Finding the SWEET Spot: Analysis and Improvement of Adaptive Inference in Low Resource Settings

Daniel Rotem[♡] **Michael Hassid**[♡] **Jonathan Mamou**[♠] **Roy Schwartz**[♡]
[♡]School of Computer Science & Engineering, Hebrew University of Jerusalem
[♠] Intel Labs, Israel
{daniel.rotem,michael.hassid,roy.schwartz1}@mail.huji.ac.il
jonathan.mamou@intel.com

Abstract

Adaptive inference is a simple method for reducing inference costs. The method works by maintaining multiple classifiers of different capacities, and allocating resources to each test instance according to its difficulty. In this work, we compare the two main approaches for adaptive inference, Early-Exit and Multi-Model, when training data is limited. First, we observe that for models with the same architecture and size, individual Multi-Model classifiers outperform their Early-Exit counterparts by an average of 2.3%. We show that this gap is caused by Early-Exit classifiers sharing model parameters during training, resulting in conflicting gradient updates of model weights. We find that despite this gap, Early-Exit still provides a better speed-accuracy trade-off due to the overhead of the Multi-Model approach. To address these issues, we propose SWEET,[1] an Early-Exit fine-tuning method that assigns each classifier its own set of unique model weights, not updated by other classifiers. We compare SWEET's speed-accuracy curve to standard Early-Exit and Multi-Model baselines and find that it outperforms both methods at fast speeds while maintaining comparable scores to Early-Exit at slow speeds. Moreover, SWEET individual classifiers outperform Early-Exit ones by 1.1% on average. SWEET enjoys the benefits of both methods, paving the way for further reduction of inference costs in NLP. We publicly release our code.[2]

1 Introduction

Pre-trained Transformer-based language models such as BERT (Devlin et al., 2019), DeBERTa (He et al., 2020), and GPT3 (Brown et al., 2020) have become the go-to tool in NLP . Although powerful, the growing size of these models has been a major drawback (Thompson et al., 2020; Schwartz et al., 2020a), making them costly to run. Various attempts to reduce inference cost have been proposed, including distillation (Hinton et al., 2015), pruning (LeCun et al., 1989) and quantization (Courbariaux et al., 2014). This work focuses on adaptive inference (Graves, 2016; Liu et al., 2020), a recent approach in which the variability of sample difficulty is leveraged toward a smarter allocation of computational resources. An appealing property of adaptive inference is that it enables dynamic control of the speed-accuracy trade-off.

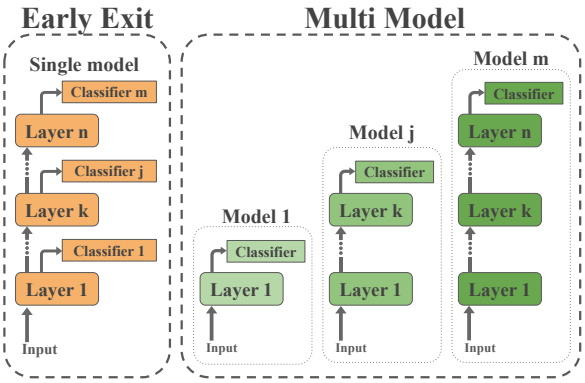

Figure 1: Illustration of the adaptive inference approaches compared in this work. In both methods, multiple classifiers of increasing sizes are run serially, until a confident prediction is made. In Early-Exit (left), a single model with multiple classifiers is used, such that early computations are reused by later classifiers. In Multi-Model (right), a sequence of independent models is used, allowing each classifier to decouple its parameters from other classifiers.

There are two main approaches to adaptive inference, both using a set of classifiers of different sizes. In *Early-Exit* (Schwartz et al., 2020b; Xin et al., 2020), multiple classification heads are added to the same model at different layers, allowing for early exit during inference (Fig. 1, left). Another approach (henceforth *Multi-Model*) is to apply multiple *independent* classifiers of varying capacities serially until a prediction is made (Varshney and

[1] **S**eparating **W**eights in **E**arly **E**xit **T**ransformers.
[2] https://github.com/schwartz-lab-NLP/SWEET

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 14836–14851
July 9-14, 2023

Baral, 2022a; Li et al., 2020, Fig. 1, right). These approaches have complementary benefits; Multi-Model allows for easier batching at inference time, and potentially larger savings due to using very efficient models (Mamou et al., 2023). Early-Exit on the other hand is more memory efficient, faster to train, and enables re-use of early computation if an early exit was not taken.

In this work, we compare the speed-accuracy behavior of the two approaches when training data is limited.[3] We first observe that Early-Exit model weights are updated by multiple conflicting gradient signals throughout the training process (Fig. 2, left). We show that this leads to a decrease in performance of *individual* Early-Exit classifiers compared to Multi-Model ones (2.3% gap on average). We find that this gap is higher for the earliest classifiers (5.2% on average) than for later ones (1.4%).

We also find that while each Multi-Model classifier outperforms its Early-Exit counterpart, it does not translate to an overall better speed-accuracy trade-off. Instead, we find that each method dominates performance in a different region: Multi-Model outperforms Early-Exit at fast inference speeds, while Early-Exit is better at slow speeds. Multi-Model downgraded scores at slow speeds are likely caused by the overhead of running models sequentially to predict hard samples.

Inspired by our findings, we present SWEET,[4] an Early-Exit method for bridging the performance gap between standard Early-Exit and Multi-Model. In SWEET, each Early-Exit classifier only updates the parameters of layers preceding it up to the previous classifier. This way, each model parameter is updated by a single classifier, thus avoiding conflicting gradients during the training of Early-Exit models (Fig. 2, right).

We experiment with two established pre-trained models: BERT (Devlin et al., 2019) and DeBERTa (He et al., 2020). We fine-tune Early-Exit models using SWEET on seven text classification tasks from GLUE (Wang et al., 2018) and compare them to Early-Exit and Multi-Model baselines. The speed-accuracy curve of SWEET dominates both baselines at fast speeds for 21 out of 28 experiments conducted. As for individual classifiers, SWEET performs 1.1% better on average than Early-Exit, mostly improving earlier classifiers, where conflicting gradients are more dominant.

