Towards Efficient NLP

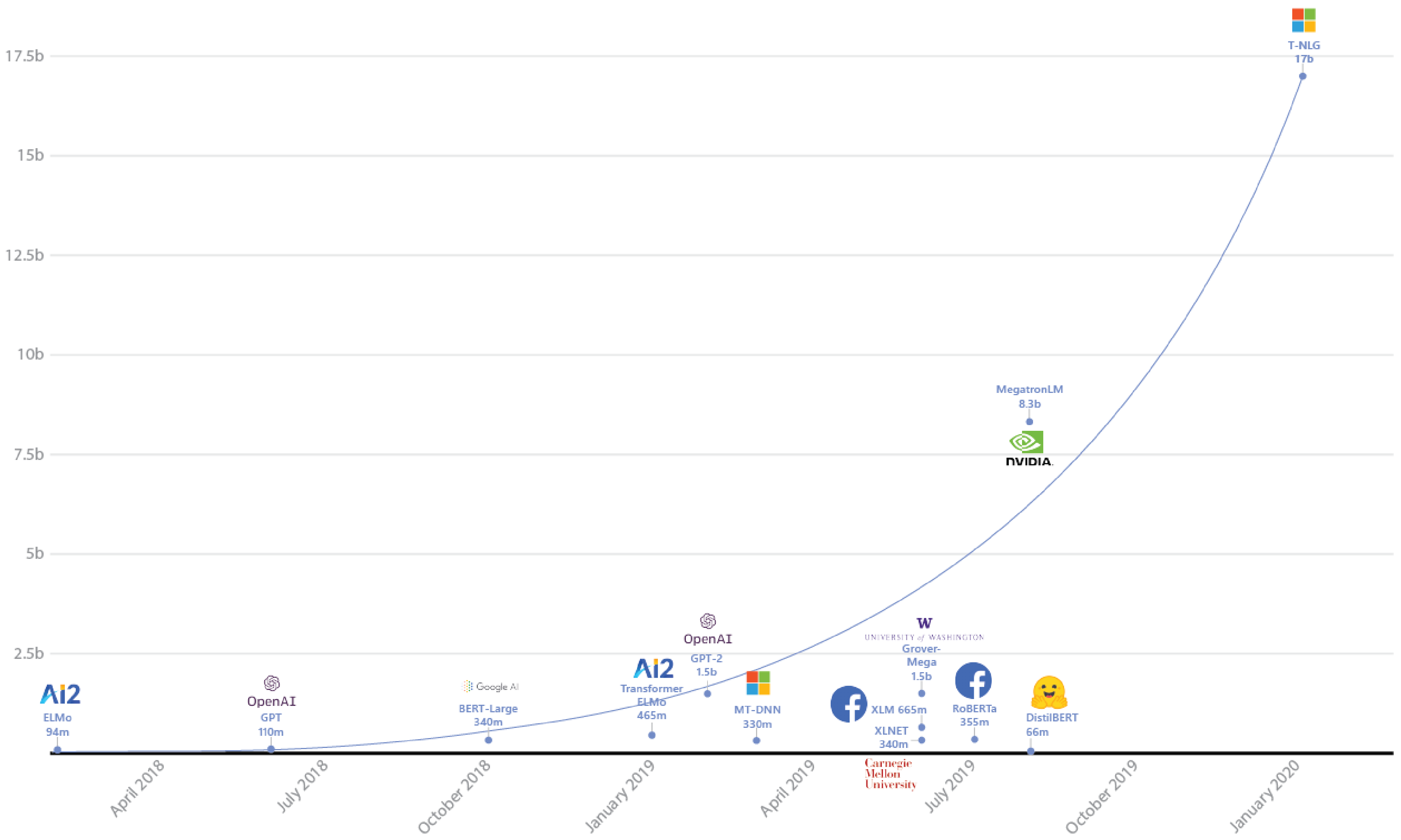
A Standard Evaluation and A Strong Baseline

Tianxiang Sun

txsun19@fudan.edu.cn

31 Oct 2021

The Era of Big Models



From SOTA to "Pareto SOTA"

