CEEBERT: Cross-Domain Inference in Early Exit BERT

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

Abstract

Pre-trained Language Models (PLMs), like BERT, with self-supervision objectives exhibit remarkable performance and generalization across various tasks. However, they suffer in inference latency due to their large size. To address this issue, side branches are attached at intermediate layers, enabling early inference of samples without requiring them to pass through all layers. However, the challenge is to decide which layer to infer and exit each sample so that the accuracy and latency are balanced. Moreover, the distribution of the samples to be inferred may differ from that used for training necessitating cross-domain adaptation. We propose an online learning algorithm named Cross-Domain Inference in Early Exit BERT (CEEBERT) that dynamically determines early exits of samples based on the level of confidence at each exit point. CEEBERT learns optimal thresholds from domain-specific confidence observed at intermediate layers on the fly, eliminating the need for labeled data. Experimental results on five distinct datasets with BERT and ALBERT models demonstrate CEEBERT's ability to improve latency by reducing unnecessary computations with minimal drop in performance. By adapting to the threshold values, CEEBERT can speed up the BERT/ALBERT models by $2\times$ - $3.5\times$ with minimal drop in accuracy. The source code is available at https://github.com/Div290/CeeBERT.

1 Introduction

In recent years, Pre-trained Language Models (PLMs) have demonstrated substantial advancements in enhancing Natural Language Processing (NLP) tasks. These models, such as ELMo (Peters et al., 1802), BERT (Devlin et al., 2018), ALBERT (Lan et al., 2019), GPT (Radford et al., 2019), XLNet (Yang et al., 2019), and RoBERTa (Liu et al., 2019), encapsulate extensive knowledge transferable to diverse downstream tasks. Despite their remarkable efficacy, large-scale PLMs suffer from inference latencies attributed to their substantial size. With millions or even billions of parameters, these models become computationally expensive, inefficient in terms of memory consumption, and exhibit latency challenges.

Moreover, past approaches like (Fan et al., 2019; Michel et al., 2019; Zhou et al., 2020) have highlighted the concern that PLMs tend to be overparameterized, leading to the 'overthinking' issue. This problem arises when shallow representations at early layers are often adequate for accurate predictions on many input samples, while the final layer's representations may suffer distractions due to over-complicated or irrelevant features that lack generalization. Overthinking in PLMs wastes the computation, degrades model performance and hinders model generalization.

To circumvent this, many variants of the BERT, like DeeBERT (Xin et al., 2020a), ElasticBERT (Liu et al., 2021a), FastBERT (Liu et al., 2020), PABEE (Zhou et al., 2020), etc. facilitate inference at the intermediary layers of PLMs through early exits. In this setup, each sample must ascertain whether the inference can be completed at intermediary layers or the last layer. The decisions of early exit are based on the confidence at the intermediary layers being above a threshold. In the following, we consider the cost quantified as inference time (latency). However, depending on the application, the cost can also be present as other factors like power and computational resources.

The threshold used to compare the confidence levels significantly impacts the amount of latency and accuracy: with a lower threshold, more samples exit early, but with a lower confidence value, leading to lower accuracy and lower latency. With a higher threshold, fewer samples exit early, leading to higher latency but improved accuracy.

Hence, one has to set the threshold that optimally trades-off between latency and accuracy.

The threshold is often determined using a labeled dataset (Xin et al., 2020a; Liu et al., 2021b; Schuster et al., 2021) or by using some fraction of training data (Schwartz et al., 2020; Huang et al., 2017; Yang et al., 2020; Han et al., 2023) during training and serves as a crucial reference point for decision-making during inference. However, a significant challenge arises when deploying pre-trained models that are later tested on samples whose latent distribution can be different from the training samples in a zero-shot setting (Pushp and Srivastava, 2017; Wang et al., 2023). For instance, a sentiment analysis model pre-trained on electronic product reviews (source data) to analyze sentiment in a distinct domain, like movie reviews (target data), poses a challenge. The language and sentiment expressions in movie-related content may significantly differ from electronic reviews as illustrated in Fig 1.

PLMs exhibit strong generalization across domains on datasets with similar task types (Wang et al., 2023). Nevertheless, the distribution of confidence in the prediction of classifiers attached to the intermediary layers can change when transitioning between domains. This can render early exit methods less efficient if the thresholds are not adjusted as per the latent distribution of input samples. This real-world challenge prompts the question: How to adapt the threshold in deployed early exit PLMs to maintain efficiency and robustness to domain shifts in a zero-shot setting? Also, in post-deployment scenarios, samples are fed sequentially, in an online fashion, where inference for each sample needs to be performed before the next sample is fed, this leads to the challenge of learning the optimal threshold in an online and unsupervised manner. To tackle this issue, we introduce an online learning algorithm using the Multi-Armed Bandit framework. (Auer et al., 2002b).

Our algorithm, <u>C</u>ross Domain Inference in <u>E</u>arly <u>E</u>xits <u>BERT</u> (CEEBERT), learns a threshold from a set of thresholds that achieves optimal trade-off between accuracy and latency. We extensively evaluate the performance of CEEBERT on five datasets viz. IMDB, MRPC, SciTail, SNLI, Yelp, and QQP cover different classification tasks –sentiment, entailment, and natural language inference. In our evaluation, we trained the PLM with exits on a source dataset and assessed its performance on a target dataset which exhibits variation in the latent data distribution from the source dataset but shares a similar task type.

