Towards Efficient NLP: A Standard Evaluation and A Strong Baseline

Xiangyang Liu[1]*, Tianxiang Sun[1]*, Junliang He[1], Lingling Wu[1], Xinyu Zhang[2], Hao Jiang[2], Zhao Cao[2], Xuanjing Huang[1], Xipeng Qiu[1]

[1]School of Computer Science, Fudan University

[1]Shanghai Key Laboratory of Intelligent Information Processing, Fudan University

[2]Huawei Poisson Lab

{xiangyangliu20,txsun19,xjhuang,xpqiu}@fudan.edu.cn

{zhangxinyu35,jianghao66,caozhao1}@huawei.com

Abstract

Supersized pre-trained language models have pushed the accuracy of various NLP tasks to a new state-of-the-art (SOTA). Rather than pursuing the reachless SOTA accuracy, most works are pursuing improvement on other dimensions such as efficiency, leading to "Pareto SOTA". Different from accuracy, the metric for efficiency varies across different studies, making them hard to be fairly compared. To that end, this work presents **ELUE** (**E**fficient **L**anguage **U**nderstanding **E**valuation), a standard evaluation, and public leaderboard for efficient NLP models. ELUE is dedicated to depict the Pareto Front for various language understanding tasks, such that it can tell whether and how much a method achieves Pareto improvement. Along with the benchmark, we also pre-train and release a strong baseline, **ElasticBERT**, whose elasticity is both static and dynamic. ElasticBERT is static in that it allows reducing model layers on demand. ElasticBERT is dynamic in that it selectively executes parts of model layers conditioned on the input. We demonstrate the ElasticBERT, despite its simplicity, outperforms or performs on par with SOTA compressed and early exiting models. The ELUE benchmark is publicly available at http://eluebenchmark.fastnlp.top/[1].

1 Introduction

Driven by the large-scale pre-training, today's NLP models have become much more powerful (Devlin et al., 2019; Yang et al., 2019; Lan et al., 2020; Raffel et al., 2020; Sun et al., 2020a; Brown et al., 2020; Qiu et al., 2020). As a consequence of this drastic increase in performance, these pre-trained language models (PLMs) are notorious for becoming more and more computationally expensive due to the increasing number of parameters. Therefore, rather than pre-training a larger model to achieve

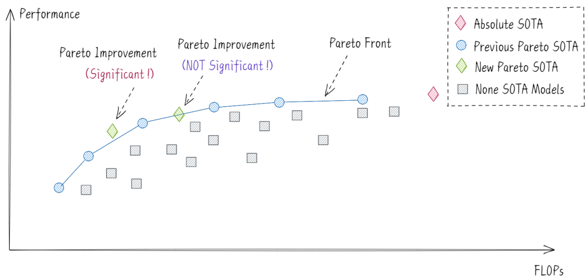


Figure 1: An illustration to show our motivation, that is, building the Pareto front can help recognizing whether and how much a method achieves Pareto improvement.

a new state-of-the-art (SOTA) accuracy, most studies are pursuing improvement on other dimensions such as the number of parameters or FLOPs (Gordon et al., 2020; Sanh et al., 2019; Jiao et al., 2020; Lan et al., 2020; Shen et al., 2020). For these works, the goal has shifted from simple SOTA to "Pareto SOTA". A Pareto SOTA model means that there is no other model is currently better than it on all the dimensions of interest. For example, a model may claim to be Pareto SOTA as long as it achieves the best accuracy under the same number of parameters or FLOPs. For these efficient models with fewer parameters or FLOPs, it is unfair to get them evaluated on the accuracy-centric benchmarks such as GLUE (Wang et al., 2019b), and ranked among many large-scale models.

The shifted goal has outpaced the existing benchmarks, which cannot provide a comprehensive and intuitive comparison for efficient methods. In the absence of a proper benchmark, measures of efficiency in different studies cannot be standardized, and different methods cannot be fairly compared. As a result, it is difficult to say *whether* and *how much* a method achieves Pareto improvement. To that end, we aim to build the Pareto front for various tasks with standard evaluation for both performance and efficiency. Our motivation can be briefly illustrated by Figure 1.

*Equal contribution. Order determined by coin flip.

[1]Work in progress.

Need for a standard evaluation As the goal has shifted, a new benchmark is urgently needed to comprehensively compare the NLP models in multiple dimensions. Currently, this multi-dimensional comparison is done in the individual papers, resulting in the following issues: **(a) Incomprehensive comparison.** The comparison is usually point-to-point, e.g. comparing model performance under the same FLOPs. The comparison in a broader range is usually missed, especially for works in conditional computation where the model performance varies with FLOPs. **(b) Unaccessible results.** Even if the comprehensive line-to-line comparison is conducted, the results are usually presented in form of figure, in which the data points are not accessible for the following work. As a result, the following work has to reproduce or estimate the results (e.g. Xin et al. (2021) estimate values from the figures of Zhou et al. (2020a)). **(c) Non-standard measurements.** Different works may adopt different metrics such as physical elapsed time, FLOPs, and executed model layers, making them hard to directly compare. Even if the adopted metrics are the same, there is no guarantee that they will be calculated in the same way (e.g. the hardware infrastructure, or the software to calculate FLOPs can be very different[2]). **(d) Inconvenience.** Recent studies usually choose GLUE (Wang et al., 2019b) as the main benchmark, which, however, is not suitable for dynamic methods due to its submission limitation that is designed to avoid overfitting on test sets.

Need for a strong baseline Currently, there are roughly two branches of efficient methods in NLP: static methods (e.g. distillation, pruning, quantization, etc.) and dynamic methods (e.g. early exiting). **(a) Static models** are obtained given an expected number of parameters or inference latency. These methods often use the first few layers (to keep the same number of parameters or FLOPs) of some pre-trained model followed by a classification head as their baseline, which, however, is too weak to serve as a baseline. **(b) Dynamic models** usually add multiple internal classifiers to the pre-trained LMs, and therefore allow flexible inference conditioned on the input. Nevertheless, the injected internal classifiers introduce a gap between pre-training and fine-tuning. Training the internal classifiers on downstream tasks often degenerates the performance of the entire model (Xin et al., 2021). Thus, static models need a strong baseline, and dynamic models need a strong backbone.