- **The Shifted Goal**
  - Instead of pursuing the reachless SOTA accuracy, most works are pursuing improvement on other dimensions (like efficiency), leading to Pareto SOTA.
- **The Lagging Benchmarks**
  - Most of these works are evaluated on accuracy-centric benchmarks (e.g., GLUE, SuperGLUE, CLUE…

Need For A Standard Evaluation

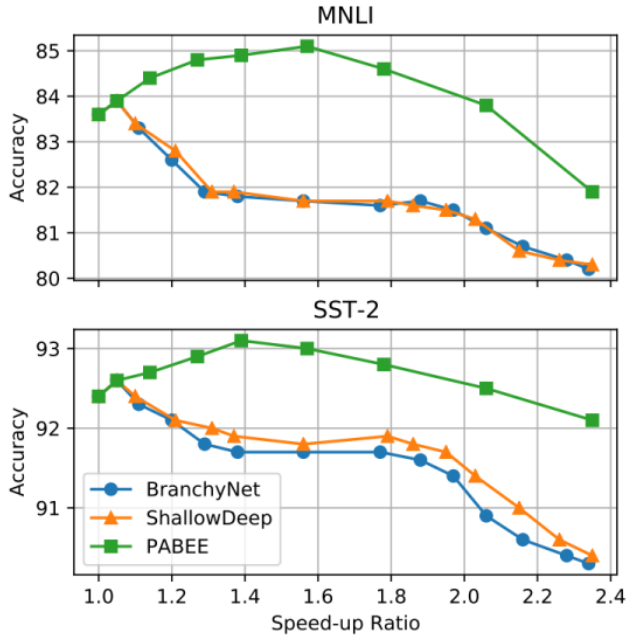
- **Incomprehensive Comparison**
  - Current comparison is usually point-to-point.

Method	Speed -up	CoLA (8.5k)	MRPC (3.7K)	QQP (364k)	RTE (2.5K)	SST-2 (67K)	Macro Avg.
				Dev Set			
ALBERT-base [3]	1.0×	58.9	89.5	89.6	78.6	92.8	81.9
ALBERT-6L	2.0×	53.4	85.8	86.8	73.6	89.8	77.9
ALBERT-9L	1.3×	55.2	87.1	88.3	75.9	91.3	79.6
LayerDrop [31]	2.0×	53.6	85.9	87.3	74.3	90.7	78.4
HeadPrune [32]	1.2×	54.1	86.2	88.0	75.1	90.5	78.8
DeeALBERT † [5]	1.5×	57.6	89.8	89.1	79.1	92.9	81.7
FastALBERT † [6]	1.5×	58.0	89.8	89.3	79.5	92.9	81.9
PABEE [8]	1.5×	61.2	90.0	89.6	80.1	93.0	82.8
Ours							
w/ Patience	1.5×	61.4	92.4	89.6	**80.9**	93.2	83.5
w/ Voting	1.5×	**61.6**	**92.7**	**89.8**	**80.9**	**93.5**	**83.7**
				Test Set			
ALBERT-base † [3]	1.0×	54.1	86.9	71.1	76.4	94.0	76.5
PABEE [8]	1.5×	55.7	87.4	71.2	77.3	94.1	77.1
Ours							
w/ Patience	1.5×	**56.2**	87.7	71.4	77.9	94.1	77.5
w/ Voting	1.5×	**56.2**	**88.0**	**71.5**	**78.2**	**94.4**	**77.7**

Sun et al. Early Exiting With Ensemble Internal Classifiers.

Need For A Standard Evaluation

- **Incomprehensive Comparison**
  - Current comparison is usually point-to-point.
- **Unaccessible Results**
  - The data points in line-to-line comparison are not publicly accessible.



We also compare LTE with the concurrent patience-based baseline PABEE (Zhou et al., 2020) in Table 3, showing their speedups and average exit layers at the same relative scores. PABEE does not provide exact speedup numbers; therefore we estimate the values from their figures. We can see that *Alternating* fine-tuning plus LTE is marginally better than PABEE on regression tasks.

Zhou et al. BERT Loses Patience: Fast and Robust Inference with Early Exit. NeurIPS 2020
Xin et al. BERxiT: Early Exiting for BERT with Better Fine-Tuning and Extension to Regression. EACL 2021.