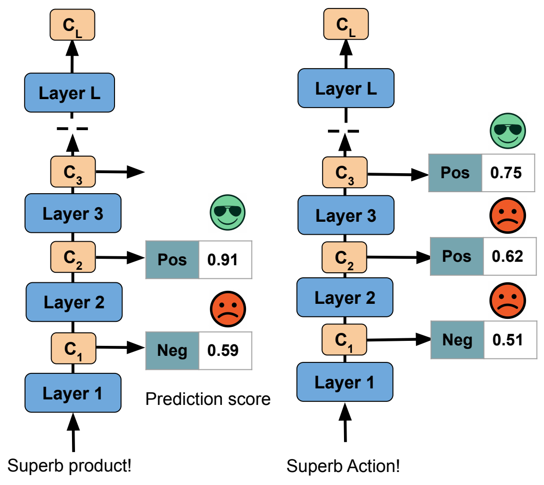


Figure 1: Depiction of cross-domain inference setup for early exit models. The backbone is trained on the source dataset. i) **Left:** Inference on the source dataset. ii) **Right:** Inference on target dataset from a different domain. The changes in confidence value is due to change in target domain distribution.

CEEBERT achieves speedup ranging from $2\times$ to $3.5\times$ in inference time, while maintaining a minimal accuracy loss of 0.1% to 3% compared to the naive BERT classification. Notably, instead of using pre-fixed thresholds learned on the source dataset, CeeBERT learns the thresholds on the fly for incoming samples from the target dataset. This sets it apart from previous early exit PLMs that rely on validation datasets representative of the target dataset to determine threshold values.

Our primary contributions are as follows:

- We introduce cross-domain adaptation in Early Exit PLMs where the thresholds of exit classifiers are adapted without requiring any validation dataset from the target task as used in the previous baselines (see Section 5.2), thus making Early Exit PLMs robust to domain changes and enhancing their efficiency.

- In Section 4, we introduce CEEBERT, an upper confidence-based algorithm that relies solely on confidence scores for learning the optimal threshold to make exit decisions.

- In Section 5, we evaluate CEEBERT on five different datasets drawn from diverse tasks to demonstrate its proficiency in improving inference latency compared to the state-of-the-art algorithms.

2 Related work

Early exit methods are applied for various tasks such as image classification, image captioning and NLP tasks to reduce the computational resources and inference latency.

Early exits in Image tasks: For image classification tasks, BranchyNet (Teerapittayanon et al., 2016) uses classification entropy at each attached exit to decide whether to infer the sample at the side branch based on the entropy of prediction. Shallow-deep(Kaya et al., 2019) and MSDNet (Huang et al., 2017) improve upon BranchyNet by effectively choosing the thresholds based on the confidence distribution. Similar architectures (Laskaridis et al., 2020; Pacheco et al., 2021; Dai et al., 2020) split the NN to be deployed on edge and cloud. SEE (Wang et al., 2019c) work in service outage scenarios. FlexDNN (Fang et al., 2020) and Edgent (Li et al., 2019) focus mainly on the most appropriate Neural Network (NN) depth. Other works such as Dynexit (Wang et al., 2019b) focus on deploying the multi-exit NN in hardware. It trains and deploys the NN on Field Programmable Gate Array (FPGA) hardware while Paul *et al.* (Kim and Park, 2020) explains that implementing a multi-exit NN on FPGA board reduces inference time and energy consumption. In a parallel vein, the MuE and DeeCap (Tang et al., 2023; Fei et al., 2022) model employs a distinctive approach to apply early exits to the image captioning. DeeCap only applies to to decoder while MuE applies to the encoder as well as the decoder.

Early Exit in PLMs: Multiple approaches have been proposed to effectively apply early exits to PLMs and solve multiple NLP tasks (Bapna et al., 2020; Elbayad et al., 2020; Liu et al., 2021b; Xin et al., 2020b; Zhou et al., 2020; He et al., 2021; Banino et al., 2021; Balagansky and Gavrilov, 2022; Sun et al., 2022; Ji et al., 2023). DeeBERT (Xin et al., 2020b), ElasticBERT (Liu et al., 2021b) and BERxiT (Xin et al., 2021) are based on the transformer-based (Vaswani et al., 2017) BERT model. BERxiT proposes an efficient fine-tuning strategy for the BERT model with attached exits. DeeBERT is obtained by training the exit points attached before the last module to the BERT backbone separately. In contrast, ElasticBERT is obtained by training all the exit points attached to the BERT backbone jointly. PABEE (Zhou et al., 2020) is another multi-exit model that makes the exit decision based on the stability of the predictions after different exits. LeeBERT (Zhu, 2021) proposed a self-distillation framework that has similar exiting criteria as PABEE. ETFEE (Ji et al., 2023) adds an adapter on top of the transformer layers and an (Entangled Frame) ETF classifier to make intermediate exits learn better.

Multi-Armed Bandits in Early Exit NN: LEE (Ju et al., 2021b), DEE (Ju et al., 2021a) and UEE-UCB (Hanawal et al., 2022) leverage the MAB framework to learn the optimal exit in early exit NNs. LEE and DEE mainly focus on learning optimal depth in image classification tasks, while UEE-UCB finds optimal depth for NLP tasks employing a pre-trained ElasticBERT (Liu et al., 2021a) model. UEE-UCB does not need any labels but works under the assumption that the prediction of the intermediary layers follows strong dominance property (Verma et al., 2019). SplitEE (Bajpai et al., 2023) and I-SplitEE (Bajpai et al., 2024) utilize MABs to find the optimal splitting layers in a mobile-cloud co-inference setup.

Our approach differs from past works as 1) Unlike previous studies, our work is primarily concerned with making unsupervised cross-domain inference efficient in early exit PLMs. 2) Our work is focused on adapting the threshold values based on the underlying distribution of a dataset. 3) We use the Multi-Armed Bandits framework to solve the problem of threshold learning. We compare against different early exiting models in table 2.