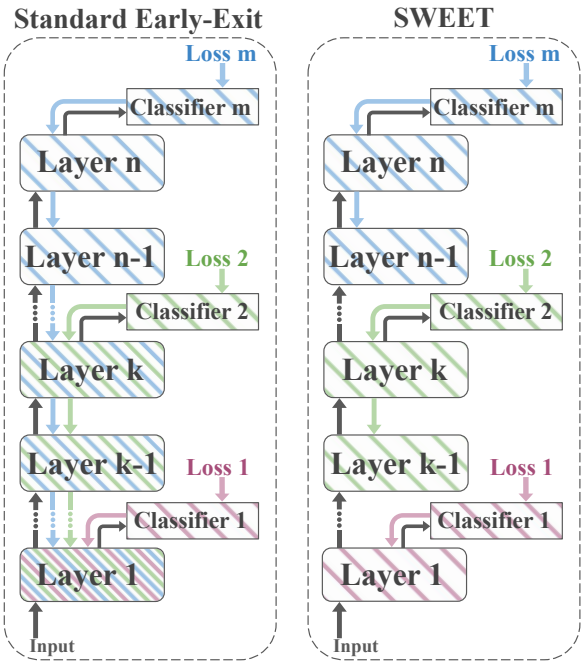

Figure 2: **Left**: standard Early-Exit fine-tuning, where lower layers get gradient updates from multiple classifiers. **Right**: our SWEET method, in which each layer parameters are updated only by the next classifier.

We summarise our main contributions: **(1)** We propose a way of measuring conflicting gradients, and show that they exist in Early-Exit training process; **(2)** We empirically compare Early-Exit and Multi-Model classifiers and show that conflicting gradients lead to individual Early-Exit classifiers being less accurate; **(3)** We propose a novel fine-tuning method, SWEET, which alleviates the conflicting gradients problem in Early-Exit, and leads to improved results at fast inference speeds; **(4)** We publicly release our code.[5]

2 Background: Adaptive Inference

Adaptive inference aims to reduce inference costs of deep neural nets by matching model and sample complexities. Sample difficulty usually varies in real-world data, meaning not all instances require processing by the most powerful classifier. Therefore, we can allocate fewer resources to easier instances, and reduce the average inference cost, potentially at the cost of performance degradation.

[3] Reducing inference costs is particularly helpful when computational resources are limited. Such conditions are often paired with restricted access to labeled data or a low budget for data annotation. To evaluate the effectiveness of adaptive inference methods in such scenarios, we limit training data size to a few thousand examples in all our experiments.

[4] **S**eparating **W**eights for **E**arly-**E**xit **T**ransformers.

[5] https://github.com/schwartz-lab-NLP/SWEET

Several exit strategies have been developed to control the speed-accuracy trade-off of a model by deciding when to make an early prediction and halt computation (Xin et al., 2020; Zhou et al., 2020; Xin et al., 2021; Schuster et al., 2021; Zhang et al., 2022). In this work, we mainly experiment with confidence-based exiting (Schwartz et al., 2020b), in which computation halts if the softmax probability assigned to a given label exceeds a predetermined threshold. Dynamic control of model's inference speed is gained through setting different threshold values. There are two main adaptive inference approaches, Early-Exit and Multi-Model, which we describe below.

Early-Exit Early-Exit models are deep neural nets with multiple output points, following intermediate layers. In this work, we focus on Early-Exit implementation as presented in Schwartz et al. (2020b). During fine-tuning,[6] instances are passed through the model, and a loss is calculated based on predictions made by all classifiers. This leads to some model weights being updated by gradient signals from multiple classifiers (Fig. 2, left).

At inference time, an instance is passed through the model and its label is predicted sequentially until a decision to exit early is taken, and computation is halted. An appealing property of Early-Exit is that it allows for efficient re-use of previous computation from lower layers in higher classifiers. However, this means that some model parameters are updated by multiple classifiers, which may lead to sub-optimal performance of individual classifiers due to conflicting gradients. In this work, we study the effect of this property on Early-Exit models.

Multi-Model In Multi-Model adaptive inference, a set of independent models of increasing capacity are fine-tuned separately for the same task. At inference time, the models are used sequentially from smallest to largest until a prediction meets some predetermined criterion or until the largest model has been used. This method is more robust than Early-Exit, being easier to extend and enabling the use of different architectures. Additionally, Multi-Model potentially allows for further computational savings by using models smaller than the smallest Early-Exit model (a single layer of the backbone model, Mamou et al., 2023). However, it may add overhead to the prediction time of hard instances, as those pass through multiple models with early computations being discarded, bringing the total runtime to exceed that of using the largest model.

[6]Some works have pre-trained Early-Exit Models from scratch (Liu et al., 2021b) but due to budgetary constraints, it is common practice to fine-tune pre-trained models with the added classifiers on the downstream task (Liu et al., 2020; Xin et al., 2020; Schwartz et al., 2020b).

3 Early-Exit vs. Multi-Model

3.1 Conflicting Gradients in Early-Exit

Unlike Multi-Model, when fine-tuning Early-Exit models, model weights are updated by multiple gradient signals, originating in different classification layers (Fig. 2, left). We hypothesize that this leads to sub-optimal performance for all classifiers involved, as gradient signals might conflict with one another and derail the classifiers from their goal. To test this hypothesis, we compare the gradient similarity of different Early-Exit classifiers.

We fine-tune a BERT_{BASE} model with four exit points (following layers $[1, 4, 6, 12]$) for 400 training steps on the MNLI dataset (Williams et al., 2018), using a batch size of 16 with a learning rate of 2e-5. We pass a new training batch through the model,[7] and inspect the gradients with respect to the last feed-forward matrix in each layer preceding a classifier.[8] To measure the degree of alignment between gradient updates of different classifiers, we average the cosine similarity between the rows of gradient matrices for every pair of classifiers. High similarity indicates that the classifiers are updating the weights in a similar direction, while low similarity (in absolute values) suggests that the updates are close to orthogonal and potentially detrimental to both classifiers.

This section studies the strengths and weaknesses of both adaptive inference approaches. We start by describing a limitation of Early-Exit approaches: lower model layers are updated by conflicting gradient signals during training. We show that this leads to inferior performance of individual Early-Exit classifiers compared to corresponding Multi-Model classifiers. We then compare the effects of these performance drops on the speed-accuracy curve of Early-Exit models compared Multi-Model ones.

Fig. 3 shows that the gradients of future classifiers are generally orthogonal to those of the current classifier for each examined Transformer block. This indicates that model weights indeed suffer

[7]We repeat this experiment with an additional batch. Results (Appendix B) show a very similar trend.

[8]Except for the last layer, updated by a single classifier.

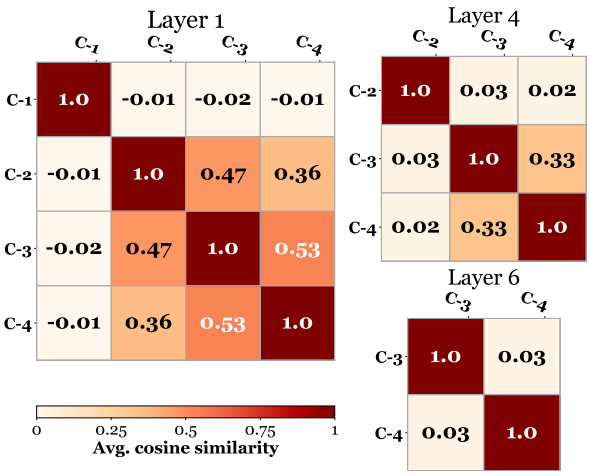

Figure 3: Average cosine similarity between classifiers' gradient updates of model layers. C-i stands for Classifier i. Layer 1 (preceding C-1) is updated by 3 classifiers, while layers 4 (preceding C-2), and 6 (C-3) are updated by 3 & 2 classifiers respectively. For each layer, the gradient update of the following classifier is roughly orthogonal to those of future classifiers, whereas gradient updates of higher classifiers tend to better align with one another.