Need For A Standard Evaluation

- **Incomprehensive Comparison**
  - Current comparison is usually point-to-point.
- **Unaccessible Results**
  - The data points in line-to-line comparison are not publicly accessible.
- **Non-standard Measurements**
  - Different works may adopt different metrics (FLOPs, physical time, number of layers, number of parameters…)
  - Even the same metric is adopted, the evaluation can be different (due to hardware infrastructure, software libraries, etc.)

Model		Parameters
BERT	base	108M
	large	334M
ALBERT	base	12M
	large	18M
	xlarge	60M
	xxlarge	235M

Model	# param. (Millions)	Inf. time (seconds)
ELMo	180	895
BERT-base	110	668
DistilBERT	66	410

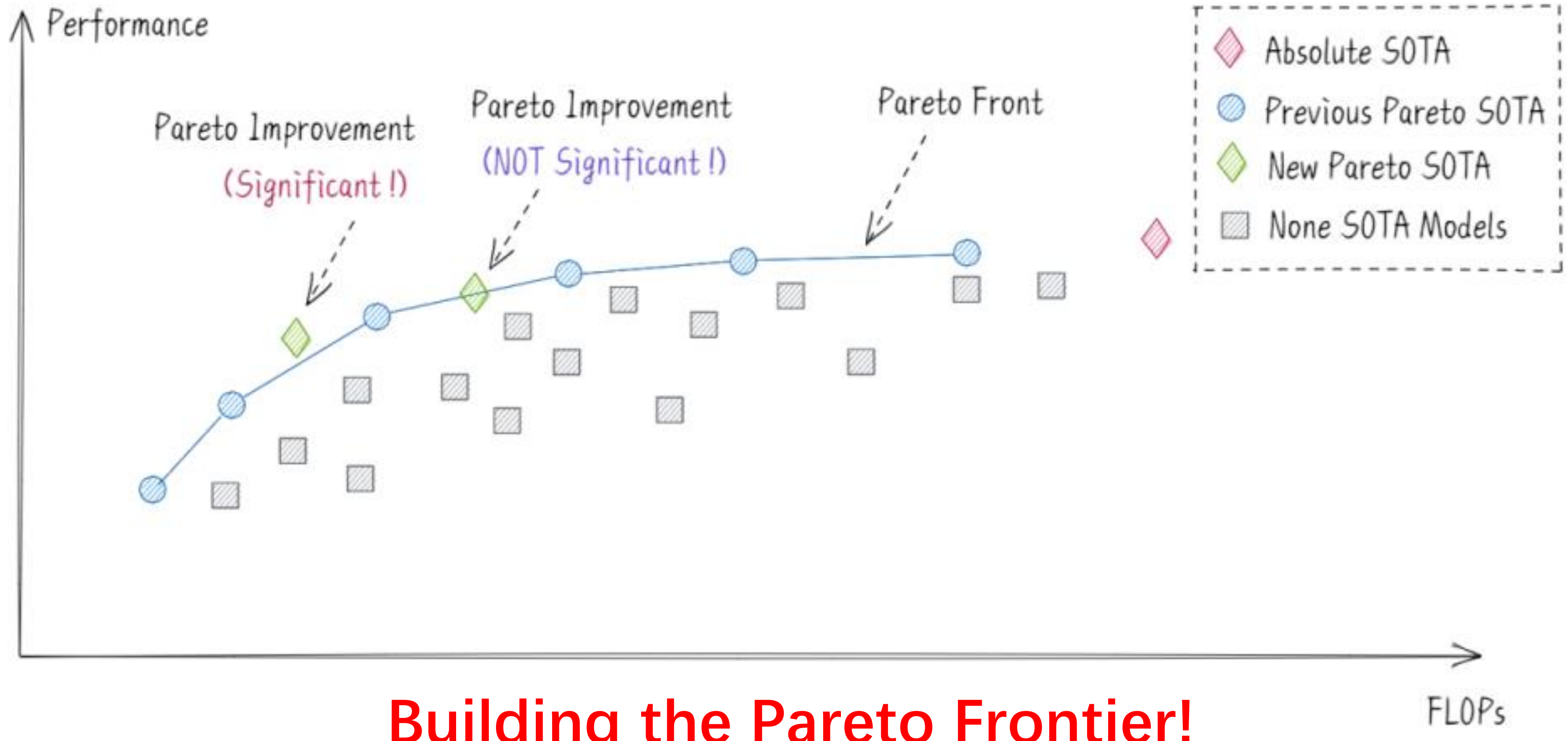
Dataset/ Model	**ChnSentiCorp** Acc.	FLOPs (speedup)
FastBERT (speed=0.1)	95.25	10741M (2.02x)
FastBERT (speed=0.5)	92.00	3191M (6.82x)
FastBERT (speed=0.8)	89.75	2315M (9.40x)