Contributions In this work, we address the above needs by contributing the following:

- **ELUE** (**E**fficient **L**anguage **U**nderstanding **E**valuation) – a standard benchmark for efficient NLP models. (1) ELUE supports online evaluation for model performance, FLOPs, and number of parameters. (2) ELUE is also an open-source platform that can facilitate future research. We reproduce and evaluate multiple compressed and early exiting methods on ELUE. All of the results are publicly accessible on ELUE. (3) ELUE provides an online leaderboard that uses a specific metric to measure how much a model oversteps the current Pareto front. ELUE leaderboard also maintains several separate tracks for models with different sizes. (4) ELUE covers six NLP datasets spanning sentiment analysis, natural language inference, similarity and paraphrase tasks.

- **ElasticBERT** – a strong baseline (backbone) for static (dynamic) models. ElasticBERT is a multi-exit Transformer (Vaswani et al., 2017) pre-trained on $\sim$160GB corpus. The pre-training objectives, MLM and SOP (Lan et al., 2020), are applied to multiple Transformer layers instead of only the last layer. Gradient equilibrium (Li et al., 2019) is adopted to alleviate the conflict of the losses at different layers. For static models, ElasticBERT is a strong baseline that can reach or even outperform distilled models. For dynamic models, ElasticBERT is a robust backbone that closes the gap between pre-training and fine-tuning.

2 Related Work

NLP Benchmarks Evaluating the quality of language representations on multiple downstream tasks has become a common practice in the community. These evaluations have measured and pushed the progress of NLP in recent years. SentEval (Conneau and Kiela, 2018) introduces a standard evaluation toolkit for universal sentence representations, covering multiple NLP tasks including classification, natural language inference and sentence sim-

[2]We find that the FLOPs of Transformers calculated by different libraries (`thop`, `ptflops`, and `torchstat`) can be different. And besides, all of them missed FLOPs in some operations such as self-attention and layer normalization.

ilarity. Further, GLUE (Wang et al., 2019b) and SuperGLUE (Wang et al., 2019a) provide a set of more difficult datasets for model-agnostic evaluation. Another line of work is multi-dimensional evaluations. EfficientQA (Min et al., 2020) is an open-domain question answering challenge that evaluates both accuracy and system size. The system size is measured as the number of bytes required to store a Docker image that contains the submitted system. Dynabench (Kiela et al., 2021), an open-source benchmark for dynamic dataset creation and model evaluation, also supports multi-dimensional evaluation. In particular, Dynabench measures model performance, throughput, memory use, fairness, and robustness. Both EfficientQA and Dynabench require the user to upload the model along with the required environment to the server, which is costly for users to upload and also for the server to evaluate. In contrast, ELUE adopts a cheaper way to evaluate performance and efficiency of the model. Recently, Long-Range Arena (LRA) (Tay et al., 2021) is proposed to evaluate sequence models under the long-context scenario. Different from ELUE, LRA mainly focuses on Xformers (Lin et al., 2021). Besides, some long-context tasks included in LRA are not NLP tasks, or even not real-world tasks, while ELUE consists of common language understanding tasks. In addition, ELUE is also inspired by other well-known benchmarks, such as SQuAD (Rajpurkar et al., 2016), SNLI (Bowman et al., 2015), MultiNLI (Williams et al., 2018), DecaNLP (McCann et al., 2018), CLUE (Xu et al., 2020b), HotpotQA (Yang et al., 2018), etc.

Efficient NLP Models Existing efficient NLP models can be categorized as two branches: model compression (static methods) and conditional computation (dynamic methods). Compressing a cumbersome model to reduce the number or precision of parameters is a straightforward and effective solution. Currently, there are several approaches to achieve model compression: (1) *model pruning*, which removes parts of neural network that are less important (Gordon et al., 2020), (2) *knowledge distillation*, which learns a compact student model that learns from the prediction distributions from the cumbersome teacher model (Sanh et al., 2019; Jiao et al., 2020), (3) *weight sharing* across different parts (e.g., layers) of the model (Lan et al., 2020), (4) *quantization*, which uses low bit precision for parameter storage and speed-up inference with low bit hardware operations (Shen et al., 2020), and (5) *module replacing*, which replaces the modules of large-scale models with more compact substitutes (Xu et al., 2020a). Instead of pursuing a more compact static model, conditional computation is to selectively activate only parts of the model conditioned on a given input (Bengio et al., 2013; Davis and Arel, 2014). Graves (2016) developed an end-to-end halting mechanism, Adaptive Computation Time (ACT), to perform input-adaptive computation, which is later used in Universal Transformer (Dehghani et al., 2019). Recently, as the emergence of large-scale models for natural language processing, early exiting is also used to speed up inference of transformer-based models, such as Depth-Adaptive Transformer (Elbayad et al., 2020), DeeBERT (Xin et al., 2020), FastBERT (Liu et al., 2020), RightTool (Schwartz et al., 2020), PABEE (Zhou et al., 2020b), LeeBERT (Zhu, 2021), CascadeBERT (Li et al., 2021a), etc.

3 ELUE: A Standard Benchmark for Efficient NLP Models

ELUE is aimed to offer a standard evaluation for various efficient NLP models, such that the methods can be fairly and comprehensively compared. In Section 3.1, we list the design considerations to achieve this motivation. In Section 3.2, we describe the tasks and datasets included in ELUE. In Section 3.3, we illustrate how to make a submission on ELUE, and how the submission is evaluated. In Section 3.4, we discuss the design of our leaderboard.

3.1 Design Considerations

Now we enumerate main considerations in the design of ELUE to ensure that it meets the needs mentioned early.

Multi-dimensional Evaluation The evaluation of ELUE should be multi-dimensional for comprehensive comparison. Instead of point-to-point comparison, methods can be compared in a line-to-line style in ELUE, where the "line" is a performance-efficiency trade-off curve.

Public Accessible All data points in ELUE should be publicly accessible such that the following work does not need to reproduce or estimate results from previous work. To facilitate future research, some representative methods should be reproduced and evaluated in ELUE.

Standard Evaluation The measurement of model efficiency should be standardized in ELUE such that this line of methods can be fairly compared. Current studies usually use number of parameters (Lan et al., 2020; Jiao et al., 2020), FLOPs (Jiao et al., 2020; Liu et al., 2020; Li et al., 2021b), actual inference time (Sanh et al., 2019; Schwartz et al., 2020), or number of executed layers (Zhou et al., 2020a; Sun et al., 2021b) to measure model efficiency. Among these metrics, measuring actual inference time is costly for both users and the server, and highly depends on the computation infrastructure and software implementation, while number of executed layers ignores the shape of input and hidden layers, therefore is inaccurate. Thus, ELUE adopts number of parameters and FLOPs as the metrics for model efficiency.