3 Problem Setup

We start with a pre-trained PLM like BERT or ALBERT and attach exit classifiers at each layer. In the following, we discuss how the exit layers are trained and used for early inference.

3.1 Training exits classifers

Let $\mathcal{D}$ represent the distribution of the source dataset with label class $\mathcal{C}$ used for backbone training. For any input sample $(x, y) \sim \mathcal{D}$ and the ith intermediate classifier, the loss is computed as:

$$\mathcal{L}_i(\theta) = \mathcal{L}_{CE}(f_i(x, \theta), y) \tag{1}$$

Here, $f_i(x, \theta)$ is the output of the classifier attached at the ith layer, where θ denotes the set of learnable parameters, and $\mathcal{L}_{CE}$ is the cross-entropy loss. We learn the parameters for all classifiers simultaneously following the approach outlined by Kaya et al. (Kaya et al., 2019), with the loss function defined as $\mathcal{L} = \frac{\sum_{i=1}^{L} i \cdot \mathcal{L}i}{\sum_{i=1}^{L} i}$, where L

is the number of layers in the backbone. This weighted average considers the relative inference cost of each internal classifier. Subsequently, the model is ready for testing on related tasks from different domains.

3.2 Inference on the Target data

Let $\tilde{\mathcal{D}}$ be the distribution of the target dataset. Consider an intermediary layer $1 \leq i < L$. For an input $x \sim \tilde{\mathcal{D}}$, let $\hat{P}_i(c)$ denote the estimated probability that x belongs to class $c \in \mathcal{C}$ and C_i denote the confidence in the estimate at the ith layer. We define C_i as maximum of the estimated probability class, i.e., $C_i := \max_{c \in \mathcal{C}} \hat{P}_i(c)$. The decision to exit at the ith layer is made based on the value of C_i. For a given threshold α, if $C_i \geq \alpha$ the sample x will be assigned a label $\hat{y} = \arg\max_{c \in \mathcal{C}}(\hat{P}_i(c))$. In this case, x is not further processed, and *exits* with a label $\hat{y}$. If $C_i < \alpha$, then the sample is processed to the next layer. If the sample's confidence is below the threshold for all intermediate classifiers then the sample is inferred at the final layer. We denote the cost incurred in moving the sample from the 1st layer to the ith layer as o_i. It denotes the latency or computational cost of processing the sample between the layers 1 and i. Since all the layers of transformer-based PLMs require the same amount of computation, we assume the latency cost $o_i \propto i$.

The confidence can be compared against one of the k possible thresholds denoted by set $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$. The goal is to identify the threshold which provides the best trade-off between loss in accuracy and latency cost for the latent distribution of the target task.

3.3 Multi-Armed Bandit (MAB) Setup

In the MAB setup, a decision-maker iteratively selects actions, adapting to an unknown environment. Each action represents a specific choice, and the goal is to learn which actions result in the most favourable outcomes (highest reward) over time. This dynamic learning process, inherent to MAB setups, resonates with the sequential decision-making required in online learning problems, where choices are made based on incoming data. MAB frameworks, guided by exploration and exploitation principles, enable the learning of optimal actions tailored to the latent distribution.

We treat the set of thresholds $\mathcal{A}$ as the set of actions. Following the terminology of MAB, we refer to them as arms. We define $[L] = \{1, 2, \ldots, L\}$. For any arm $\alpha \in \mathcal{A}$, suppose a sample is processed till layer i and exits i.e. $C_j < \alpha$ for $j \in [i-1]$ and $C_i \geq \alpha$, we define the reward as follows:

$$r(\alpha) = (C_i - C_1) - \mu o_i \tag{2}$$

where μ models the trade-off between accuracy and cost and doubles up as a unit converter to bring both confidence and latency in the same units. If the sample does not exit till $(L-1)$th layer then it is inferred at the final layer, where the reward is $r(\alpha) = (C_L - C_1) - \mu o_L$.

The reward could be interpreted as follows: The confidence gained while processing the sample from 1st layer to ith layer reduced by the cost incurred in achieving it (latency). Then mean reward for arm $\alpha \in \mathcal{A}$ is

$$\mathbb{E}[r(\alpha)] = \sum_{i=1}^{L} \mathbb{E}[\Delta C_i - \mu o_i | \text{exit at } i] P(i) \tag{3}$$

where $\Delta C_i = C_i - C_1$ and $P(i)$ is the probability that the sample exits from ith layer. Our goal is to find an arm with the highest mean reward. Since labels of the target samples are not available, we depend on the reward of each threshold to learn their performance. Let $\alpha^* = \arg\max_{\alpha \in \mathcal{A}} \mathbb{E}[r(\alpha)]$ denote the optimal threshold. Consider a policy π that selects threshold $\alpha_t \in \mathcal{A}$ in round t based on past observations. We define cumulative regret of π over T rounds as

$$R(\pi, T) = \sum_{t=1}^{T} \mathbb{E}[r(\alpha^*) - r(\alpha_t)], \tag{4}$$

where the expectation is with respect to the randomness in the selection of thresholds induced by the past samples. A policy π^* is said to be sub-linear if average cumulative regret vanishes, i.e., $R(\pi^*, T)/T \to 0$. Our objective is to develop a policy learning algorithm with a sub-linear regret guarantee.