Need For A Standard Evaluation

- **Incomprehensive Comparison**
  - Current comparison is usually point-to-point.
- **Unaccessible Results**
  - The data points in line-to-line comparison are not publicly accessible.
- **Non-standard Measurements**
  - Different works may adopt different metrics (FLOPs, physical time, number of layers, number of parameters…)
  - Even the same metric is adopted, the evaluation can be different (due to hardware infrastructure, software libraries, etc.)
- **Inconvenience**
  - It is hard to plot an accuracy-speed curve on GLUE test set due to the submission limitation.

A New Benchmark Should…

- **Overall, it should tell you *whether* and *how much* a model achieves Pareto improvement.**



Building the Pareto Frontier!

A New Benchmark Should…

- **Multi-dimensional Evaluation** ~~(Incomprehensive Comparison)~~
  - Comprehensive line-to-line comparison
- **Public Accessible** ~~(Unaccessible Results)~~
  - Open source all the results to facilitate future research
- **Standard Evaluation** ~~(Non-standard Measurements)~~
  - Unified metrics (FLOPs and #params), and standardized evaluation toolkit
- **Easy-to-Use** ~~(Inconvenience)~~
  - Submission should be easy

ELUE

Efficient Language Understanding Evaluation

http://eluebenchmark.fastnlp.top

ELUE Tasks & Datasets

- **Sentiment Analysis**
  - SST-2
  - IMDb
- **Natural Language Inference**
  - SNLI
  - SciTail
- **Similarity and Paraphrase**
  - MRPC
  - STS-B

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

ELUE Submission

- **Submit a model, or submit a test file?**
  - **Submit a model (SQuAD-like)**

    😊 Easy to measure model efficiency

    ☹ Costly for submitting and serving

    ☹ Engineering and implementation matters too much

  - **Submit a test file (GLUE-like)**

    😊 Easy to submitting and evaluating

    ? But how to measure model efficiency

ELUE Submission

- **Submit test files, and model.py**

Test files

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
...	...	...

  - Combining the two kinds of files, we can calculate the FLOPs for each sample!
  - Token-level early exit and MoE models are also supported in this way