Easy-to-Use ELUE should be friendly to users, which means that the submission should be as simple as possible. Roughly speaking, there are currently two ways of submissions: (1) submitting the trained model such as SQuAD (Rajpurkar et al., 2016), Dynabench (Kiela et al., 2021), and (2) submitting the predicted test files such as GLUE (Wang et al., 2019b), SuperGLUE (Wang et al., 2019a), and CLUE (Xu et al., 2020b). The submission of ELUE lies in the latter way. Nevertheless, to evaluate number of parameters and FLOPs, the submitted test files should conform to a specific format, and besides, a Python file to define the used model is also required. For more details about submission and evaluation, see Section 3.3.

3.2 Task and Dataset Selection

Following GLUE (Wang et al., 2019b), SuperGLUE (Wang et al., 2019a), and CLUE (Xu et al., 2020b), we collect tasks that can be formatted as single sentence classification or sentence pair classification. Since ELUE mainly focuses on efficient models, the difficulty of dataset is not a primary consideration. Instead, we collect tasks and datasets that are commonly used and publicly available in the community. The statistics of the collected datasets are listed in Table 1.

Sentiment Analysis Sentiment analysis, which is to classify the polarity of a given text, is a fundamental task in NLP. We select two well-known movie review datasets, Stanford Sentiment Treebank (SST) (Socher et al., 2013) and IMDb (Maas et al., 2011). For SST, we use the two-way class split, i.e. SST-2. Different from GLUE, SST-2 samples in ELUE are complete sentences instead of phrases. For IMDb, we randomly select 2.5k positive samples and 2.5k negative samples from training set to construct a development set.

Tasks	Datasets	|Train|	|Dev|	|Test|
Sentiment Analysis	SST-2	8,544	1,101	2,208
	IMDb	20,000	5,000	25,000
Natural Language Inference	SNLI	549,367	9,842	9,824
	SciTail	23,596	1,304	2,126
Similarity and Paraphrase	MRPC	3,668	408	1,725
	STS-B	5,749	1,500	1,379

Table 1: Statistics of datasets in ELUE.

Natural Language Inference Natural language inference (NLI) is a task to predict whether the premise entails the hypothesis, contradicts the hypothesis, or neither. NLI is often formulated as a sentence pair classification task (Devlin et al., 2019; Sun et al., 2021a). We select two NLI datasets, SNLI (Bowman et al., 2015) and SciTail (Khot et al., 2018). SNLI is a crowd-sourced collection of sentence pairs with balanced labels: *entailment*, *contradiction*, and *neutral*. We use the spell-checked version of the test and development sets[3]. The hard samples, which do not have golden labels due to the disagreement of annotators, are removed from the dataset and left for model diagnostic. SciTail is a two-way (*entail* or *neutral*) entailment classification dataset, which is derived from multiple-choice science exams and web sentences.

Similarity and Paraphrase For similarity and paraphrase tasks, we also select two datasets, Microsoft Research Paraphrase Corpus (MRPC) (Dolan and Brockett, 2005), and Semantic Textual Similarity Benchmark (STS-B) (Cer et al., 2017), both of which are also included in GLUE. MRPC is a collection of automatically extracted sentence pairs, each manually-labeled with a judgment to indicate whether the pair constitutes a paraphrase. STS-B is a corpus of sentence pairs, each of which is labeled with a score from 0 to 5 to represent the degree to which two sentences are semantically equivalent.

3.3 Submission and Evaluation

ELUE supports two kinds of submissions: submitting test files, or submitting from a paper.

[3] https://nlp.stanford.edu/projects/snli/

Submit test files Users are required to submit two kinds of files: (1) predicted test files, and (2) a model definition file in Python. The predicted test files can be multiple, each indicates the prediction under a certain efficiency. The submitted test files should be in the following format:

index	pred	modules
0	1	(10),emb; (10,768),layer_1; (768),exit_1
1	0	(15),emb; (15,768),layer_1; (768),exit_1; (15,768),layer_2; (768),exit_2
2	1	(12),emb; (12,768),layer_1; (768),exit_1
...	...	...

Different from traditional predicted test files as in GLUE, an additional column "modules" is required to indicate the activated modules to predict each sample. The numbers before each module represent the input shape of that module, e.g. the "(10)" before "emb" indicates that the input of "emb" is a sequence of length 10. Note that this format is also compatible with token-level early exiting methods (Li et al., 2021b), where the sequence length is progressively reduced as the processing of layers.

Along with the test files, a Python file to define the model is also required. The following is an example Python file using PyTorch (Paszke et al., 2019) and Transformers (Wolf et al., 2020).

```python
# import packages
import torch.nn as nn
from transformers import BertConfig
...

# module definitions
class ElasticBERTEmbeddings(nn.Module):
    def __init__():
        ...
    def forward(x):
        ...

class ElasticBERTLayer(nn.Module):
    def __init__():
        ...
    def forward(x)
        ...

class ElasticBERT(nn.Module):
    def __init__():
        ...
    def forward(x)
        ...

# module dict
config = BertConfig(num_labels=2)
module_list = {
    'emb': ElasticBertEmbeddings(config),
    'layer_1': ElasticBertLayer(config),
    'exit_1': nn.Linear(config.hidden_size, num_labels),
    'layer_2': ElasticBertLayer(config),
    'exit_2': nn.Linear(config.hidden_size, num_labels),
    ...
}
entire_model = ElasticBERT(config)
```

With the submitted Python file, ELUE is able to evaluate the average FLOPs on a dataset, and the number of parameters of the model.

In cases that the evaluation is not applicable, e.g. the programming language, or dependencies of the submitted Python file is not supported in ELUE, the user is allowed to evaluate FLOPs and number of parameters by themselves and upload their results along with the predictions to the ELUE website.

Submit from a paper Inspired by Paper with Code[4], we also expect that ELUE can serve as an open-source platform that can facilitate future research. Therefore, there is a track for the authors of published papers to share their experimental results on ELUE datasets.

Performance Metrics Since the classes in MRPC are imbalanced, we report the unweighted average of accuracy and F1 score. For STS-B, we evaluate and report the Pearson and Spearman correlation coefficients. For other datasets, we simply adopt accuracy as the metric.