4 Algorithm

We develop an algorithm named Cross Domain Inference in Early Exits in BERT (CeeBERT). Its pseudo-code is given in algorithm 1. The inputs to the algorithm are exploration constant γ and latency factor o_i for each layer i. For the first $|\mathcal{A}|$ samples, the algorithm plays each arm once. In the subsequent rounds, it plays the arm with the highest Upper Confidence Bound (UCB) index denoted as β_t. UCB indices are obtained by taking the weighted sum of the empirical average of the rewards $Q_t(\alpha)$ and the confidence bonuses with γ as the weight factor. If C_i at the ith layer is larger than β_t then the sample exits, otherwise, the sample is passed to the next layer in the backbone while adding latency. If the sample does not exit at any intermediate classifier then it is inferred at the final layer. Finally, the algorithm updates the number of pulls ($N(\beta_t)$) and empirical mean ($Q(\beta_t)$) of the played arm. Note that the algorithm is applied directly to the inference (target) dataset.

Algorithm 1 CeeBERT

```
1:  Input: $o_i$ $\forall i, \gamma \geq 1$
2:  Initialize: Play each threshold once. Observe
    $r(\alpha)$ and set $Q(\alpha) \leftarrow \mathbf{0}, N(\alpha) \leftarrow \mathbf{1}, \forall \alpha \in \mathcal{A}$.
3:  for $t = |\mathcal{A}| + 1, |\mathcal{A}| + 2, \cdots$ do
4:      Observe an instance $x_t$
5:      $\beta_t \leftarrow \arg\max_{\alpha \in \mathcal{A}} \left( Q(\alpha) + \gamma \sqrt{\frac{\ln(t)}{N(\alpha)}} \right)$
6:      for $i = 1$ to $L$ do
7:          Pass $x_t$ till layer $i$
8:          Apply threshold $\beta_t$ and observe $C_i$
9:          if $C_i \geq \beta_t$ and $i < L$ then
10:             Infer at layer $i$ and exit
11:             $r_t(\alpha) \leftarrow C_i - C_1 - o_i$
12:             $N_t(\alpha) \leftarrow N_{t-1}(\alpha) + 1$
13:             $Q_t(\alpha) \leftarrow \frac{\sum_{j=1}^{t} r_j(\alpha_j) \mathbb{1}_{\{\alpha_j = \alpha\}}}{N_t(\alpha)}$
14:             break
15:         else if $i = L$ then
16:             Process and infer at the last layer.
17:             $r_t(\alpha) \leftarrow (C_l - C_p - o)$
18:             $N_t(\alpha) \leftarrow N_{t-1}(\alpha) + 1$
19:             $Q_t(\alpha) \leftarrow \frac{\sum_{j=1}^{t} r_j(\alpha_j) \mathbb{1}_{\{\alpha_j = \alpha\}}}{N_t(\alpha)}$
20:         end if
21:     end for
22: end for
```

Following the analysis of UCB1 (Auer et al., 2002b), one can show that the regret of CeeBERT is of $\mathcal{O}\left(\sum_{\alpha \in \mathcal{A}_p \setminus \alpha^*} \frac{log(n)}{\Delta_\alpha}\right)$ where $\Delta_\alpha = r(\alpha^*) - r(\alpha)$ denotes the optimality gap. For completeness, the proof outline is given in the Appendix (see theorem A.1). Hence, CeeBERT comes with a sub-linear regret guarantee.

Tgt data	#Samples	Src Data	#Samples
IMDb	25K	SST-2	68K
Yelp	560K	SST-2	68K
SNLI	550K	MNLI	433K
QQP	365K	MRPC	4K
SciTail	24K	RTE	2.5K

Table 1: This table provides the sizes of the datasets. Src (Source data) is used to train the model to test on Tgt (target data) to evaluate its generalization.

5 Experiments

5.1 Datasets

We utilized most of the GLUE (Wang et al., 2019a) datasets as source datasets and the ELUE (Liu et al., 2021a) datasets as the target datasets. We evaluated CeeBERT on five datasets covering four types of classification tasks. The datasets used for evaluation are:

1) IMDb and 2) Yelp (Asghar, 2016): IMDb is a movie review classification dataset and Yelp consists of reviews from various domains such as hotels, restaurants etc. The source dataset for these two datasets is the **SST-2** (Stanford-Sentiment Treebank) dataset which has a sentiment classification task. **3) SciTail:** is an entailment classification dataset created from multiple questions from science and exams and web sentences. The source data used was **RTE**(Recognizing Textual Entailment) dataset which is an entailment classification dataset but with a different context. **4) SNLI(Stanford Natural Language Inference:)** is a collection of human-written English sentence pairs manually labelled for balanced classification with labels *entailment*, *contradiction* and *neutral*. The source data in this case is **MNLI**(Multi-Genre Natural Language Inference) which also contains sentence pairs as premise and hypothesis, the task is the same as for SNLI but with more general sentences. **5) QQP(Quora Question Pairs)** is a semantic equivalence classification dataset which contains question pairs from the community question-answering website Quora. In this case, the source dataset is **MRPC**(Microsoft Research Paraphrase Corpus) dataset which also has a semantic equivalence task of a sentence pair extracted from online news sources. Details about the size of these datasets are in table 1. Observe from the table that the size of the source dataset is much smaller as compared to the size of the corresponding target dataset.

5.2 Baselines

We compare our model against three types of baselines:

1) Backbone models: We choose BERT-base and ALBERT-base as the backbone models which have almost similar performance[1].

2) Reducing layers: We directly reduce computation layers, experimenting with the initial 6 and 9 layers of the original (AL)BERT model, denoted as (AL)BERT-6L and (AL)BERT-9L. These serve as baselines, setting the lower limit for techniques without additional modifications.