Size	Method	Exit Layer			
		1	**4**	**6**	**12**
BASE	MM	$\mathbf{60.9}_{0.6}$	$71.4_{0.1}$	$\mathbf{74.7}_{1.2}$	$\mathbf{79.9}_{0.9}$
	EE	$57.3_{0.3}$	$70.3_{0.3}$	$74.4_{0.7}$	$78.7_{0.5}$
	SWEET	$\mathbf{60.9}_{0.5}$	$\mathbf{71.6}_{0.5}$	$74.0_{0.4}$	$77.4_{0.6}$
		1	**6**	**12**	**24**
LARGE	MM	$\mathbf{60.1}_{0.1}$	$\mathbf{66.9}_{0.4}$	$74.4_{0.9}$	$\mathbf{81.6}_{0.3}$
	EE	$56.6_{0.3}$	$65.6_{0.9}$	$74.0_{0.6}$	$79.9_{1.9}$
	SWEET	$59.8_{0.3}$	$66.5_{0.7}$	$\mathbf{74.5}_{0.3}$	$81.3_{1.0}$

Table 1: Results of individual classification layers averaged across all tasks using BERT as a backbone model. Multi-Model (**MM**) classifiers outperform their Early-Exit (**EE**) counterparts, with the gap being the largest for early classifiers. SWEET closes much of this gap, especially for early classifiers. Standard deviation (across random seeds) is reported in subscript. Results for DeBERTa are presented in Table 3.

from conflicting gradient updates during the training process, which might affect the performance of each individual classifier. Interestingly, when multiple future classifiers are present, they tend to align with each other, as indicated by the relatively high similarity between layer 1's gradient updates originating in classifiers 2, 3, and 4, and between layer 4's gradient updates from classifiers 3 and 4.

3.2 Effects of Conflicting Gradients

To evaluate the effect of conflicting gradients on individual Early-Exit classifiers, we compare the different classifiers of an Early-Exit model and a Multi-Model model. For a clean comparison, we use the same backbone model, exit points, hyper-parameters,[9] and random seeds. As an example, for a given 12-layer Transformer model, an Early-Exit model with exit points following layers $[4, 12]$ would be compared to a Multi-Model one consisting of two classifiers: the first four layers fit with a classification head, and a full (12 layer) model. The models differ only in the fact that for the Early-Exit model, model weights are updated by multiple future classifiers during the fine-tuning process, while each Multi-Model classifier is an independent model. To isolate the effect of conflicting gradients on the classifiers, we evaluate each one separately on the entire validation set. The performance gap between each individual Multi-Model classifier and its corresponding Early-Exit classifier, allows us to directly measure the latter's downgrade in performance caused by conflicting gradients.

We experiment with BERT and DeBERTa {BASE (∼110M parameters), and LARGE (∼350M parameters)}. For BASE versions, we install classifiers following layers $[1, 4, 6, 12]$. For LARGE versions, we use layers $[1, 6, 12, 24]$. We fine-tune an Early-Exit model and a corresponding Multi-Model model on seven NLP tasks from the GLUE benchmark (Wang et al., 2019): SST-2 (Socher et al., 2013), MRPC (Dolan and Brockett, 2005), RTE (Dagan et al., 2006; Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009), CoLA (Warstadt et al., 2019), MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), and QQP (Iyer et al., 2016). We report accuracy for all tasks except for CoLA (Matthews corr.). We fine-tune the models for two epochs on all tasks. As our goal is to test Adaptive inference in a low resource setting, we limit the training set size for each task to $6K$.[10] We report the mean validation scores and standard deviation across three random seeds. Other hyper-parameters are listed in Appendix A.

[9] Except for the learning rate, which was tuned for each task and model individually.

[10] For datasets smaller than $6K$, we use the entire dataset. The training sets are randomly sampled for each random seed, and are shared across methods with the same seed.

Table 1 shows results for BERT classifiers,[11] averaged across all tasks. Our results show that multiple classifiers updating the same weights during fine-tuning diminishes the performance of Early Exit classifiers by 2.3% on average (1.7% for BERT, 3.0% for DeBERTa), with earliest classifiers affected the most (3.5% for BERT, 7.0% for DeBERTa). The increased effect on early classifiers supports our hypothesis regarding conflicting gradients: parameters used by early classifiers receive updates from the largest number of classifiers, thus increasing the chance for conflicting gradients and leading to a larger decrease in performance.

3.3 Speed-Accuracy Trad-eoff

So far we observed that Multi-Model classifiers outperform Early-Exit ones. On the other hand, we also note that there is considerable overhead when using a Multi-Model model. For Multi-Model, an instance that makes an exit on classifier C_i, runs all classifiers up to (including) C_i, each being an independent model, while Early-Exit early layer computations are reused by later classifiers. We turn to evaluate the trade-off between these two factors and compare the overall speed-accuracy curve of each adaptive inference method.

We evaluate model scores across different inference speeds using 11 threshold values, evenly spaced in the range $\left(\frac{1}{\#\text{ of labels}}, 1\right)$. We note that $t = \frac{1}{\#\text{ of labels}}$ corresponds to the earliest classifier predicting all labels, while for $t = 1$, the final classifier is used on the entire validation set.

We compute the speedup ratio for each instance as the number of layers it passes through divided by the number of layers of the full backbone model (12 for BASE models, 24 for LARGE models), and average across all instances. As an example, for classifiers following layers [1, 4, 6, 12], an instance that exits on the third classifier (i.e., after layer 6), will have a speedup ratio of $\frac{6}{12}$ for Early-Exit and $\frac{11}{12}$ for Multi-Model ($\frac{6}{12} + \frac{4}{12} + \frac{1}{12}$).[12]

We evaluate models on the seven tasks listed in Section 3.2, and report the average scores across all tasks. BERT$_{BASE}$ results are presented in Fig. 4, while the results for all other models can be found in Fig. 5. The speed-accuracy curves reveal two important observations. First, as expected, Multi-Model achieves better scores at fast speeds (when most instances are predicted by early classifiers).

[11] See Appendix C, Table 3 for DeBERTa results.

[12] Further experimental details can be found in Appendix A.