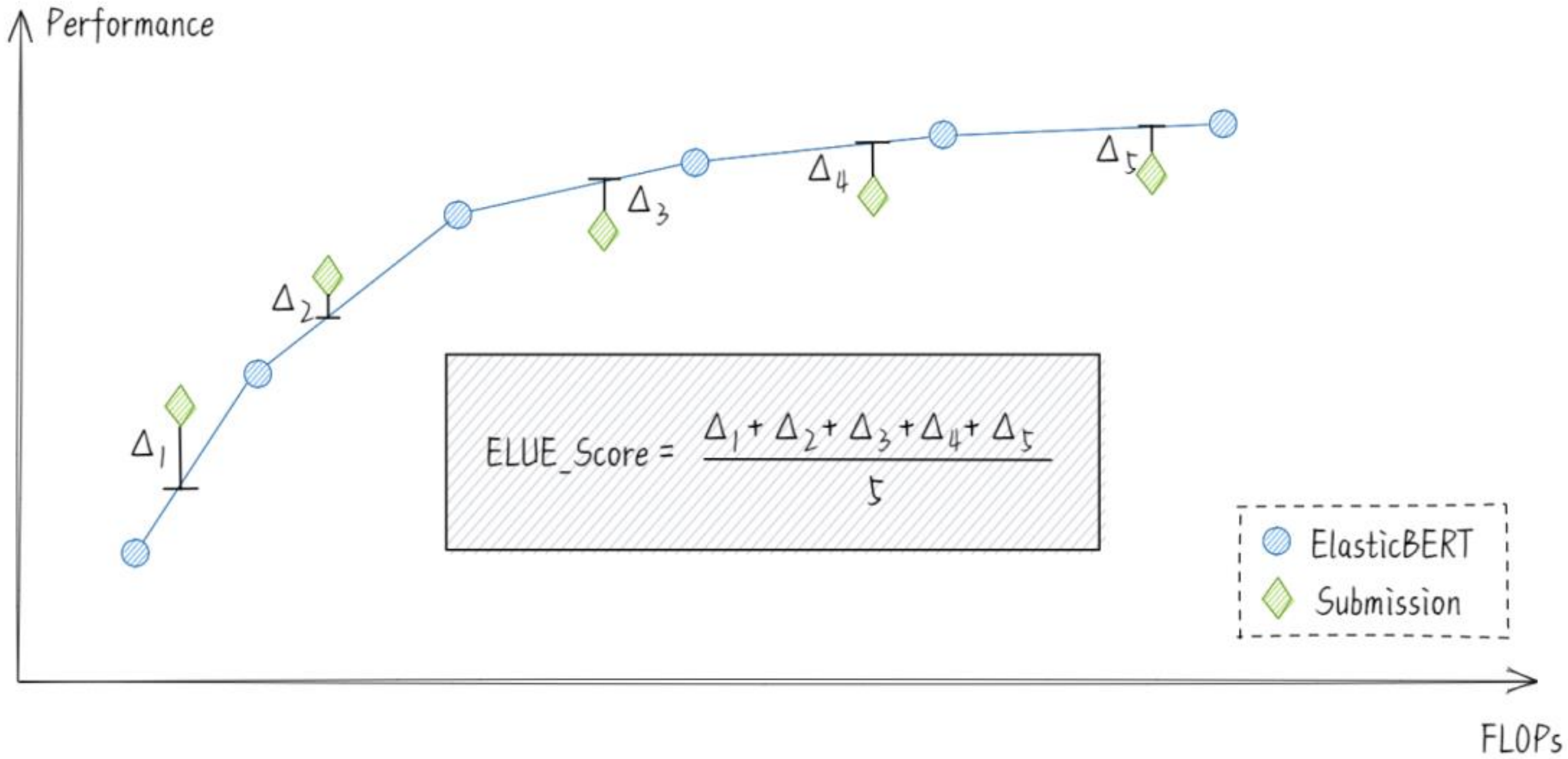
- **Submit from a paper**
  - Similar to paper-with-code

Model.py

```
# 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)
```

ELUE Leaderboard

- **How can we rank these efficient models?**
  - We need to score the (performance, FLOPs) pairs



ELUE Leaderboard

ELUE Score | ELUE (40M params) | ELUE (55M params) | ELUE (70M params) | ELUE (110M params)

Model	Sentiment Analysis		Natural Language Inference		Similarity and Paraphrase		Params (MParams)	Score
	SST-2	IMDb	SNLI	SciTail	MRPC	STS-B		
ElasticBERT-BASE	0.0	0.0	0.0	0.0	0.0	0.0	109.0	0
ElasticBERT-patience	0.37	0.2	0.02	0.38	-1.04	-0.45	116.0	-0.09
ALBERT-PABEE	-1.34	-0.22	-0.85	-0.4	-2.97	-2.1	18.0	-1.31
ALBERT-BASE	-2.3	-1.07	-1.66	-1.49	-0.31	-2.7	12.0	-1.59
RoBERTa-BASE	-0.9	-0.12	-0.69	-3.31	-2.86	-5.15	125.0	-2.17
BERT-BASE	-4.55	-2.15	-1.5	-3.35	-5.88	-4.75	109.0	-3.7