3) Early-exit models: **DeeBERT** (Xin et al., 2020a) and 2) **ElasticBERT** (Liu et al., 2021a) employ fixed confidence thresholds for early exits. 3) **FastBERT** (Liu et al., 2020) utilizes a self-distillation framework to train the intermediate exits. 4) **PABEE** (Zhou et al., 2020) and 5) **LeeBERT** (Zhu, 2021) uses prediction stability to decide early exits, LeeBERT also distils the knowledge from deeper layers. 6) **MuE** (Tang et al., 2023) relies on hidden representation similarity for early exit decisions and is applied to the BERT-base model for comparative analysis, originally designed for image captioning tasks. 7) **ETFEE** (Ji et al., 2023) and 8) **PALBERT** (Balagansky and Gavrilov, 2022), state-of-the-art methods, face challenges to adapt to different domains due to bias towards the training dataset. PALBERT uses Lambda layers as explained on (Banino et al., 2021), and ETFEE has an adapter on top of intermediate layers, both contributing to strong bias. Test data must be representative of the training dataset for these baselines.

All the baselines learn the threshold values to decide to exit using a validation dataset representative of the training dataset. Other hyperparameters for these baseline models remain consistent with their original implementations, and when applied to the target dataset, we use the same hyperparameters learned on the source dataset.

5.3 Experimental setup

i) Training of the backbone on source data: Initially, we train the backbone on the source dataset. We add a linear output layer after each of the intermediate layers of the pre-trained BERT/ALBERT model. We run the model for 5 epochs. We perform a grid search over batch size of $\{8, 16, 32\}$ and learning rates of $\{$1e-5, 2e-5, 3e-5, 4e-5 5e-5$\}$ with Adam (Kingma and Ba, 2014) optimizer. We apply an early stopping mechanism and select the model with the best performance on the development set. The experiments are conducted on NVIDIA RTX 2070 GPU with an average runtime of $\sim$ 3 hours and a maximum run time of $\sim$ 10 hours for the MNLI dataset.

ii) Adapting thresholds using CeeBERT: In this stage, use the model learnt in step (i) to perform inference on the target dataset. In this step, CeeBERT is utilized to dynamically learn optimal thresholds in an unsupervised and online manner for the target dataset. This post-deployment step allows the model to autonomously adapt threshold values based on real-time data, enhancing adaptability in inference. This part is also computed on the same GPU with an average runtime of $\sim$ 1 hour. We run each experiment 5 times where each run includes an online feed of randomly reshuffled input samples to CeeBERT.

Choice of the action set: The choice of the action set depends on the total number of output classes, denoted as C, within a given dataset. To ensure efficiency and avoid redundancy, we observe that any value in the action set below $1/C$ is extraneous. Consequently, we choose ten equidistant values ranging from $1/C$ to 1.0. For instance, in a binary classification scenario where the worst confidence value is 0.5, our action set becomes $\mathcal{A} = \{0.55, 0.6, 0.65, \ldots, 0.95, 1.0\}$.

Latency cost and μ: Recall from section 3.2, we have $o_i \propto i$ i.e. the latency cost for each layer is directly proportional to the depth of the layer in the network. Hence we set $o_i = \lambda i$ where λ is the per-layer processing cost. The value of λ is user-defined and we set it to $1/L$ so that it is directly comparable to the confidence values. The parameter μ used to model the trade-off between accuracy and latency is set to 0.5. The value of μ should be set between $\mu \in [0, 1/o_L]$ based on the user's preferences for improved cost or accuracy. The set of choices of μ is made such that the factor μo_i in the equation 2 is directly comparable to confidence gain. We analyse the model behaviour on changing μ in section 6.1.

To maintain consistency with previous methods, we use the speedup ratio as the metric to asses out 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

[1]Our method works on any transformer-based PLMs

Model/Data	SST-IMDb		SST-Yelp		MRPC-SciTail		MNLI-SNLI		RTE-QQP	
	Acc	Speed	Acc	Speed	Acc	Speed	Acc	Speed	Acc	Speed
BERT	83.3	1.00x	77.8	1.00x	79.1	1.00x	80.2	1.00x	71.5	1.00x
BERT-6L	-3.1	2.00x	-3.0	2.00x	-1.6	2.00x	-1.9	2.00x	-1.5	2.00x
BERT-9L	-2.6	1.33x	-2.8	1.33x	-0.9	1.33x	-1.3	1.33x	-1.2	1.33x
DeeBERT	-2.9	2.31x	-3.5	2.13x	-0.5	1.21x	-2.5	2.11x	-0.9	1.35x
ElasticBERT	-2.6	2.51x	-3.2	2.95x	0.0	1.49x	-1.3	2.32x	-0.3	1.80x
PABEE	-2.4	2.42x	-2.9	2.56x	-0.9	1.51x	-1.1	2.39x	-0.2	1.91x
FastBERT	-2.5	2.55x	-2.8	2.62x	-0.6	1.59x	-1.3	2.45x	-0.5	1.96x
MuE	-2.8	2.58x	-3.3	2.81x	-1.1	1.75x	-1.6	2.52x	-0.2	1.88x
LeeBERT	**-2.3**	2.40x	-2.7	2.49x	-0.1	1.54x	-1.0	2.37x	0.0	1.92x
PALBERT	-2.5	2.51x	**-2.4**	2.39x	-0.7	1.63x	-1.4	2.54x	-0.1	1.85x
ETFEE	-2.6	2.45x	-2.5	2.54x	-0.6	1.67x	-1.8	2.55x	-0.3	1.96x
CeeBERT	**-2.3**	**2.95x**	**-2.4**	**3.15x**	**+0.2**	**1.78x**	**-0.8**	**2.63x**	**+0.1**	**2.15x**

Table 2: Experimental results (median of 5 runs) of early exit models with BERT backbone on the target datasets with 5 random seeds. The format of datasets on top of the table is (source-target) i.e. the dataset before the hyphen is the source and after the hyphen is the target. The accuracy (Acc) is in % and speed is the Speedup ratio.

ith layer. This metric could be interpreted as the increase in speed of the model as compared to the naive (AL)BERT model.