3.4 Leaderboard

Following prior work (Yang et al., 2018; Wang et al., 2019b; Xu et al., 2020b), we also integrate a leaderboard in ELUE. For dynamic models that have multiple performance-FLOPs coordinates on each dataset, we need to sum up these coordinates as a score. A critical problem is to measure how good a coordinate is. In other words, to measure a coordinate (p, f), where p is performance and f is FLOPs, we need a baseline performance under the same FLOPs. We choose ElasticBERT as the baseline curve. We evaluate different layers of ElasticBERT, and obtained 12 coordinates $(p_i^{EB}, f_i^{EB})_{i=1}^{12}$, which are then used to interpolate to get a performance-FLOPs function $p^{EB}(f)$. With the baseline curve at hand, we can score a submission curve as

$$score = \frac{1}{n}\sum_{i=1}^{n}[p_i - p^{EB}(f_i)]. \qquad (1)$$

Note that the coordinates of ElasticBERT are separately interpolated on different datasets. The final ELUE score is an unweighted average of the scores on all the 6 datasets. Figure 2 gives an illustration of how ELUE score is computed. The ELUE score reflects the extent to which the submission oversteps the ElasticBERT, which can be seen as the current Pareto front.

In addition, following EfficientQA (Min et al., 2020), ELUE leaderboard also maintains four additional separate tracks, corresponding to models below 40M, 55M, 70M, 110M parameters. Models

[4] https://paperswithcode.com/

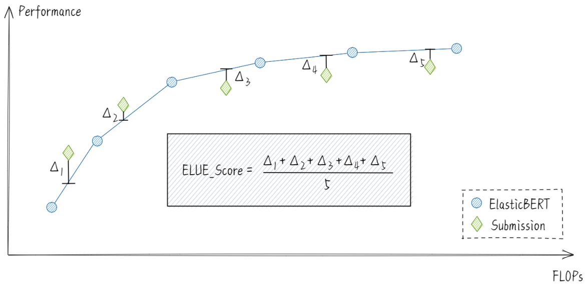


Figure 2: An illustration to show how ELUE score is computed.

in these tracks are ranked by the average performance on all the datasets.

4 ElasticBERT: A Strong Baseline for Efficient Inference

Despite the encouraging results achieved by existing efficient models, we argue that a strong baseline (backbone) is needed for both static methods and dynamic methods. Static methods often choose the first few layers of some pre-trained model as their baseline (e.g. Sun et al. (2019); Jiao et al. (2020)), which can be weak. Dynamic methods that enable early exiting by training multiple internal classifiers usually introduce a gap between pre-training and fine-tuning, and therefore hurt the performance of the entire model (Xin et al., 2021). Thus, in this section, we present the ElasticBERT that bridges the gap between static and dynamic methods, and therefore can serve as a strong baseline for static methods and also a strong backbone for dynamic methods.

ElasticBERT is a multi-exit pre-trained language model with the following training objective:

$$\mathcal{L} = \sum_{l=1}^{L} (\mathcal{L}_l^{\text{MLM}} + \mathcal{L}_l^{\text{SOP}}), \tag{2}$$

where L is the total number of layers, $\mathcal{L}^{\text{MLM}}$ is the n-gram masked language modeling loss, $\mathcal{L}^{\text{SOP}}$ is the sentence order prediction loss (Lan et al., 2020). The two losses are applied to each layer of the model, such that the number of layers can be flexibly scaled on downstream tasks, and therefore it is named "ElasticBERT".

Bridge the Gap Between Static and Dynamic Methods As a baseline for static methods, the depth of ElasticBERT can be flexibly reduced on demand. Compared with the first l layer of BERT (Devlin et al., 2019), the l-layered ElasticBERT is a complete model (Turc et al., 2019; Li et al., 2021a) and can achieve better performance. It is worth noticing that ElasticBERT can be regarded as a special instance of LayerDrop (Fan et al., 2020) where the dropped layers are constrained to the top consecutive layers. As a backbone for dynamic methods, training classifiers injected in intermediate layers would be consistent with pre-training. Therefore, ElasticBERT can not only be used as a static complete model, but also be used as a backbone model of dynamic early exiting.

Gradient Equilibrium Pre-training with the simply summed loss in Eq. (2) could lead to a *gradient imbalance* issue (Li et al., 2019). In particular, due to the overlap of subnetworks, the variance of the gradient may grow overly large, leading to unstable training. To address this issue, we follow Li et al. (2019) and adopt the gradient equilibrium (GE) strategy[5] in the pre-training of ElasticBERT.

Grouped Training In our preliminary experiments, we found that summing up losses at all layers could slow down pre-training and increase memory footprints. To alleviate this, we divide L exits into G groups. During training, we optimize the losses of the exits within each group by cycling alternately between different batches:

$$\mathcal{L} = \sum_{l \in \mathcal{G}_i} (\mathcal{L}_l^{\text{MLM}} + \mathcal{L}_l^{\text{SOP}}). \tag{3}$$

In Section 5.3 we explore the performance of different grouping methods. As a result, we group the 12 exits of ElasticBERT$_{\text{BASE}}$ into $\mathcal{G}_1$={1, 3, 5, 7, 9, 11, 12} and $\mathcal{G}_2$={2, 4, 6, 8, 10, 12}, and group the 24 exits of ElasticBERT$_{\text{LARGE}}$ into $\mathcal{G}_1$={1, 4, 7, ..., 22, 24}, $\mathcal{G}_2$={2, 5, 8, ..., 23, 24}, and $\mathcal{G}_3$={3, 6, 9, ..., 21, 24}. Our experiments demonstrate that grouped training can significantly speedup the process of pre-training without a loss in performance.

5 Experiments

5.1 Experimental Setups

Pre-training Setup Following BERT (Devlin et al., 2019), we train ElasticBERT in two different configurations: ElasticBERT$_{\text{BASE}}$ and ElasticBERT$_{\text{LARGE}}$, which have the same model

[5] The reader is referred to the original paper for more details. In brief, the gradients of $\mathcal{L}_j$ w.r.t. the parameters of the i-th layer ($i < j$) would be properly rescaled.

