B → K	90.4	92.1	91.9	92.5	**92.8**
B → A	84.6	85.9	86.3	86.9	**86.9**
B → I	85.7	85.5	85.9	86.2	**86.3**
D → B	87.2	89.4	89.7	90.1	**90.3**
D → E	89.6	89.8	90.1	90.4	**90.5**
D → K	88.5	91.1	92.0	91.9	**92.1**
D → A	84.8	85.6	85.5	86.2	**86.2**
D → I	86.3	86.4	86.9	87.3	**87.8**
E → B	85.9	88.2	88.7	89.8	**90.0**
E → D	84.1	87.8	87.3	88.5	**88.8**
E → K	92.4	92.9	92.6	93.4	**93.4**
E → A	84.0	87.1	86.4	87.7	**87.9**
E → I	79.4	84.9	83.7	85.4	**85.6**
Average	86.7	88.3	88.3	89.0	**89.2**
Avg.Speed	1.00	1.00	1.00	1.00	**1.67**

Table 2: This method shows the results of DADEE and other baselines on the RoBERTa-base(RBERT) model.

4.7 Feature visualization

For an intuitive understanding of the effects of multi-level domain adaptation on BERT, we further perform visualization of feature representations of our method and the DDA method. We perform the feature representation visualization on the test split of the source and target domain for the D → E task. In figure 2b, 2c, we apply t-SNE (t-distributed Stochastic Neighbor Embedding) on the set of all representations of source and target data points. Every sample is mapped into a 768-dimensional feature space by BERT and projected back to the two-dimensional plane by the t-SNE.

From figure 2b, we can observe that there is a lack of distinction between source positive and source negative samples in DDA and many points overlap when in target positive and target negatives. Still, it manages to distinguish between the larger portion of data from the target domain due to adaptive inference. For our method, the target domain positives and negatives are well-separated with few overlaps showing that the multi-level domain adaptation in an adversarial setup can benefit in bridging the domain gap more efficiently. Also, in our method, there is a better overlap between source positives, target positives, and source negatives, as well as target negatives, which further shows that our method can give better domain in-



(a) $\mathcal{A}$-distance comparison (b) t-SNE visualization of DDA (c) t-SNE visualization of DADEE

Figure 2: **Left** figure shows $\mathcal{A}$-distance comparison of our method with different baselines. **Center and Right** figures show the t-SNE plots for DDA and DADEE respectively for D $\rightarrow$ E task.

variant features.

S to T	BERT	CBRT	DDA	Final	Our
Mn to Sn	80.5	79.8	81.8	83.1	**83.3**
Sn to Mn	86.9	87.1	88.5	89.3	**89.3**
R to Q	67.2	68.5	69.2	71.0	**71.1**
Q to R	65.8	66.3	67.9	68.4	**68.7**
Mr to Sc	79.5	80.1	81.5	82.9	**83.0**
Sc to Mr	85.3	85.7	86.3	87.6	**87.6**
Average	77.5	77.9	79.2	80.3	**80.5**
Avg Speed	1.00	1.68	1.32	1.00	**1.71**

Table 3: Results on NLI and entailment classification tasks. The abbreviations are MNLI (Mn), SNLI (Sn), RTE (R), QQP (Q), MRPC (M) and SciTail (Sc).

S $\rightarrow$ T	#	BERT	DAAT	UDA	Our
B $\rightarrow$ D	500	79.0	81.5	82.5	**85.8**$\pm$**1.0**
D $\rightarrow$ B	500	75.6	81.8	82.1	**85.1**$\pm$**0.7**
B $\rightarrow$ D	1000	81.5	82.0	83.3	**86.9**$\pm$**0.6**
D $\rightarrow$ B	1000	80.1	83.4	83.7	**86.5**$\pm$**0.5**
B $\rightarrow$ D	1500	84.5	86.9	87.1	**88.2**$\pm$**0.5**
D $\rightarrow$ B	1500	84.2	86.7	87.0	**87.9**$\pm$**0.2**

Table 4: This table shows the scalability as well as stability results of our method. # shows the number of samples of source dataset used to train the backbone.

4.8 Ablation study

In this section, we show the robustness of our method to source dataset size and its stability. The effect of changing the hyperparameter α is given in the Appendix A.2.