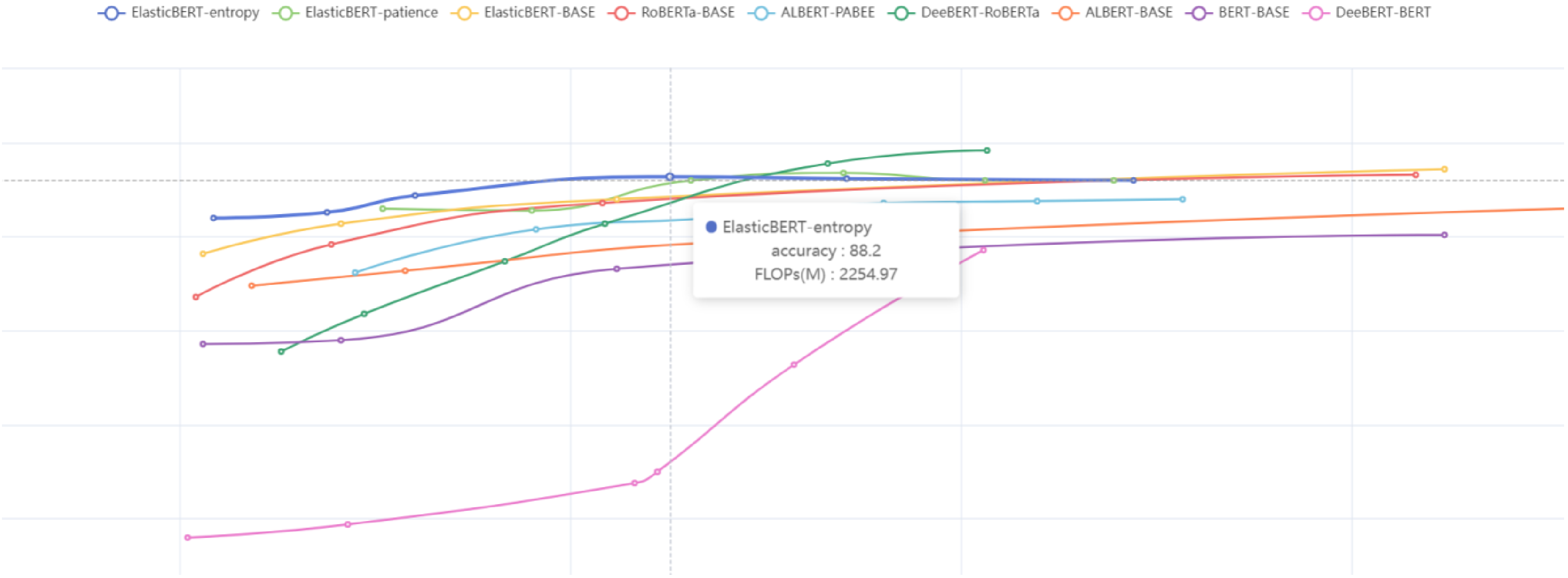
ELUE Task Page

Sentiment Analysis

Sentiment analysis is classifying one or more sentences by their positive/negative sentiment.

Choose Dataset | SST-2 | Download Dataset | Submit Paper | Submit Testfile | Submission Guide

Model Performance VS. FLOPs



Elastic BERT

https://github.com/fastnlp/ElasticBERT

ElasticBERT Pre-Training

- **Pre-trained Multi-Exit Transformer Encoder**

$$\mathcal{L} = \sum_{l=1}^{L} (\mathcal{L}_l^{\mathrm{MLM}} + \mathcal{L}_l^{\mathrm{SOP}})$$

- **Pre-trained on ~160GB English corpora**
  - Wikipedia, BookCorpus, OpenWebText, C4
- **Pre-trained for 125k steps on 64 32G V100**
- **Using *Gradient Equilibrium* and *Grouped Training* to stabilize and speedup pre-training**