Models	#Params	CoLA	MNLI-(m/mm)	MRPC	QNLI	QQP	RTE	SST-2	STS-B	Average
				BASE Models						
$\text{BERT}_{\text{BASE}}$	109M	56.5	84.6/84.9	87.6	91.2	89.6	69.0	92.9	89.4	82.9
$\text{ALBERT}_{\text{BASE}}$	12M	56.8	84.9/85.6	90.5	91.4	89.2	**78.3**	92.8	90.7	84.5
$\text{RoBERTa}_{\text{BASE}}$	125M	63.6	**87.5/87.2**	90.8	**92.7**	**90.3**	77.5	**94.8**	**90.9**	**86.1**
ElasticBERT$_{\text{BASE}}$	109M	**64.3**	85.3/85.9	**91.0**	92.0	90.2	76.5	94.3	90.7	85.6
$\text{BERT}_{\text{BASE}}$-6L	67M	44.6	81.4/81.4	84.9	87.4	88.7	65.7	90.9	88.1	79.2
$\text{ALBERT}_{\text{BASE}}$-6L	12M	52.4	82.6/82.2	89.0	89.8	88.7	70.4	90.8	89.6	81.7
$\text{RoBERTa}_{\text{BASE}}$-6L	82M	44.4	84.2/**84.6**	87.9	90.5	**89.8**	60.6	92.1	89.0	80.3
MobileBERT	25M	52.1	83.9/83.5	89.3	**91.3**	88.9	63.5	91.3	87.2	81.2
TinyBERT-6L	67M	46.3	83.6/83.8	88.7	90.6	89.1	73.6	92.0	89.4	81.9
ElasticBERT$_{\text{BASE}}$-6L	67M	**53.7**	**84.3**/84.2	**89.7**	90.8	89.7	**74.0**	**92.7**	**90.2**	**83.3**
				Test Set Results						
TinyBERT-6L	67M	42.5	83.2/82.4	86.2	89.6	79.6	**73.0**	91.8	85.7	79.3
ElasticBERT$_{\text{BASE}}$-6L	67M	**49.1**	**83.7/83.4**	**87.3**	**90.4**	**79.7**	68.7	**92.9**	**86.9**	**80.3**
				LARGE Models						
$\text{BERT}_{\text{LARGE}}$	335M	61.6	86.2/86	90.1	92.2	90.1	72.9	93.5	90.4	84.8
$\text{ALBERT}_{\text{LARGE}}$	18M	60.1	86/86.1	90.4	91.6	89.6	83.0	95.2	91.4	85.9
$\text{RoBERTa}_{\text{LARGE}}$	355M	**66.4**	**89/89.6**	91.6	**94.2**	90.7	**86.6**	**95.4**	**92.3**	**88.4**
ElasticBERT$_{\text{LARGE}}$	335M	66.3	88/88.5	**92.0**	93.6	**90.9**	83.1	95.3	91.7	87.7
$\text{BERT}_{\text{LARGE}}$-12L	184M	42.6	81/81.1	81.6	87.2	89.3	65.7	89.3	88.7	78.5
$\text{ALBERT}_{\text{LARGE}}$-12L	18M	59.0	85.3/85.8	**90.1**	91.4	89.6	76.7	93.3	91.3	84.7
$\text{RoBERTa}_{\text{LARGE}}$-12L	204M	**62.3**	**86.3**/86.2	89.4	92.3	90.4	71.8	**93.5**	91.1	84.8
ElasticBERT$_{\text{LARGE}}$-12L	184M	62.1	86.2/**86.4**	89.5	**92.5**	**90.6**	**79.1**	93.0	**91.6**	**85.7**
				Test Set Results						
$\text{RoBERTa}_{\text{LARGE}}$-12L	204M	**59.4**	**86.4/85.2**	87.6	91.6	80.4	67.3	**94.6**	89.5	82.4
ElasticBERT$_{\text{LARGE}}$-12L	184M	57.0	85.4/84.9	**87.7**	**92.3**	**81.2**	**71.8**	92.9	**89.7**	**82.6**

Table 2: ElasticBERT and static baseline performance on GLUE tasks. For MRPC, we report the mean of accuracy and F1. For STS-B, we report Pearson and Spearman correlation. For CoLA, we report Matthews correlation. For all other tasks we report accuracy.

sizes with $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$, respectively. The parameters of ElasticBERT are initialized with BERT, and therefore it has the same vocabulary and tokenizer as BERT. ElasticBERT is pre-trained on ∼160GB uncompressed English text corpora, which is comprised of English Wikipedia (12GB), BookCorpus (4GB) (Zhu et al., 2015), OpenWebText (38GB) (Gokaslan and Cohen, 2019), and part of the C4 corpus (110GB) (Raffel et al., 2020). We use Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.999$ to pre-train $\text{ElasticBERT}_{\text{BASE}}$ and $\text{ElasticBERT}_{\text{LARGE}}$ for 125K steps with the batch size of 4096 and learning rate of 2e-4. Our implementation is based on Huggingface's Transformers (Wolf et al., 2020) and the Megatron-LM toolkit (Shoeybi et al., 2019). ElasticBERT is trained on 64 32G NVIDIA Tesla V100 GPUs.

Downstream Evaluation We evaluate ElasticBERT on the ELUE benchmark, as a static model and as a dynamic model. As a static model, we evaluate different layers of ElasticBERT, denoted as ElasticBERT-nL. As a dynamic model, we inject and train internal classifiers in $\text{ElasticBERT}_{\text{BASE}}$ and adopt two strategies, entropy (Xin et al., 2020) and patience (Zhou et al., 2020a), to enable early exiting, denoted as $\text{ElasticBERT}_{\text{entropy}}$ and $\text{ElasticBERT}_{\text{patience}}$. To compare with previous work, we also evaluate ElasticBERT on the GLUE benchmark (Wang et al., 2019b). For static usage, we fine-tune ElasticBERT and our baselines for 10 epochs using AdamW optimizer (Loshchilov and Hutter, 2019) with learning rates of {1e-5, 2e-5, 3e-5} and batch size of 32. For dynamic usage, we train for 5 epochs with the same optimization setup as static scenario.

Baselines We compare ElasticBERT with three types of baselines: **(1)** Directly fine-tuning pre-trained models and their first n layers. We choose BERT (Devlin et al., 2019), ALBERT (Lan et al., 2020) and RoBERTa (Liu et al., 2019) as our baselines. For the use of the first n layers, we sim-