In Table 4, we demonstrate the scalability of our method by varying the size of the source dataset used for training. We use different fractions of samples as training data for the source backbone. Our findings highlight a key strength of our approach: robustness to variations in the size of the source dataset. As shown in Table 4, reducing the size of the labeled set significantly impacts other models' performance, whereas our method experiences only a slight performance decline. This robustness is attributed to the multi-level domain adaptation integrated into our framework through early exits, effectively utilizing limited labeled data and incorporating valuable representations from the training dataset. The distillation loss applied at every layer also enhances robustness, as previous methods often only attach knowledge distillation to the final layer, potentially leading to catastrophic forgetting or mode collapse. Our approach addresses this by mitigating noise accumulation across all layers.

Moreover, early exit models reduce the chances of overthinking which further helps in better results when the labeled dataset size varies as reflected in our results of table 4.

We also assess the stability of our method by calculating the standard deviation across five random runs with different seeds. Early exiting models have shown good generalization capabilities in previous works (Zhu, 2021; Zhou et al., 2020), further verifying the scalability and stability of our method. The stability performance of other methods is provided in Table 5 in the Appendix.

5 Conclusion

We present a new method DADEE for multi-level domain adaptation in PLMs using the early exit approach and adversarial training. The exits attached lower the chances of catastrophic forgetting by incorporating knowledge distillation across all layers. Also, the attached exits decrease the inference time by inferring the easier samples early. t-SNE plots and $\mathcal{A}$-distance demonstrate the effectiveness of our method in domain adaptation. Extensive experiments demonstrate that our method not only bridges the domain gap in a cross-domain setup but also provides faster inference making it suitable for real-world applications.

6 Limitations

Our model uses knowledge distillation at every layer which is a sensitive part of the method if we remove the knowledge distillation loss from a few layers, it might lead to noise accumulation from those layers as each layer is forced to give similar representations as the target domain. If a few layers face the mode collapse, it might lead to degraded performance.

Also, after adapting to the target domain, we can adapt to the threshold values based on the confidence values given by the exits at each layer so that the inference at each layer becomes more effective.

Acknowledgements

Divya Jyoti Bajpai is supported by the Prime Minister's Research Fellowship (PMRF), Govt. of India. Manjesh K. Hanawal thanks funding support from SERB, Govt. of India, through the Core Research Grant (CRG/2022/008807) and MATRICS grant (MTR/2021/000645), and DST-Inria Targeted Programme.

References

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. 2014. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*.

Divya Jyoti Bajpai and Manjesh Kumar Hanawal. 2024. Ceebert: Cross-domain inference in early exit bert. *arXiv preprint arXiv:2405.15039*.

Andrea Banino, Jan Balaguer, and Charles Blundell. 2021. Pondernet: Learning to ponder. *arXiv preprint arXiv:2107.05407*.

Ankur Bapna, Naveen Arivazhagan, and Orhan Firat. 2020. Controlling computation versus quality for neural sequence models. *arXiv preprint arXiv:2002.07106*.

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*, 79:151–175.

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.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. *Advances in neural information processing systems*, 29.

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

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

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. Depth-adaptive transformer. In *In Proc. of ICLR*.

Zhengcong Fei, Xu Yan, Shuhui Wang, and Qi Tian. 2022. Deecap: Dynamic early exiting for efficient image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12216–12226.

Xuanli He, Iman Keivanloo, Yi Xu, Xiang He, Belinda Zeng, Santosh Rajagopalan, and Trishul Chilimbi. 2021. Magic pyramid: Accelerating inference with early exiting and token pruning. *arXiv preprint arXiv:2111.00230*.

Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. 2018. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. Pmlr.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*.

Yixin Ji, Jikai Wang, Juntao Li, Qiang Chen, Wenliang Chen, and Min Zhang. 2023. Early exit with disentangled representation and equiangular tight frame. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 14128–14142.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. 2019. Shallow-deep networks: Understanding and mitigating network overthinking. In *International conference on machine learning*, pages 3301–3310. PMLR.

Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2021. I-bert: Integer-only bert quantization. In *International conference on machine learning*, pages 5506–5518. PMLR.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Shuang Li, Jinming Zhang, Wenxuan Ma, Chi Harold Liu, and Wei Li. 2021. Dynamic domain adaptation for efficient inference. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7832–7841.

Xiangyang Liu, Tianxiang Sun, Junliang He, Lingling Wu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. 2021. Towards efficient NLP: A standard evaluation and A strong baseline.

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 preprint arXiv:1907.11692*.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. 2015. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR.

Andrew 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.