ElasticBERT For Downstream Fine-Tuning

- **A Strong Baseline for Static Models**

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**

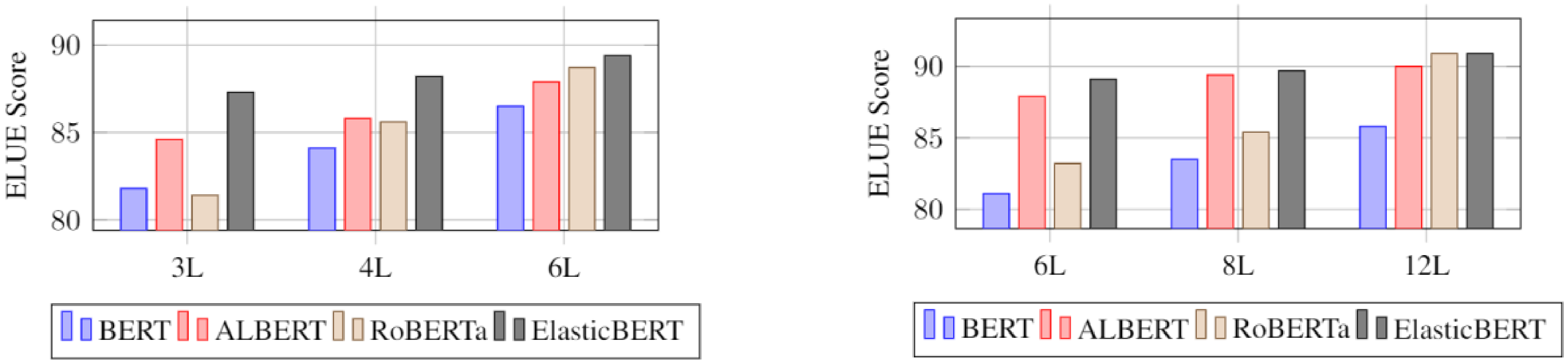
ElasticBERT For Downstream Fine-Tuning

- **A Strong Baseline for Static Models**

Models	#Params	CoLA	MNLI-(m/mm)	MRPC	QNLI	QQP	RTE	SST-2	STS-B	Average
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**

ElasticBERT For Downstream Fine-Tuning

- A Strong Baseline for Static Models

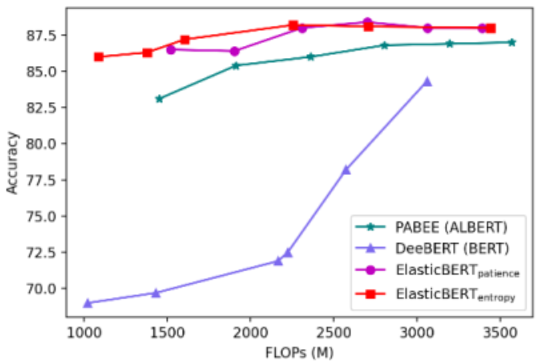


(a) BASE models.

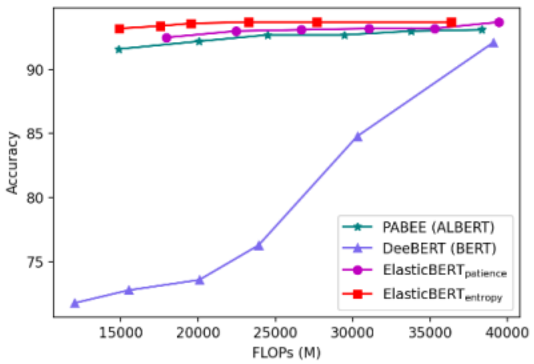
(b) LARGE models.

ElasticBERT For Downstream Fine-Tuning

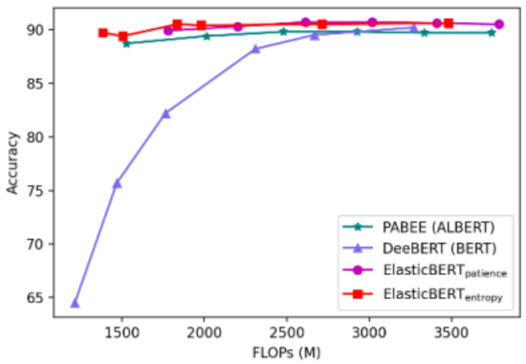
- A Strong Baseline for **Static** Models
- A Strong Backbone for **Dynamic** Models