Models	#Params	#FLOPs	SST-2	IMDb	MRPC	STS-B	SNLI	SciTail	Average
				BASE Models					
$\text{BERT}_{\text{BASE}}$	109M	13687M	85.1	93.0	83.1	84.2	90.4	93.2	88.2
$\text{ALBERT}_{\text{BASE}}$	12M	14018M	86.6	92.9	87.8	88.3	90.1	93.4	89.9
$\text{RoBERTa}_{\text{BASE}}$	125M	13188M	88.3	**94.9**	**88.0**	**89.6**	91.3	92.8	**90.8**
$\text{ElasticBERT}_{\text{BASE}}$	109M	13687M	**88.6**	93.9	87.9	87.6	**91.3**	**93.8**	90.5
$\text{BERT}_{\text{BASE}}$-6L	67M	6788M	83.3	91.0	82.6	82.5	88.9	90.7	86.5
$\text{ALBERT}_{\text{BASE}}$-6L	12M	7063M	84.7	92.0	85.3	83.5	89.3	92.3	87.9
$\text{RoBERTa}_{\text{BASE}}$-6L	82M	6638M	86.8	92.6	86.7	84.5	**90.2**	91.3	88.7
MobileBERT	25M	3622M	86.5	92.6	85.9	83.4	89.8	**93.6**	88.6
TinyBERT-6L	67M	6788M	85.3	89.0	84.7	85.0	89.3	90.0	87.2
$\text{ElasticBERT}_{\text{BASE}}$-6L	67M	6788M	**87.0**	**92.7**	**87.3**	**86.9**	90.1	92.5	**89.4**
				LARGE Models					
$\text{BERT}_{\text{LARGE}}$	335M	47214M	87.9	94.0	85.9	86.7	90.8	93.9	89.9
$\text{ALBERT}_{\text{LARGE}}$	18M	49038M	87.7	93.8	88.1	89.3	90.2	93.6	90.5
$\text{RoBERTa}_{\text{LARGE}}$	355M	46194M	**90.5**	**95.7**	**89.9**	90.5	**91.6**	**95.8**	**92.3**
$\text{ElasticBERT}_{\text{LARGE}}$	335M	47214M	89.8	95.0	89.8	**90.9**	91.4	95.7	92.1
$\text{BERT}_{\text{LARGE}}$-12L	184M	23686M	81.4	90.7	78.6	83.9	89.6	90.3	85.8
$\text{ALBERT}_{\text{LARGE}}$-12L	18M	24611M	87.4	93.7	87.1	88.1	90.1	93.7	90.0
$\text{RoBERTa}_{\text{LARGE}}$-12L	204M	23174M	**89.0**	**94.3**	87.6	89.5	**91.3**	93.7	**90.9**
$\text{ElasticBERT}_{\text{LARGE}}$-12L	184M	23686M	88.6	94.2	**87.7**	**89.7**	91.2	**93.8**	**90.9**

Table 3: ElasticBERT and static baseline performance on ELUE task test sets. For MRPC, we report the mean of accuracy and F1. For STS-B, we report Pearson and Spearman correlation. For all other tasks we report accuracy. The reported FLOPs is the average over all the datasets.

ply add a linear classifier on top of the truncated model. **(2)** Compressed models. Compression techniques such as knowledge distillation are competitive baselines. Here we choose TinyBERT (Jiao et al., 2020)[6] and MobileBERT (Sun et al., 2020b) as our baselines. **(3)** Dynamic early exiting models. To verify the validity as a strong backbone of dynamic early exiting methods, we also compare $\text{ElasticBERT}_{\text{entropy}}$ and $\text{ElasticBERT}_{\text{patience}}$ with two representative early exiting models: DeeBERT (Xin et al., 2020) and PABEE (Zhou et al., 2020a), which adopts entropy-based and patience-based strategies, respectively, for early exiting.

5.2 Evaluating ElasticBERT on GLUE

To verify the effectiveness and the elasticity of ElasticBERT, we first evaluate ElasticBERT and our static baselines on the GLUE benchmark. We evaluate the first 6/12 layers of the BASE models, and the first 12/24 layers of the LARGE models. MobileBERT and TinyBERT are also included for comparison.

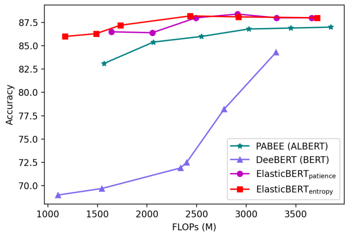
Experimental results of ElasticBERT and our baseline models on GLUE are presented in Table 2, from which we find that ElasticBERT outperforms BERT and ALBERT with the same number of layers, but is weaker than RoBERTa in the same configuration. Compared with ElasticBERT that is trained for 125K steps with batch size of 4K, RoBERTa is trained for 500K steps with batch size of 8K, which makes its training sample 8 times larger than that of ElasticBERT. When using fewer layers (6 layers of BASE models and 12 layers of LARGE models), ElasticBERT achieves the best performance among the static baselines, confirming its great elasticity.

5.3 Ablation Study

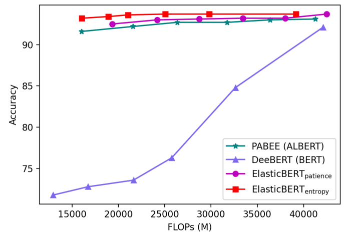
About the Training Strategy ElasticBERT adopts the gradient equilibrium (GE) to alleviate the conflict between the losses at different exits. Here, we compare GE with two other existing training strategies, two-stage training (Xin et al., 2020) and weighted training (Zhou et al., 2020a). Two-stage training is that, first training the top classifier along with the backbone model, and then freeze the parameters of the backbone model and train the injected internal classifiers. By this, two-stage training maintains the performance of the top classifier. Weighted training is to weight the loss of each exit according to the corresponding layer, which is

$$\mathcal{L} = \frac{\sum_{l=1}^{L} l \cdot \mathcal{L}_l}{\sum_{l=1}^{L} l}. \tag{4}$$

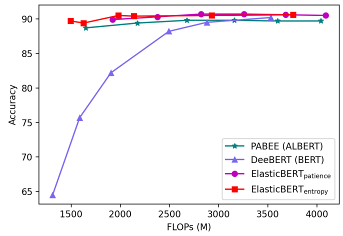
[6]Data augmentation in the TinyBERT paper is not used for fair comparison.



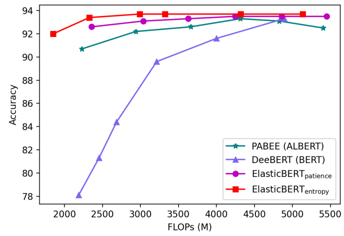
(a) SST-2



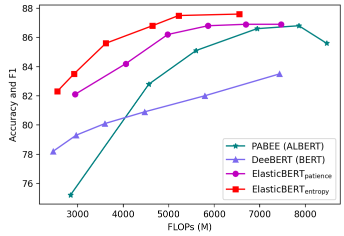
(b) IMDb



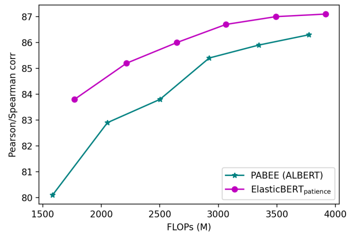
(c) SNLI



(d) SciTail

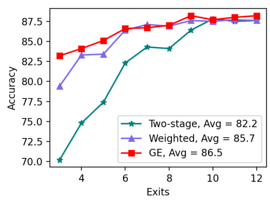


(e) MRPC

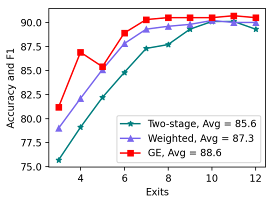


(f) STS-B

Figure 3: Performance-FLOPs trade-offs on ELUE task test sets. For MRPC, we present the mean of accuracy and F1. Because STS-B is a regression task, for which the entropy-based methods are not applicable, we only evaluate patience-based methods, i.e., PABEE and ElasticBERT$_{\text{patience}}$.