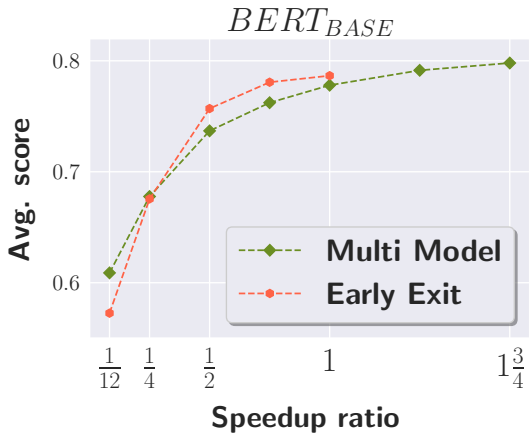

Figure 4: Speed-accuracy trade-off comparison of Multi-Model and Early-Exit. Multi-Model performs better only at fast inference times (up to $1/4$ of original run time), while Early-Exit dominates the remainder of the range. The graph shows the average task scores (y-axis) as a function of the speedup ratio (x-axis).

Second, although each Multi-Model classifier outperforms its corresponding Early-Exit one, the Multi-Model overhead leads to this approach being outperformed by Early-Exit at slow speeds (when more instances are predicted by later classifiers).

4 SWEET: Separating Weights for Early-Exit Transformers

Based on our findings, we aim to design an Early-Exit method that takes advantage of the benefits of both Multi-Model and Early-Exit approaches: making the lower Early-Exit classifiers as accurate as the Multi-Model ones, without the additional overhead of the Multi-Model approach.

4.1 SWEET

We present SWEET—a method for fine-tuning Early-Exit models that avoids the harmful impact of conflicting gradients. SWEET grants each classifier exclusive control over part of the model's parameters, such that each model parameter is only updated by a single classifier. This is done by truncating the loss signal from classifier C_i when reaching the Transformer layer corresponding to classifier C_{i-1} (Fig. 2, right).

In SWEET, each classification head receives complete control over a portion of the Model's parameters and can alter them in a way that optimizes its own goals. Truncating future loss signals makes the very first classifier an independent model, as

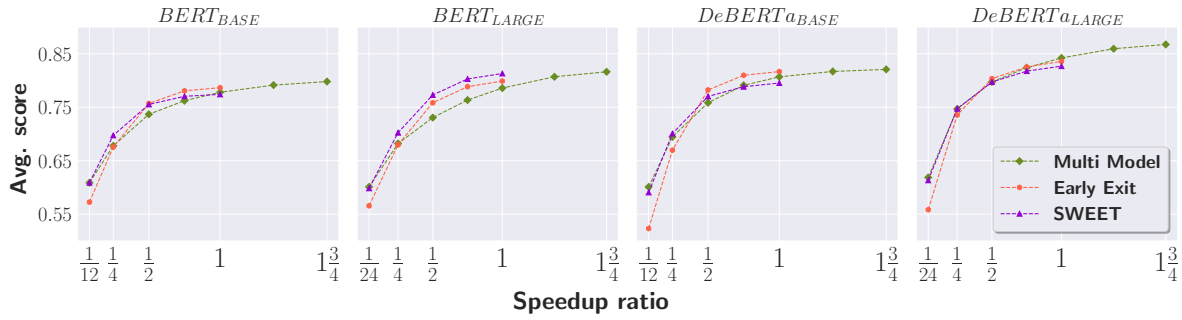

Figure 5: Speed-accuracy tradeoff averaged across tasks. SWEET matches the performance of Multi-Model at fast speeds, while maintaining results comparable to Early-Exit at slow speeds.

opposed to a model that shares its parameters with future classifiers. The following classifiers update only a subset of the model's parameters but are affected by early model parameters, allowing them to still make use of earlier computations when processing an instance. We clarify that all model weights are updated simultaneously, as opposed to training the classifiers (and their corresponding layers) sequentially, which would have led to significantly longer fine-tuning, matching that of a Multi-Model model.

4.2 A Better Speeds-Accuracy Curve

We turn to evaluate the speed-accuracy curve of SWEET. We use the same experimental setup as in Section 3.2; we fine-tune two pre-trained LMs (BERT and DeBERTa) in two sizes (BASE, LARGE) over the same seven text classification tasks. We use the same exit points and evaluate over the entire validation set using confidence-based exiting. We compare SWEET to two baselines: a standard Early-Exit model and a Multi-Model model. We compute the speedup ratio using the same method as described in Section 3.3. For further implementation details, see Appendix A.

Fig. 5 presents the speed-accuracy curves of all models, averaged across all tasks. The figure shows that at the fastest speeds, SWEET outperforms Early-Exit for all models and is comparable to, or outperforms, Multi-Model. However, for 3 out of the 4 models (all but BERT_{LARGE}), Early-Exit surpasses the performance of SWEET at slow speeds. SWEET's reduced scores at slow inference speeds are likely due to the lower capacity of the later classifiers, stemming from their restricted influence on early model parameters during fine-tuning.

Fig. 6 presents results on individual tasks for BERT_{BASE}.[13] On five out of seven tasks, SWEET outperforms both baselines at fast speeds (up to a speedup ratio of $1/2$), suggesting that SWEET improves the performance of lower Early-Exit classifiers by avoiding conflicting gradients during training. For two tasks (MRPC and CoLA), SWEET suffers a considerable decrease in performance at slow inference speeds. For the other five tasks SWEET maintains comparable results to Early-Exit.

It is also interesting to examine how SWEET affects *individual* Early-Exit classifiers. Table 1 shows the results of BERT's individual classifiers trained with SWEET compared to Early-Exit and Multi-Model.[14] SWEET classifiers close much of the gap between Early-Exit and Multi-Model: they outperform those of Early-Exit by 1.1% on average (1.2% for BERT, 1.0% for DeBERTa), with the margin being larger for the earliest classifier (3.4% for BERT, 6.2% for DeBERTa). The final two classifiers trained with SWEET achieve lower scores than those of Early-Exit (0.9% on average), probably due to the restricted influence those classifiers have on early model parameters. Our results hint that SWEET is able to effectively bridge the gap between Early-Exit and Multi-Model early classifiers, leading to a speed-accuracy curve favoring fast inference speeds.

We note that during our experiments with Early-Exit and Multi-Model models, some models did not converge. In order to ensure the validity of our results, we repeated these experiments with different random seeds, which led to convergence. We emphasize that this did not occur during the training of models using SWEET.

[13] Fig. 10 in Appendix D shows individual results for the other models.

[14] See Table 3 in Appendix C for DeBERTa results.