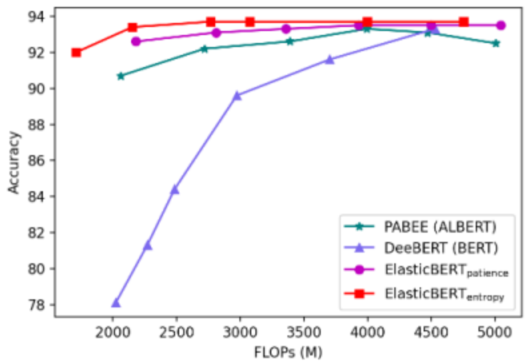
(a) SST-2



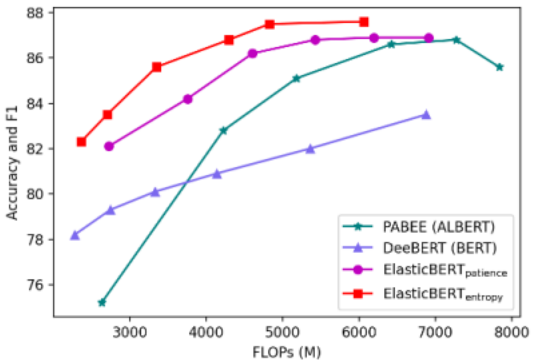
(b) IMDb



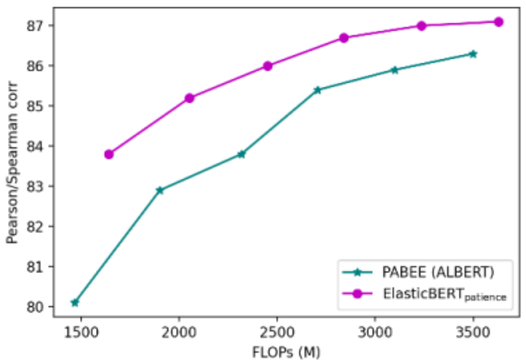
(c) SNLI



(d) SciTail



(e) MRPC



(f) STS-B

ElasticBERT on ELUE

	SST-2	**IMDb**	**MRPC**	**STS-B**	**SNLI**	**SciTail**	**Average**
ElasticBERT$_{\text{BASE}}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Static Models							
BERT$_{\text{BASE}}$	-4.55	-2.15	-5.88	-4.75	-1.50	-3.35	-3.70
ALBERT $_{\text{BASE}}$	-2.30	-1.07	**-0.31**	-2.70	-1.66	**-1.49**	**-1.59**
RoBERTa $_{\text{BASE}}$	**-0.90**	**-0.12**	-2.86	-5.15	**-0.69**	-3.31	-2.17
TinyBERT-6L	-1.70	-3.70	-2.60	**-1.90**	-0.80	-2.50	-2.20
Dynamic Models							
PABEE	-1.34	-0.22	-2.97	-2.10	-0.85	-0.40	-1.31
DeeBERT (BERT)	-12.10	-14.00	-4.92	-	-8.36	-6.17	-
DeeBERT (RoBERTa)	-1.98	-4.40	-2.72	-	-23.39	-9.82	-
ElasticBERT$_{\text{patience}}$	0.37	0.20	-1.04	**-0.46**	**0.02**	0.38	**-0.09**
ElasticBERT$_{\text{entropy}}$	**0.96**	**1.03**	**-0.22**	-	0.01	**0.69**	-

ElasticBERT Usage

- **An example using Huggingface Transformers**

```
>>> from transformers import BertTokenizer as ElasticBertTokenizer
>>> from models.configuration_elasticbert import ElasticBertConfig
>>> from models.modeling_elasticbert import ElasticBertForSequenceClassification
>>> num_output_layers = 1
>>> config = ElasticBertConfig.from_pretrained('fnlp/elasticbert-base', num_output_layers=num_output_layers )
>>> tokenizer = ElasticBertTokenizer.from_pretrained('fnlp/elasticbert-base’)
>>> model = ElasticBertForSequenceClassification.from_pretrained('fnlp/elasticbert-base', config=config)
>>> input_ids = tokenizer.encode('The actors are fantastic .', return_tensors='pt’)
>>> outputs = model(input_ids)
```

- **Github: https://github.com/fastnlp/ElasticBERT**
- **Huggingface: https://huggingface.co/fnlp/elasticbert-base**

Thanks!

https://arxiv.org/abs/2110.07038

http://eluebenchmark.fastnlp.top

https://github.com/fastnlp/ElasticBERT

https://huggingface.co/fnlp/elasticbert-base