5.4 Results

In Tables 2 and 4, we provide the main results of this paper. We provide median results over 5 runs. The results make evident that CeeBERT consistently outperforms all previous methods both in terms of accuracy and efficiency, due to its ability to adapt and select different thresholds for tasks from different domains.

This behaviour aligns with the fact that as the target dataset is from a different domain, there is a change in the semantic mapping. This change in turn differs the confidence distribution at the exit points. Previous baselines do not adapt the threshold based on the distribution of target data, hence all the methods get a hit in terms of efficiency. We observe that models trained with more bias toward the training dataset get higher hits in performance, as seen in the current state-of-the-art methods ETFEE and PALBERT. They experience significant performance degradation. This occurs due to the addition of more complex layers (instead of linear layers) and classifiers to the output of intermediate layers, leading to the requirement of the test set to be representative of the training dataset.

CeeBERT's dynamic threshold selection during inference, without requiring full retraining or fine-tuning, offers a substantial advantage. This adaptability speeds up the inference time on average by 2.5x while preserving accuracy. Notably, CeeBERT achieves this without relying on labeled data; instead, it makes real-time threshold decisions based on evolving data distributions. CeeBERT converges to optimal thresholds after just a few thousand samples, as shown in figure 2c.

The gain on QQP and SciTail datasets as compared to the final layer is explained as the effect of overthinking. Some easy samples get misclassified at the final layer as they suffer from distractions due to over-complicated features available at the final layer. More details on the stability of results i.e. the standard deviation values can be found in the table 3 in the Appendix.

6 Ablation study and Analysis

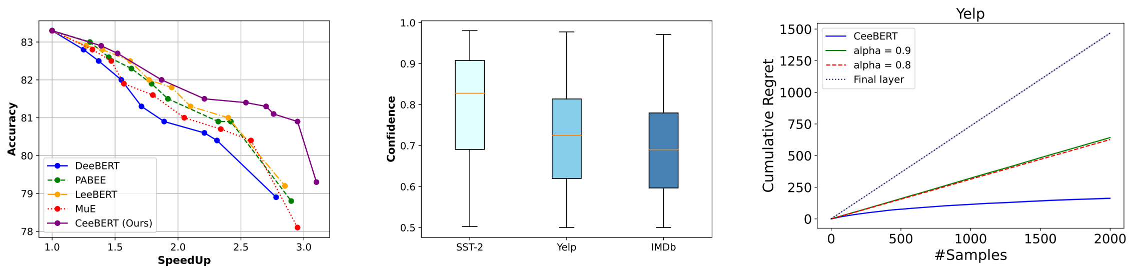
6.1 Accuracy vs Speedup

In figure 2a, we analyse the behaviour of performance drop over an increase in speedup. DeeBERT and MuE make exit decisions based on confidence values above a pre-defined threshold. PABEE and LeeBERT both use patience-based exiting. By varying the threshold values, we observe variation in the accuracy-latency trade-off. CeeBERT uses the trade-off factor μ as given in equation 2. Increasing μ will increase the impact of cost in the reward hence it will choose lower thresholds forcing samples to exit early while decreasing it increases the impact of confidence hence improving upon accuracy. We vary these hyperparameters to get the results in figure 2a.

Observe that, adapting thresholds has minimal impact if the confidence is given higher weight in eq.2 i.e. μ is small since it boils down to the case in which confidence is maximized hence many samples are inferred at the final layer similar to other baselines. However, as the impact of cost increases the efficiency of CeeBERT also increases as it forces the samples to exit early reducing overthinking as well as unnecessary computations. It

Model/Data	SST-IMDb		SST-Yelp		MRPC-SciTail	
	Acc	Speed	Acc	Speed	Acc	Speed
BERT	83.3	1.00x	77.8	1.00x	79.1	1.00x
CeeBERT	**-2.3** $\pm$ **0.15**	**2.95x** $\pm$ **0.008**	**-2.4** $\pm$ **0.05**	**3.15x** $\pm$ **0.003**	**+0.2** $\pm$ **0.18**	**1.78x** $\pm$ **0.001**
ALBERT	82.7	1.00x	77.1	1.00x	80.4	1.00x
CeeBERT	**-2.1** $\pm$ **0.12**	**2.89x** $\pm$ **0.006**	**-1.9** $\pm$ **0.08**	**2.71x** $\pm$ **0.002**	**-0.1** $\pm$ **0.16**	**1.81x** $\pm$ **0.004**

Table 3: The median and standard deviation values of CeeBERT over 5 runs.



(a) Speedup vs Accuracy curve (b) Change in Confidence distribution (c) Cumulative regret

Figure 2: **Left:** We show the trade-off between accuracy and speedup by changing the tunable parameters for various methods. **Center:** figure states the change in the distribution of confidence at the intermediate exits when the dataset distribution changes. The backbone was trained on SST-2 in this case. **Right:** figure shows the cumulative regret observed by CeeBERT on the Yelp dataset showing that CeeBERT achieves sub-linear regret.

also states that adapting the threshold values while considering both cost and confidence can make early exit models more efficient.

6.2 Change in confidence distribution

In figure 2b, we plot the confidence distribution of SST-2, IMDb and Yelp datasets when the backbone was trained on SST-2. We plot the boxplots for confidence values for the SST-2 dataset, IMDb and Yelp datasets. The confidence distribution evolves with shifts in the latent distribution of the dataset. Our observations reveal a decrease in the confidence distribution, rendering the threshold learned on the source data less adaptable to the target dataset. This diminished adaptability introduces inefficiencies in the threshold application. The decline in confidence values can be attributed to the distinct semantic structures present in the target dataset compared to the source data. Despite the generalization capabilities of the BERT model to accommodate these changes, a reduction in confidence values becomes evident.