(a) SST-2



(b) MRPC

Figure 4: Performance of the ElasticBERT exits at different layers with different training strategies.

Experimental results of ElasticBERT with the three training strategies are shown in Figure 4. It can be observed that training with GE strategy performs the best on both SST-2 and MRPC.

About the Grouped Exits As shown in Eq. (3), we divide the L exits into different groups to speedup the pre-training. Therefore how to group these exits needs to be explored. Here we evaluate four different grouping methods, as described in Table 4. To keep the overall performance of the entire model, the exit classifier on the top of the model is included in each group. According to the experimental results in Table 4, we choose the odd/even grouping method for ElasticBERT$_{\text{BASE}}$. Similarly, our experiments demonstrate that grouping 24 exits into $\mathcal{G}_1$={1, 4, 7, ..., 22, 24}, $\mathcal{G}_2$={2, 5, 8, ..., 23, 24}, and $\mathcal{G}_3$={3, 6, 9, ..., 21, 24} works well for BERT$_{\text{LARGE}}$.

Grouping	**Accuracy**
w/o Grouping	**76.7**
$\mathcal{G}_1$={1, 3, 5, 7, 9, 11, 12} $\mathcal{G}_2$={2, 4, 6, 8, 10, 12}	**76.7**
$\mathcal{G}_1$={1, 4, 7, 10, 12} $\mathcal{G}_2$={2, 5, 8, 11, 12} $\mathcal{G}_3$={3, 6, 9, 12}	75.7
$\mathcal{G}_1$={1, 2, 3, 4, 12} $\mathcal{G}_2$={5, 6, 7, 8, 12} $\mathcal{G}_3$={9, 10, 11, 12}	75.5
$\mathcal{G}_1$={1, 2, 3, 4, 5, 6, 12} $\mathcal{G}_2$={7, 8, 9, 10, 11, 12}	75.9

Table 4: The average accuracy acorss all the BERT exits on the MNLI dataset with different grouping.

5.4 Evaluating ElasticBERT on ELUE

ElasticBERT and our baselines are also evaluated on ELUE tasks. For the BASE version of ElasticBERT, BERT, ALBERT, and RoBERTa, we evaluate the first 3/4/6/12 layers. For the LARGE version of the models, we evaluate the first 6/8/12/24 layers. For dynamic methods, we fine-tune ElasticBERT along with the injected internal classifiers using the gradient equilibrium (GE) strategy (Li et al., 2019), and adopt two different early exiting strategies: entropy-based strategy (Xin et al., 2020) and patience-based strategy (Zhou et al., 2020a).

Results of Static Models The performance of ElasticBERT and our baseline models on ELUE

	SST-2	IMDb	MRPC	STS-B	SNLI	SciTail	Average
ElasticBERT$_{\text{BASE}}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Static Models							
BERT$_{\text{BASE}}$	-4.6	-2.2	-5.9	-4.7	-1.5	-3.3	-3.7
ALBERT $_{\text{BASE}}$	-2.3	-1.1	**-0.3**	-2.6	-1.7	**-1.5**	**-1.6**
RoBERTa $_{\text{BASE}}$	**-0.9**	**-0.1**	-2.9	-5.1	**-0.7**	-3.3	-2.2
TinyBERT-6L	-1.7	-3.7	-2.6	**-1.6**	-0.8	-2.5	-2.2
Dynamic Models							
PABEE	-1.5	-0.3	-3.2	-2.3	-0.9	-0.6	-1.5
DeeBERT (BERT)	-12.3	-14.1	-5.2	-	-8.4	-6.4	-
DeeBERT (RoBERTa)	-2.3	-4.5	-3.1	-	-23.5	-9.9	-
ElasticBERT$_{\text{patience}}$	0.2	0.1	-1.1	**-0.3**	**0.0**	0.2	**-0.2**
ElasticBERT$_{\text{entropy}}$	**0.8**	**0.9**	**-0.4**	-	-0.1	**0.7**	-

Table 5: ELUE scores calculated using Eq. (1) for static and dynamic baseline models. '-' denotes that the dataset/metric is not applicable to the model. An online leaderboard is publicly available at http://eluebenchmark.fastnlp.top/.

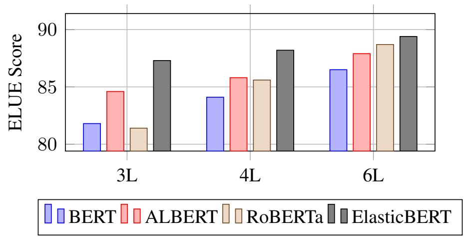
task test sets is shown in Table 3, where we find that ElasticBERT$_{\text{BASE}}$ and ElasticBERT$_{\text{LARGE}}$ outperform BERT and ALBERT with the same number of layers, but are slightly weaker than RoBERTa$_{\text{BASE}}$ and RoBERTa$_{\text{LARGE}}$. Besides, we find that the superiority of ElasticBERT over its baselines can be significant with fewer layers (See Figure 5 for the results of 3/4 (6/8) layers of the BASE (LARGE) models).

Results of Dynamic Models We compare ElasticBERT$_{\text{entropy}}$ and ElasticBERT$_{\text{patience}}$ with two dynamic models: DeeBERT (Xin et al., 2020) and PABEE (Zhou et al., 2020a). The performance-FLOPs trade-off of the dynamic models on ELUE task test sets are shown in Figure 3, which demonstrates that ElasticBERT can achieve better performance-FLOPs trade-off.