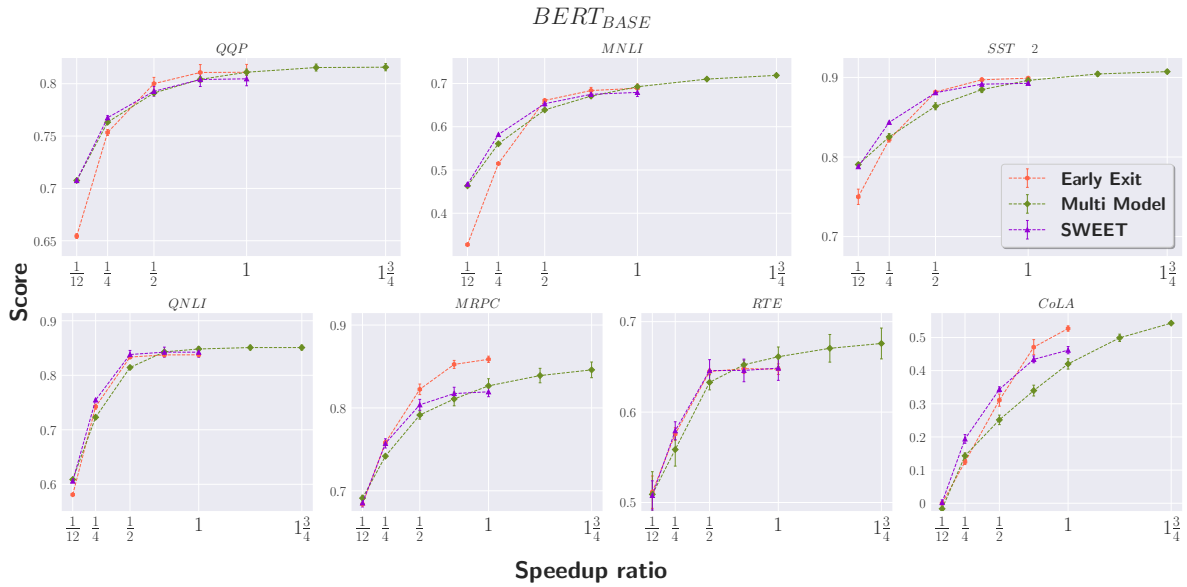

Figure 6: Speed-accuracy curve for individual tasks, using a BERT$_{BASE}$ model. SWEET outperforms both baselines at fast speeds (lower values of x) on five out of seven tasks. See Fig. 10 for BERT$_{LARGE}$ and DeBERTa results.

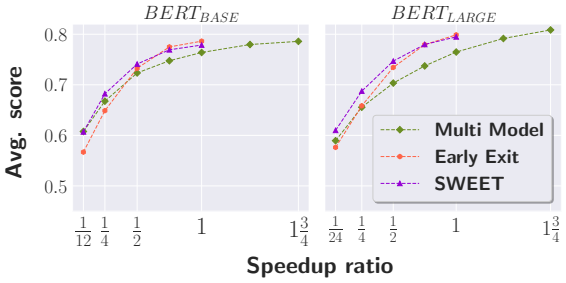

Figure 7: Speed-Accuracy curves with the learning-to-exit strategy. As with confidence-based methods, SWEET outperforms both baselines at fast speeds, while maintaining comparable results at slow speeds.

5 Further Analysis

We turn to evaluate the robustness of our approach. We start with evaluating SWEET using a different exit strategy. We then test how our results generalize when using varying amounts of training data.

A different exit strategy: Learning to Exit Our experiments so far have used a confidence-based exit strategy (Section 2). Here we consider a different strategy—learning to exit (Xin et al., 2021), in which a separate module is trained for each classifier, learning whether the classifier's prediction should be trusted, and an exit should be made. This method also allows for a natural extension to regression problems, unlike confidence-based methods that are limited to classification tasks.

We use BERT {BASE, LARGE} as a backbone model and fine-tune in the same procedure described in Section 3.3. Our results (Fig. 7) reveal that the methods behave similarly to the confidence-based setup; SWEET outperforms both baselines at the early stage of the curve (fast speeds), while performing on par with Early-Exit at slow speeds. We also measure the performance of individual exit layers, as in Section 3.2, for models fine-tuned using learning-to-exit. Our results (Table 4 in Appendix E) reveal a similar behavior to that of models fine-tuned using confidence based early exiting: Multi-Model classifiers surpass the performance of Early-Exit classifiers. Moreover, SWEET's early classifiers outperform those of Early-Exit, while later ones show a slight decrease in performance.

Varying data sizes Our experiments so far have focused on low-resource setups (a training set of several thousand examples). We now evaluate how SWEET performs as we vary the amount of training data. We fine-tune a BERT$_{BASE}$ model on different portions of the MNLI dataset using the same experimental setup as in Section 3.2. Our results, shown in Fig. 8, indicate that SWEET is mostly beneficial when training data is limited to a few thousand examples, but still somewhat useful even with as many as 60K training instances. Nonetheless, the effects of conflicting gradients on the earliest Early-Exit classifier tend to disappear when training on the full dataset (400K instances), making it as good as the smallest Multi-Model classifier. Moreover, in that setup, the use of SWEET seems to substan-

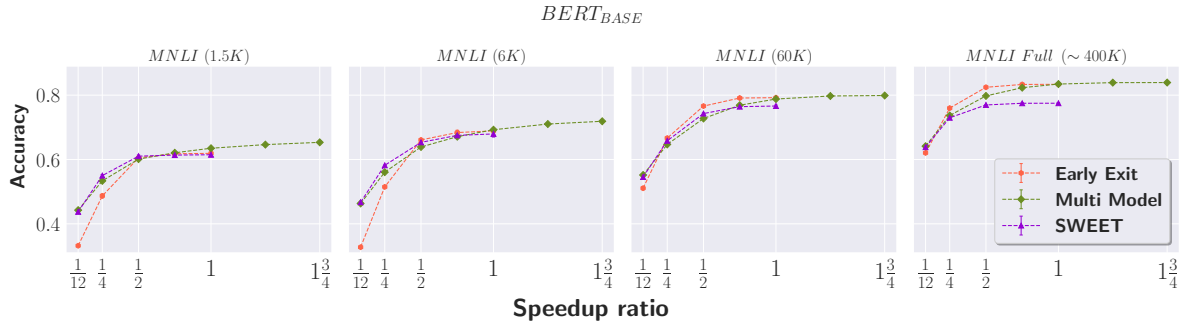

Figure 8: Speed-accuracy curves of $BERT_{BASE}$ models trained on varying sizes of the MNLI dataset. The title of each plot describes the amount of training data used in the fine-tuning phase. The benefits of using SWEET vanish with large amounts of training data.

tially decrease the performance of later classifiers, suggesting that the harm caused by limiting the influence of classifiers on model parameters may grow with the amount of training data.

6 Related Work

Several approaches have been proposed for reducing the inference time of large models (Treviso et al., 2022). These include knowledge distillation (Ba and Caruana, 2014; Hinton et al., 2015), in which the knowledge of a large teacher model is transferred to a smaller student model; pruning (LeCun et al., 1989; Frankle and Carbin, 2018), which removes unnecessary model parameters; and weight quantization (Jacob et al., 2018; Zafrir et al., 2019), which reduces the floating point precision of model weights. See (Treviso et al., 2023, Section 6), for a recent survey. Adaptive inference methods studied in this work are an orthogonal approach, which can be used in combination with these methods (Schwartz et al., 2020b).