Bhavitvya Malik, Abhinav Ramesh Kashyap, Min-Yen Kan, and Soujanya Poria. 2023. Udapter–efficient domain adaptation using adapters. *arXiv preprint arXiv:2302.03194*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

Quang Nguyen. 2015. The airline review dataset. *Scraped from www. airlinequality. com, https://github. com/quankiquanki/skytrax-reviews-dataset*.

Minlong Peng, Qi Zhang, Yu-gang Jiang, and Xuan-Jing Huang. 2018. Cross-domain sentiment classification with target domain specific information. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2505–2513.

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

Minho Ryu, Geonseok Lee, and Kichun Lee. 2022. Knowledge distillation for bert unsupervised domain adaptation. *Knowledge and Information Systems*, 64(11):3113–3128.

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 14*, pages 443–450. Springer.

Shengkun Tang, Yaqing Wang, Zhenglun Kong, Tianchi Zhang, Yao Li, Caiwen Ding, Yanzhi Wang, Yi Liang, and Dongkuan Xu. 2023. You need multiple exiting: Dynamic early exiting for accelerating unified vision language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10781–10791.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*.

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

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. Deebert: Dynamic early exiting for accelerating bert inference. *arXiv preprint arXiv:2004.12993*.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. Berxit: Early exiting for bert with better fine-tuning and extension to regression. In *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: Main Volume*, pages 91–104.

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

Jianfei Yu and Jing Jiang. 2016. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 236–246.

Kai Zhang, Hefu Zhang, Qi Liu, Hongke Zhao, Hengshu Zhu, and Enhong Chen. 2019. Interactive attention transfer network for cross-domain sentiment

classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5773–5780.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33:18330–18341.

Wei Zhu. 2021. Leebert: Learned early exit for bert with cross-level optimization. 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 2968–2980.

Yftah Ziser and Roi Reichart. 2018. Pivot based language modeling for improved neural domain adaptation. 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 1241–1251.

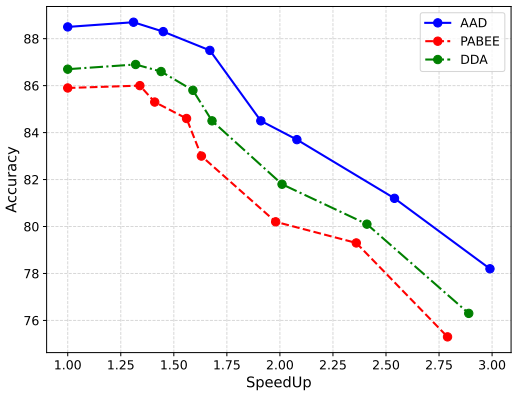


Figure 3: Speedup vs Accuracy

A Appendix

A.1 Stability

In table 5, we have shown the stability as well as scalability results of all the state-of-the-art baselines, we can observe that the stability of our method is close but slightly better than the previous methods. The reason for better stability is the better generalization achieved using the early exits.

A.2 Accuracy vs Speedup

In figure 3, we provide the results of changing the values of α used to model the accuracy efficiency trade-off. The higher values of α provide accurate predictions but with lower speedup. We have plotted the PABEE as well as our method which we get by changing the patience parameter t. Since PABEE is not adapted the performance is lower than our method. Observe that there is a slight increase in the plot which is due to the overthinking issue faced by the final layer.

A.3 Dataset statistics and # Parameters

The Amazon review dataset consists of 2000 labeled samples that are used for training and development and then the model is test split consisting of $3000-5000$ samples. For the GLUE and ELUE tasks, the dataset statistics are given in Table 6.

The number of parameters in BERT/RoBERTa-base is 110 Million and for BERT/RoBERTa-Large is 340 Million.

S → T	#	BERT	AAD	DAAT	Our
B → D	500	81.0±1.7	80.5±1.1	82.7±1.2	**85.1±1.0**
D → B	500	73.6±1.4	81.3±0.9	82.1±0.9	**83.4±0.7**
B → D	1000	79.5±1.5	82.1±0.6	82.5±0.9	**85.3±0.6**
D → B	1000	80.1±0.9	81.4±0.6	83.1±0.6	**85.7±0.5**
B → D	1500	82.5±0.5	83.4±0.4	84.2±0.7	**86.5±0.5**
D → B	1500	82.8±0.5	83.8±0.3	84.2±0.4	**84.7±0.2**

Table 5: This table shows the scalability as well as stability results of our method. # shows the number of samples of source dataset used to train the backbone.

Dataset	#Samples
MNLI	433K
MRPC	4K
SNLI	550K
QQP	365K
SciTail	24K
RTE	2.5K

Table 6: This table provides the sizes of the datasets.