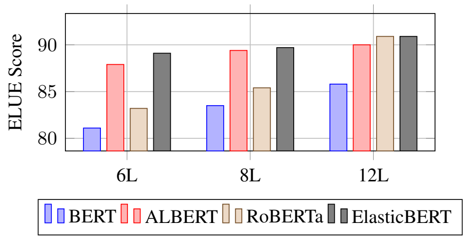
Evaluating ELUE Scores According to Eq. (1), we also evaluate the ELUE scores of these baselines. As shown in Table 5, the ELUE score of ElasticBERT$_{\text{BASE}}$ is natural to be zero on all the tasks. Among the other baselines, we find that ElasticBERT$_{\text{patience}}$ achieves the best ELUE score, while BERT$_{\text{BASE}}$ achieves the worst ELUE score. In addition, we find that dynamic models perform better than static models on average.

6 Conclusion and Future Work

In this work, we present ELUE, which is a public benchmark and platform for efficient models, and ElasticBERT, which is a strong baseline (backbone) for efficient static (dynamic) models. Both of the two main contributions are aimed to build



(a) BASE models.



(b) LARGE models.

Figure 5: Comparison of the average performance on ELUE task test sets of different layers of ElasticBERT and baselines.

the Pareto front for NLU tasks, such that the position of existing work can be clearly recognized, and future work can be easily and fairly measured.

Our future work is mainly in three aspects: (1) Including more baselines in ELUE, (2) Supporting FLOPs and parameters evaluation for more frameworks such as TensorFlow (Abadi et al., 2016), (3) Supporting diagnostics for submissions, (4) Dynamically updating the Pareto front and the corresponding computation of ELUE score.

Acknowledgements

We want to thank Hang Yan and Yining Zheng for their help in deploying the ELUE website.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016*, pages 265–283. USENIX Association.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 632–642. The Association for Computational Linguistics.

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

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity - multilingual and cross-lingual focused evaluation. *CoRR*, abs/1708.00055.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA).

Andrew S. Davis and Itamar Arel. 2014. Low-rank approximations for conditional feedforward computation in deep neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. 2019. Universal transformers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing, IWP@IJCNLP 2005, Jeju Island, Korea, October 2005, 2005*. Asian Federation of Natural Language Processing.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. Depth-adaptive transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus.

Mitchell A. Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing BERT: studying the effects of weight pruning on transfer learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2020, Online, July 9, 2020*, pages 143–155. Association for Computational Linguistics.

Alex Graves. 2016. Adaptive computation time for recurrent neural networks. *CoRR*, abs/1603.08983.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling BERT for natural language understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 4163–4174. Association for Computational Linguistics.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence,* *(AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5189–5197. AAAI Press.

Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. 2021. Dynabench: Rethinking benchmarking in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 4110–4124. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. 2019. Improved techniques for training adaptive deep networks. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 1891–1900. IEEE.

Lei Li, Yankai Lin, Deli Chen, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. 2021a. Cascadebert: Accelerating inference of pre-trained language models via calibrated complete models cascade. In *Findings of EMNLP*.

Xiaonan Li, Yunfan Shao, Tianxiang Sun, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2021b. Accelerating BERT inference for sequence labeling via early-exit. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 189–199. Association for Computational Linguistics.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2021. A survey of transformers. *CoRR*, abs/2106.04554.

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, ACL 2020, Online, July 5-10, 2020*, pages 6035–6044. Association for Computational Linguistics.

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. *CoRR*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 142–150. The Association for Computer Linguistics.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *CoRR*, abs/1806.08730.

Sewon Min, Jordan L. Boyd-Graber, Chris Alberti, Danqi Chen, Eunsol Choi, Michael Collins, Kelvin Guu, Hannaneh Hajishirzi, Kenton Lee, Jennimaria Palomaki, Colin Raffel, Adam Roberts, Tom Kwiatkowski, Patrick S. H. Lewis, Yuxiang Wu, Heinrich Küttler, Linqing Liu, Pasquale Minervini, Pontus Stenetorp, Sebastian Riedel, Sohee Yang, Minjoon Seo, Gautier Izacard, Fabio Petroni, Lucas Hosseini, Nicola De Cao, Edouard Grave, Ikuya Yamada, Sonse Shimaoka, Masatoshi Suzuki, Shumpei Miyawaki, Shun Sato, Ryo Takahashi, Jun Suzuki, Martin Fajcik, Martin Docekal, Karel Ondrej, Pavel Smrz, Hao Cheng, Yelong Shen, Xiaodong Liu, Pengcheng He, Weizhu Chen, Jianfeng Gao, Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Sejr Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Wen-tau Yih. 2020. Neurips 2020 efficientqa competition: Systems, analyses and lessons learned. In *NeurIPS 2020 Competition and Demonstration Track, 6-12 December 2020, Virtual Event / Vancouver, BC, Canada*, volume 133 of *Proceedings of Machine Learning Research*, pages 86–111. PMLR.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference* *on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *SCIENCE CHINA Technological Sciences*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. 2020. The right tool for the job: Matching model and instance complexities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6640–6651. Association for Computational Linguistics.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. 2020. Q-BERT: hessian based ultra low precision quantization of BERT. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8815–8821. AAAI Press.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *CoRR*, abs/1909.08053.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642. ACL.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4322–4331. Association for Computational Linguistics.

Tianxiang Sun, Xiangyang Liu, Xipeng Qiu, and Xuanjing Huang. 2021a. Paradigm shift in natural language processing. *CoRR*, abs/2109.12575.

Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang. 2020a. Colake: Contextualized language and knowledge embedding. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 3660–3670. International Committee on Computational Linguistics.

Tianxiang Sun, Yunhua Zhou, Xiangyang Liu, Xinyu Zhang, Hao Jiang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. 2021b. Early exiting with ensemble internal classifiers. *CoRR*, abs/2105.13792.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020b. Mobilebert: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2158–2170. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2021. Long range arena : A benchmark for efficient transformers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: The impact of student initialization on knowledge distillation. *CoRR*, abs/1908.08962.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–3275.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

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, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. 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, ACL 2020, Online, July 5-10, 2020*, 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, EACL 2021, Online, April 19 - 23, 2021*, pages 91–104. Association for Computational Linguistics.

Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2020a. Bert-of-theseus: Compressing BERT by progressive module replacing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7859–7869. Association for Computational Linguistics.

Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaoweihua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. 2020b. CLUE: A chinese language understanding evaluation benchmark. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 4762–4772. International Committee on Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian J. McAuley, Ke Xu, and Furu Wei. 2020a. BERT loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian J. McAuley, Ke Xu, and Furu Wei. 2020b. BERT loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2968–2980. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 19–27. IEEE Computer Society.

A Appendix

A.1 Benchmark Website Details

The ELUE website is built using Vue and Spring Boot. We use MySQL for data storage and our private servers to run the scoring script for each submission.