The use of Adaptive inference in deep neural networks has been extensively studied, with successful implementations in various types of networks including recurrent neural networks (Graves, 2016) and convolutional neural networks (Teerapittayanon et al., 2016; Huang and Chen, 2018). In the context of this work, it has also been applied to existing backbone pre-trained language models: Xin et al. (2020) implemented early exit strategies on top of classification tasks. Zhou et al. (2020) Introduced a patience-based exiting, requiring sequential classifiers to agree on a label for a prediction to be made. Xin et al. (2021) extended the use of Early-Exit in Transformers to regression tasks, as well as addressed the issue of reduced performance of the final classifier through the use of an alternating training algorithm. Schuster et al. (2021) proposed a confidence-based early exit model with guarantees on the agreement between early exit predictions and the final classifier.

Liu et al. (2022) presented a strong baseline for efficient NLP by adding multiple classifiers to a BERT model during pre-training. Recently, Schuster et al. (2022) adjusted the Early-Exit method to language modeling for text generation, making dynamic computation at the single token level.

Multi-Model approaches to adaptive inference have been proposed for vision tasks (Enomoro and Eda, 2021) as well as for NLP (Li et al., 2020; Varshney and Baral, 2022b). Mamou et al. (2023) introduced a two-tier design with a highly efficient model serving as the first model and a powerful model serving as the second, enabling the possibility of achieving extremely fast inference speed.

Finally, the concept of conflicting gradients has been mainly studied in the context of multi-task learning, where a single model is faced with solving different tasks (Yu et al., 2020; Liu et al., 2021a). To the best of our knowledge, no previous work examined this in the context of Early-Exit.

7 Conclusion

In this work, we analyzed the performance of two common adaptive inference methods–Early-Exit and Multi-Model. We found evidence that model weights are updated by conflicting gradients in the training process of Early-Exit models, causing classifiers to perform at a sub-optimal level. Despite this, we showed that regarding the entire speed-accuracy curve, Early-Exit is still favorable to Multi-Model due to the overhead of using independent model runs in a Multi-Model setup.

To address these findings, we proposed SWEET,

a novel Early-Exit method, which avoids conflicting gradients by allocating each Early-Exit classifier a subset of model weights which are updated solely by it. We found that for Early-Exit models trained with SWEET, early classifiers perform better than those of standard Early-Exit, but later classifiers of SWEET are not as good. These measures lead to SWEET outperforming both Early-Exit and Multi-Model in fast speeds, with slightly worse results than Early-Exit at slow speeds.

Overall, our results demonstrate that Early-Exit models can benefit from fine-tuning algorithms that are tailored to their architecture, and that SWEET is a promising approach for improving the speed-accuracy trade-off of Early-Exit models in the context of adaptive inference.

8 Limitations

This work focuses on the effects of adaptive inference in a low-resource setting, specifically when training data is limited. Our experiments (Section 5) suggest that the negative impact of conflicting gradients may be less prominent when larger amount of training data is available.

Our experiments were conducted using relatively small pre-trained language models ($\leq 350M$ parameters) due to computational constraints, and we defer the replication of our findings with larger, more powerful models to future work. Nonetheless, our results have important implications for the growing trend of increasingly large language models. We hope this work inspires further research on methods to reduce the computational cost of NLP.

This work concentrates on evaluating the speed-accuracy trade-off of Multi-Model and Early-Exit at inference time. We recognize that there are additional factors, such as memory usage, batch processing, and training duration, that could be considered when comparing these methods.

Finally, we experimented with seven text classification tasks in English. We recognize that results may vary for other tasks and languages.

Acknowledgements

We acknowledge Yarden Tal for her early contributions to this work, and Gabriel Stanovsky for his advice and meaningful feedback. This work was supported in part by the Israel Science Foundation (grant no. 2045/21), NSF-BSF grant 2020793, and by a grant from Intel Labs.

References

Jimmy Ba and Rich Caruana. 2014. Do deep nets really need to be deep? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2654–2662. Curran Associates, Inc.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2014. Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer.

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

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proc. of IWP*.

Shohei Enomoro and Takeharu Eda. 2021. Learning to cascade: Confidence calibration for improving the accuracy and computational cost of cascade inference systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9.

Alex Graves. 2016. Adaptive computation time for recurrent neural networks. arXiv:1603.08983.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *ICML*.

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

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. Cite arxiv:1503.02531 Comment: NIPS 2014 Deep Learning Workshop.

Gao Huang and Danlu Chen. 2018. Multi-scale dense networks for resource efficient image classification. *ICLR 2018*.

Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. 2016. First quora dataset release: Question pairs. https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2704–2713.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. *Advances in neural information processing systems*, 2.

Lei Li, Yankai Lin, Deli Chen, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. 2020. Cascadebert: Accelerating inference of pre-trained language models via calibrated complete models cascade. *arXiv preprint arXiv:2012.14682*.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. 2021a. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 34:18878–18890.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. FastBERT: a self-distilling BERT with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6035–6044, Online. Association for Computational Linguistics.

Xiangyang Liu, Tianxiang Sun, Junliang He, Jiawen Wu, Lingling Wu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. 2022. Towards efficient NLP: A standard evaluation and a strong baseline. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3288–3303, Seattle, United States. Association for Computational Linguistics.

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. *arXiv preprint arXiv:2110.07038*.

Jonathan Mamou, Oren Pereg, Moshe Wasserblat, and Roy Schwartz. 2023. TangoBERT: Reducing inference cost by using cascaded architecture. In *In Proc. EMC*^{2}.

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

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. *arXiv preprint arXiv:2207.07061*.

Tal Schuster, Adam Fisch, Tommi Jaakkola, and Regina Barzilay. 2021. Consistent accelerated inference via confident adaptive transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4962–4979.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2020a. Green AI. *Communications of the ACM*, 63(12):54–63.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. 2020b. The right tool for the job: Matching model and instance complexities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6640–6651, Online. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*.

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.

Neil C Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F Manso. 2020. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*.

Marcos Treviso, Tianchu Ji, Ji-Ung Lee, Betty van Aken, Qingqing Cao, Manuel R. Ciosici, Michael Hassid, Kenneth Heafield, Sara Hooker, Pedro H. Martins, André F. T. Martins, Peter Milder, Colin Raffel, Jessica Forde, Edwin Simpson, Noam Slonim, Jesse Dodge, Emma Strubell, Niranjan Balasubramanian, Leon Derczynski, Iryna Gurevych, and Roy Schwartz. 2023. Efficient methods for natural language processing: A survey. *TACL*.