6.3 Regret Performance

In figure 2c, we plot the average cumulative regret on the Yelp dataset over 5 runs where the samples were randomly reshuffled and fed to the algorithm. This figure gives an idea about the time it takes for CeeBERT to converge for a given dataset. CeeBERT converges to the optimal threshold after exploring on few thousand samples. The plot also contains the cumulative regret when all the samples exited from the final layer and when the thresholds were not adapted and were fixed (same as previous methods). For fixed thresholds, we use $\alpha = 0.5, 0.8, 0.9$. This indicates the importance of learning the threshold values instead of fixing them. For other datasets regret curves refer to the Appendix (figure 3).

6.4 Stability of CeeBERT

In table 3, we provide the standard deviation values of 5 runs of CeeBERT on different datasets. These results state the stability of the algorithm. CeeBERT has good stability as it converges fast within the first thousand samples and then the exploration phase is over, now it exploits the optimal threshold to which it has converged. As observed, CeeBERT's stability increases with large datasets as on Yelp the standard deviation values are lower as compared to other datasets. On larger datasets, the fraction of samples over which CeeBERT explores becomes small which in turn increases the stability of the algorithm. The stability is consistent across different backbone models (BERT and ALBERT).

7 Conclusion

In this work, we introduced the concept of cross-domain inference in Early exiting PLMs. We pro-

Model/Data	SST-IMDb		SST-Yelp	
	Acc	Speed	Acc	Speed
ALBERT	82.7	1.00x	77.3	1.00x
ALBERT-6L	-3.5	2.00x	-3.1	2.00x
ALBERT-9L	-2.8	1.33x	-2.4	1.33x
DeeBERT	-3.2	2.39x	-2.9	2.24x
ElasticBERT	-2.7	2.47x	-2.6	2.41x
PABEE	-2.4	2.42x	-2.3	2.35x
FastBERT	-2.3	2.55x	-2.5	2.32x
MuE	-2.5	2.61x	-2.8	2.51x
LeeBERT	-2.2	2.21x	**-1.9**	2.49x
PALBERT	-2.7	2.47x	-2.6	2.53x
ETFEE	-2.8	2.65x	-2.9	2.62x
CeeBERT	**-2.1**	**2.89x**	**-1.9**	**2.71x**

Table 4: Experimental results (median of 5 runs) of early exit models applied to ALBERT backbone.

posed an online learning algorithm, CeeBERT that enhances the efficiency of early exit PLMs by adjusting threshold values as per the latent distribution of the incoming data. This adaptation ensures robust performance even in the face of dataset variations within a specific task but across diverse domains. The resulting robustness mitigates the need for any further fine-tuning that reduces time as well as resources.

8 Limitations

In this work, we have used the same threshold for each exit point. However, one can extend this to having different thresholds at different exit points. However, it comes at the cost of the added complexity of having separate action sets for different layers. It would be interesting to explore and reduce the complexities and further reduce the latency as then the choice of threshold will be made separately for each exit. CeeBERT could be applied to any early exit NN with minor modifications, however, in the experiments, we apply it on (AL)BERT models as done by previous methods.

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

Nabiha Asghar. 2016. Yelp dataset challenge: Review rating prediction. *arXiv preprint arXiv:1605.05362*.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002a. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256.

Peter Auer et al. 2002b. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256.

Divya J Bajpai, Vivek K Trivedi, Sohan L Yadav, and Manjesh K Hanawal. 2023. Splitee: Early exit in deep neural networks with split computing. *arXiv preprint arXiv:2309.09195*.

Divya Jyoti Bajpai, Aastha Jaiswal, and Manjesh Kumar Hanawal. 2024. I-splitee: Image classification in split computing dnns with early exits. *arXiv preprint arXiv:2401.10541*.

Nikita Balagansky and Daniil Gavrilov. 2022. Palbert: Teaching albert to ponder. *Advances in Neural Information Processing Systems*, 35:14002–14012.

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*.

Xin Dai, Xiangnan Kong, and Tian Guo. 2020. Epnet: Learning to exit with flexible multi-branch network. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 235–244.

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*.

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

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*.

Biyi Fang, Xiao Zeng, Faen Zhang, Hui Xu, and Mi Zhang. 2020. Flexdnn: Input-adaptive on-device deep learning for efficient mobile vision. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 84–95. IEEE.

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.

Yizeng Han, Dongchen Han, Zeyu Liu, Yulin Wang, Xuran Pan, Yifan Pu, Chao Deng, Junlan Feng, Shiji Song, and Gao Huang. 2023. Dynamic perceiver for efficient visual recognition. *arXiv preprint arXiv:2306.11248*.

Manjesh K Hanawal, Avinash Bhardwaj, et al. 2022. Unsupervised early exit in dnns with multiple exits. *arXiv preprint arXiv:2209.09480*.

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*.

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.

Weiyu Ju, Wei Bao, Liming Ge, and Dong Yuan. 2021a. Dynamic early exit scheduling for deep neural network inference through contextual bandits. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 823–832.

Weiyu Ju, Wei Bao, Dong Yuan, Liming Ge, and Bing Bing Zhou. 2021b. Learning early exit for deep neural network inference on mobile devices through multi-armed bandits. In *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 11–20. IEEE.

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.

Geonho Kim and Jongsun Park. 2020. Low cost early exit decision unit design for cnn accelerator. In *2020 International SoC Design Conference (ISOCC)*, pages 127–128. IEEE.

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

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Stefanos Laskaridis, Stylianos I Venieris, Mario Almeida, Ilias Leontiadis, and Nicholas D Lane. 2020. Spinn: synergistic progressive inference of neural networks over device and cloud. In *Proceedings of the 26th annual international conference on mobile computing and networking*, pages 1–15.