Marcos Treviso, Tianchu Ji, Ji-Ung Lee, Betty van Aken, Qingqing Cao, Manuel R Ciosici, Michael Hassid, Kenneth Heafield, Sara Hooker, Pedro H Martins, et al. 2022. Efficient methods for natural language processing: A survey. *arXiv preprint arXiv:2209.00099*.

Neeraj Varshney and Chitta Baral. 2022a. Model cascading: Towards jointly improving efficiency and accuracy of nlp systems. *arXiv preprint arXiv:2210.05528*.

Neeraj Varshney and Chitta Baral. 2022b. Model cascading: Towards jointly improving efficiency and accuracy of nlp systems. In *In Proceedings of EMNLP*.

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

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 *Proc. of ICLR*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *TACL*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proc. of NAACL*.

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

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

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*, pages 36–39.

Zhen Zhang, Wei Zhu, Jinfan Zhang, Peng Wang, Rize Jin, and Tae-Sun Chung. 2022. Pcee-bert: Accelerating bert inference via patient and confident early exiting. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 327–338.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: fast and robust inference with early exit. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 18330–18341.

A Implementation Details

Further implementation details For fine-tuning BERT models, we use a batch size of 16. For fine-tuning DeBERTa, due to GPU memory constraints, we use a batch size of 16 for BASE and 8 for LARGE. We fine tune the models for two epochs with a maximal sequence length of 256. We use $\beta = 0.9, 0.999$ for the *AdamW* optimizer with linear LR-decay. We optimize the initial learning rate for each method, model size & task by performing a search over five values {1e-5, 2e-5, 3e-5, 4e-5, 5e-5} and choose the value leading to the largest area under the speed-accuracy curve. Chosen LRs are presented in table Table 2

All experiments were run on a single NVIDIA A5000 GPU. The overall computational budget was ~1000 GPU hours. We implement all methods using the HuggingFace Transformer library (Wolf et al., 2020).

Speedup evaluation The speed-up ratio of the model when using exit threshold t is calculated using the formula:

$$\text{speedup}_t = \frac{\sum_{i=1}^{M} S_i^t \cdot L_i}{L_M \cdot \sum_{i=1}^{M} S_i^t} \quad (1)$$

where M is the number of classifiers used for the model, L_i denotes the number of the layer preceding the i-th classification head and S_i^t denotes the number of samples classified by the i-th classifier when using threshold t.

The same exit threshold can lead to different speedup ratios amongst models trained with different random seeds. We use linear interpolation to approximate the accuracy score at set speedup ratios. We evaluate at $(1/12, 1/4, 1/2, 3/4, 1)$[15] and report the average across three random seeds as well as a 95% confidence interval.[16] We use temperature scaling (Guo et al., 2017) to make classifiers confidence, and therefore early-exiting decisions, more reliable. Note that this scaling is monotonic and therefore does not influence predictions.

B Conflicting Gradients

We replicate the experiment done in Section 3.1 with another batch of size 16 to examine if our findings generalize. results presented in Fig. 9 show a similar trend to Fig. 3: Gradient updates of current classifiers are roughly orthogonal to those of future classifiers, whereas future classifier updates are more aligned.

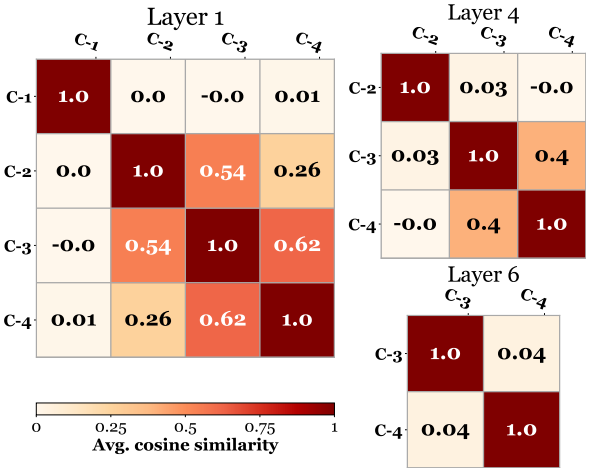

Figure 9: Average cosine similarity between classifiers' gradient updates of model layers. C-i stands for Classifier i. Layer 1 (preceding C-1) is updated by 3 classifiers, while layers 4 (preceding C-2), and 6 (C-3) are updated by 3 & 2 classifiers respectively. For each layer, the gradient update of the following classifier is roughly orthogonal to those of future classifiers, whereas gradient updates of higher classifiers tend to better align with one another.

C DeBERTa Individual Layer Comparison

Table 3 Shows individual classifier results for DeBERTa models. As with the BERT results, Multi-Model classifiers outperform corresponding Early-Exit classifiers. SWEET early classifiers are better than Early-Exit ones, while later classifiers tend to downgrade in performance.

D Individual Task Results

Fig. 10 shows results on individual tasks for BERT_{LARGE} and DeBERTa (BASE & LARGE). For BERT_{LARGE}, SWEET outperforms both baselines throughout the entire speed-accuracy curve over all tasks examined. For DeBERTa models, results are similar to those of BERT_{BASE}, where SWEET performs better at high speeds (small speedup ratio) and is dominated at low speeds (where later classifiers do most of the heavy lifting).

[15] For LARGE models we evaluate at $1/24$ as the fastest speedup ratio.

[16] For Multi-Model we also use $(1\frac{3}{8}, 1\frac{3}{4})$ as overhead causes the model to perform at such "speedup" ratios.

Model	Size	Task						
		SST-2	MRPC	CoLA	MNLI	QQP	QNLI	RTE
BERT	BASE	5\5\5	5\3\4	4\4\5	5\5\5	5\5\5	5\4\5	5\5\4
	LARGE	4\2\2	4\5\4	4\3\4	4\3\4	5\3\4	4\4\3	3\3\3
DeBERTa	BASE	3\3\3	4\5\5	4\5\4	2\4\3	4\5\5	3\4\3	2\1\4
	LARGE	2\4\3	3\2\1	3\3\2	2\2\2	2\2\3	3\2\1	1\2\2

Table 2: Chosen initial learning rate to optimize area under the speed-accuracy curve of each model, size, task. All numbers should be multiplied by 1e-5. $x\backslash y\backslash z$ represent the initial learning rate of Early-Exit \ Multi-Model \ SWEET respectively.

Size	Method	Exit Layer			
		1	4	6	12
BASE	MM	**60.1**$_{0.4}$	**74.6**$_{0.6}$	**78.3**$_{1.1}$	**82.2**$_{0.7}$
	EE	52.3$_{0.4}$	70.9$_{0.6}$	77.9$_{0.8}$	81.7$_{0.1}$
	SWEET	59.1$_{1.5}$	72.4$_{0.2}$	75.1$_{1.1}$	79.5$_{0.5}$
		1	**6**	**12**	**24**
LARGE	MM	**61.9**$_{0.1}$	**76.2**$_{0.4}$	**80.2**$_{0.9}$	**86.8**$_{0.2}$
	EE	55.8$_{0.9}$	74.7$_{1.1}$	79.2$_{0.8}$	83.6$_{1.9}$
	SWEET	61.3$_{0.7}$	76.0$_{0.5}$	77.8$_{0.4}$	82.7$_{0.4}$

Table 3: Results of individual classification layers averaged over all tasks using DeBERTa as a backbone model. Best scores are highlighted in bold, standard deviation (across random seeds) is reported in subscript. The results for BERT models presented in Table 1.