En Li, Liekang Zeng, Zhi Zhou, and Xu Chen. 2019. Edge ai: On-demand accelerating deep neural network inference via edge computing. *IEEE Transactions on Wireless Communications*, 19(1):447–457.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. 2020. Fastbert: a self-distilling bert with adaptive inference time. *arXiv preprint arXiv:2004.02178*.

Xiangyang Liu, Tianxiang Sun, Junliang He, Lingling Wu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. 2021a. Towards efficient nlp: A standard evaluation and a strong baseline. *arXiv preprint arXiv:2110.07038*.

Xiangyang Liu, Tianxiang Sun, Junliang He, Lingling Wu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. 2021b. 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*.

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

Roberto G Pacheco, Rodrigo S Couto, and Osvaldo Simeone. 2021. Calibration-aided edge inference offloading via adaptive model partitioning of deep neural networks. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6. IEEE.

ME Peters, M Neumann, M Iyyer, M Gardner, C Clark, K Lee, and L Zettlemoyer. 1802. Deep contextualized word representations. arxiv. *arXiv*.

Pushpankar Kumar Pushp and Muktabh Mayank Srivastava. 2017. Train once, test anywhere: Zero-shot learning for text classification. *arXiv preprint arXiv:1712.05972*.

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.

Tal Schuster, Adam Fisch, Tommi Jaakkola, and Regina Barzilay. 2021. Consistent accelerated inference via confident adaptive transformers. *arXiv preprint arXiv:2104.08803*.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A Smith. 2020. The right tool for the job: Matching model and instance complexities. *arXiv preprint arXiv:2004.07453*.

Tianxiang Sun, Xiangyang Liu, Wei Zhu, Zhichao Geng, Lingling Wu, Yilong He, Yuan Ni, Guotong Xie, Xuanjing Huang, and Xipeng Qiu. 2022. A simple hash-based early exiting approach for language understanding and generation. *arXiv preprint arXiv:2203.01670*.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Arun Verma, Manjesh Hanawal, Csaba Szepesvari, and Venkatesh Saligrama. 2019. Online algorithm for unsupervised sensor selection. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, pages 3168–3176. PMLR.

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

Meiqi Wang, Jianqiao Mo, Jun Lin, Zhongfeng Wang, and Li Du. 2019b. Dynexit: A dynamic early-exit strategy for deep residual networks. In *2019 IEEE International Workshop on Signal Processing Systems (SiPS)*, pages 178–183. IEEE.

Yue Wang, Lijun Wu, Juntao Li, Xiaobo Liang, and Min Zhang. 2023. Are the bert family zero-shot learners? a study on their potential and limitations. *Artificial Intelligence*, page 103953.

Zizhao Wang, Wei Bao, Dong Yuan, Liming Ge, Nguyen H Tran, and Albert Y Zomaya. 2019c. See: Scheduling early exit for mobile dnn inference during service outage. In *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 279–288.

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

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020b. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251. Association for Computational Linguistics.

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.

Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. 2020. Resolution adaptive networks for efficient inference. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2369–2378.

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.

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.

A Appendix

A.1 Upper bound on regret of CeeBERT

Theorem A.1. *For any $\gamma > 1$, the regret of CeeBERT with K arms in the action set after n rounds is given as:*

$$R(CeeBERT, n) \leq 4\gamma \sum_{\alpha \neq \alpha^*} \frac{log(n)}{\Delta_\alpha} + (\pi^2/3 + 1) \sum_{\alpha \neq \alpha^*} \Delta_\alpha \quad (5)$$

where $\Delta_\alpha = r(\alpha^*) - r(\alpha)$

Proof: The proof is very similar to the classical UCB1 (Auer et al., 2002a) and follows the same lines with noting the regret in round t as

$$R_t = r(\alpha_t) - r(\alpha^*)$$

$r(\alpha)$ is a bounded quantity by definition and more specifically $r(\alpha) \in [-1 - \lambda L, 1]$, where λL is the latency cost of the final exit, L is the number of layers and λ is the processing cost.

A.2 Regret performance

In the case of online learning algorithms, the regret observed is used to monitor the learning process of the algorithm. We observe that across all the datasets, CeeBERT only requires a few thousand samples to converge and from the figure 3, we can also observe that the observed regret is sublinear as proved on theorem A.1. Since these plots are the average of 5 random runs of the algorithm, we also plot the standard deviation observed, however, the standard deviation plot is barely visible as it is very small and the range of the y-axis is large. We also plot cumulative regret observed when we fix the threshold values as used by previous baselines.

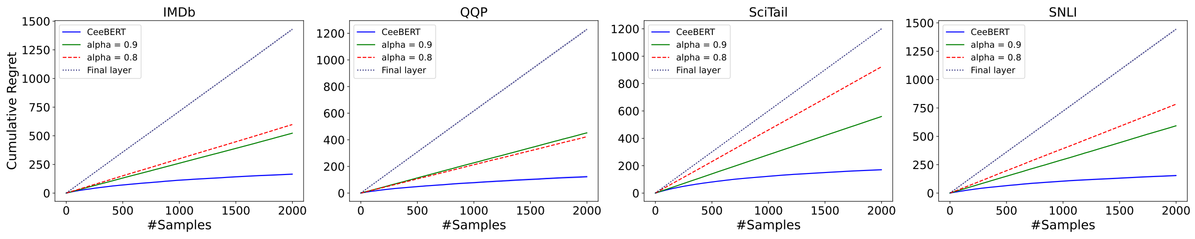


Figure 3: Cumulative regret curves for different datasets.