Size	Method	Exit Layer			
		1	4	6	12
BASE	MM	**53.7**$_{0.5}$	**66.1**$_{0.4}$	**71.1**$_{0.3}$	**75.4**$_{2.7}$
	EE	49.5$_{1.1}$	63.0$_{0.2}$	70.0$_{0.6}$	75.3$_{0.4}$
	SWEET	53.5$_{0.4}$	64.9$_{0.9}$	68.5$_{1.4}$	74.1$_{0.4}$
		1	**6**	**12**	**24**
LARGE	MM	51.6$_{0.3}$	59.9$_{0.4}$	67.6$_{2.6}$	**77.9**$_{0.8}$
	EE	50.5$_{0.8}$	59.3$_{0.9}$	68.8$_{1.3}$	76.6$_{1.6}$
	SWEET	**53.9**$_{0.5}$	**60.6**$_{0.2}$	**69.5**$_{0.4}$	75.81$_{1.0}$

Table 4: Results of individual classification layers averaged over all tasks using BERT as a backbone model, fine-tuned using learning-to-exit. Best scores are highlighted in bold, standard deviation (across random seeds) is reported in subscript.

E Learning-to-exit Individual Layer Comparison

Table 4 Shows individual classifier results for BERT models fine tuned with the learning to exit strategy. As with the confidence-based results, Multi-Model classifiers outperform corresponding Early-Exit classifiers. SWEET early classifiers are better than Early-Exit ones, while later classifiers tend to downgrade in performance.

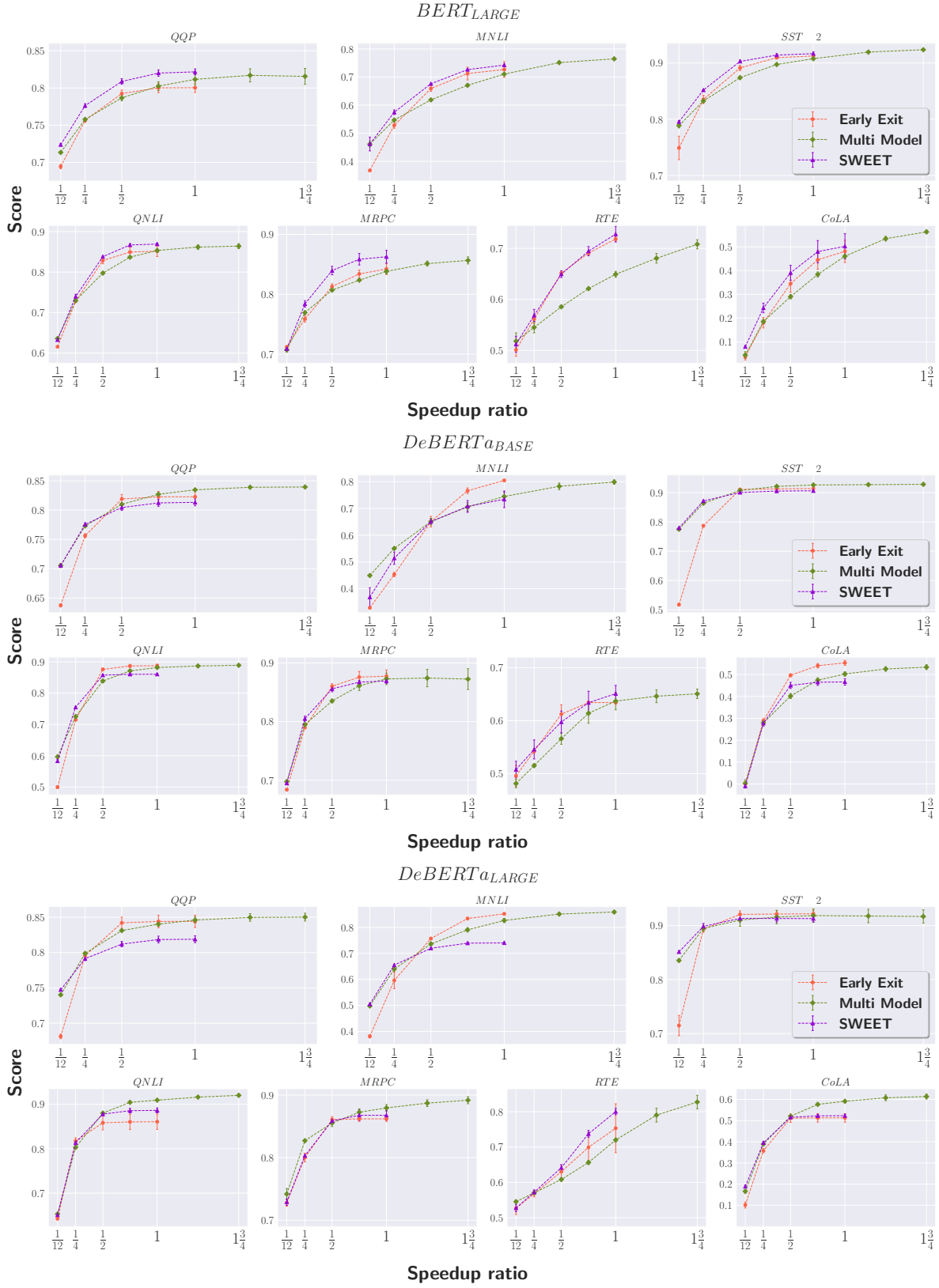

Figure 10: Speed-accuracy curves for individual tasks. For BERT_{LARGE}, SWEET outperforms both baselines across the entire curve. For DeBERTa, SWEET performs better than Early-Exit at high speeds on 9 out of 14 experiments.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 8

☒ A2. Did you discuss any potential risks of your work?
Our work does not present any risks.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1

☑ A4. Have you used AI writing assistants when working on this paper?
We used ChatGPT for assistance, purely with the language of the paper

B ☑ Did you use or create scientific artifacts?

Section 3 + 4

☑ B1. Did you cite the creators of artifacts you used?
Section 3 + 4

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
We use publicly available data sets which are commonly used during research

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Such terms were not specified.

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

☒ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
We use publicly available data sets which are commonly used during research

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 3 + 4, Appendix A

C ☑ Did you run computational experiments?

Section 3 + 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 1 + 3, Appendix A

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

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 3 + 4, Appendix A

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 3 + 4

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

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

Left blank.

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

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

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

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

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