FastBERT: a Self-distilling BERT with Adaptive Inference Time

Weijie Liu[1,2], Peng Zhou[2], Zhiruo Wang[3], Zhe Zhao[2], Haotang Deng[2] and Qi Ju[2,*]
[1]Peking University, Beijing, China
[2]Tencent Research, Beijing, China
[3]Beijing Normal University, Beijing, China
dataliu@pku.edu.cn, {rickzhou, nlpzhezhao, haotangdeng, damonju}@tencent.com, SherronWang@gmail.com

Abstract

Pre-trained language models like BERT have proven to be highly performant. However, they are often computationally expensive in many practical scenarios, for such heavy models can hardly be readily implemented with limited resources. To improve their efficiency with an assured model performance, we propose a novel speed-tunable FastBERT with adaptive inference time. The speed at inference can be flexibly adjusted under varying demands, while redundant calculation of samples is avoided. Moreover, this model adopts a unique self-distillation mechanism at fine-tuning, further enabling a greater computational efficacy with minimal loss in performance. Our model achieves promising results in twelve English and Chinese datasets. It is able to speed up by a wide range from 1 to 12 times than BERT if given different speedup thresholds to make a speed-performance tradeoff.

1 Introduction

Last two years have witnessed significant improvements brought by language pre-training, such as BERT (Devlin et al., 2019), GPT (Radford et al., 2018), and XLNet (Yang et al., 2019). By pre-training on unlabeled corpus and fine-tuning on labeled ones, BERT-like models achieved huge gains on many Natural Language Processing tasks.

Despite this gain in accuracy, these models have greater costs in computation and slower speed at inference, which severely impairs their practicalities. Actual settings, especially with limited time and resources in the industry, can hardly enable such models into operation. For example, in tasks like sentence matching and text classification, one often requires to process billions of requests per second. What's more, the number of requests varies with time. In the case of an online shopping site, the number of requests during the holidays is five to ten times more than that of the workdays. A large number of servers need to be deployed to enable BERT in industrial settings, and many spare servers need to be reserved to cope with the peak period of requests, demanding huge costs.

To improve their usability, many attempts in model acceleration have been made, such as quantinization (Gong et al., 2014), weights pruning (Han et al., 2015), and knowledge distillation (KD) (Romero et al., 2014). As one of the most popular methods, KD requires additional smaller student models that depend entirely on the bigger teacher model and trade task accuracy for ease in computation (Hinton et al., 2015). Reducing model sizes to achieve acceptable speed-accuracy balances, however, can only solve the problem halfway, for the model is still set as fixated, rendering them unable to cope with drastic changes in request amount.

By inspecting many NLP datasets (Wang et al., 2018), we discerned that the samples have different levels of difficulty. Heavy models may over-calculate the simple inputs, while lighter ones are prone to fail in complex samples. As recent studies (Kovaleva et al., 2019) have shown redundancy in pre-training models, it is useful to design a one-size-fits-all model that caters to samples with varying complexity and gains computational efficacy with the least loss of accuracy.

Based on this appeal, we propose FastBERT, a pre-trained model with a sample-wise adaptive mechanism. It can adjust the number of executed layers dynamically to reduce computational steps. This model also has a unique self-distillation process that requires minimal changes to the structure, achieving faster yet as accurate outcomes within a single framework. Our model not only reaches a comparable speedup (by 2 to 11 times) to the BERT model, but also attains competitive accuracy in comparison to heavier pre-training models.

*Corresponding author: Qi Ju (damonju@tencent.com)

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 6035–6044
July 5 - 10, 2020.

Experimental results on six Chinese and six English NLP tasks have demonstrated that FastBERT achieves a huge retrench in computation with very little loss in accuracy. The main contributions of this paper can be summarized as follows:

- This paper proposes a practical speed-tunable BERT model, namely FastBERT, that balances the speed and accuracy in the response of varying request amounts;
- The sample-wise adaptive mechanism and the self-distillation mechanism are combined to improve the inference time of NLP model for the first time. Their efficacy is verified on twelve NLP datasets;
- The code is publicly available at https://github.com/autoliuweijie/FastBERT.

2 Related work

BERT (Devlin et al., 2019) can learn universal knowledge from mass unlabeled data and produce more performant outcomes. Many works have followed: RoBERTa (Liu et al., 2019) that uses larger corpus and longer training steps. T5 (Raffel et al., 2019) that scales up the model size even more. UER (Zhao et al., 2019) pre-trains BERT in different Chinese corpora. K-BERT (Liu et al., 2020) injects knowledge graph into BERT model. These models achieve increased accuracy with heavier settings and even more data.

However, such unwieldy sizes are often hampered under stringent conditions. To be more specific, BERT-base contains 110 million parameters by stacking twelve Transformer blocks (Vaswani et al., 2017), while BERT-large expands its size to even 24 layers. ALBERT (Lan et al., 2019) shares the parameters of each layer to reduce the model size. Obviously, the inference speed for these models would be much slower than classic architectures (e.g., CNN (Kim, 2014), RNN (Wang, 2018), etc). We think a large proportion of computation is caused by redundant calculation.

Knowledge distillation: Many attempts have been made to distill heavy models (teachers) into their lighter counterparts (students). PKD-BERT (Sun et al., 2019a) adopts an incremental extraction process that learns generalizations from intermediate layers of the teacher model. TinyBERT (Jiao et al., 2019) performs a two-stage learning involving both general-domain pre-training and task-specific fine-tuning. DistilBERT (Sanh et al., 2019) further leveraged the inductive bias within large models by introducing a triple loss. As shown in Figure 1, student model often require a separated structure, whose effect however, depends mainly on the gains of the teacher. They are as indiscriminate to individual cases as their teachers, and only get faster in the cost of degraded performance.

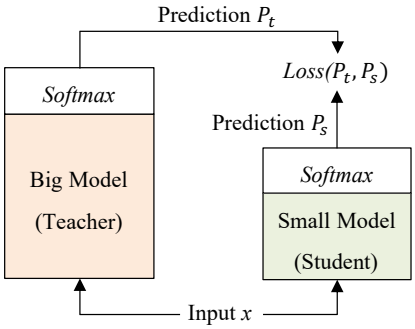

Figure 1: Classic knowledge distillation approach: Distill a small model using a separate big model.

Adaptive inference: Conventional approaches in adaptive computations are performed token-wise or patch-wise, who either adds recurrent steps to individual tokens (Graves, 2016) or dynamically adjusts the number of executed layers inside discrete regions of images (Teerapittayanon et al., 2016; Figurnov et al., 2017). To the best of our knowledge, there has been no work in applying adaptive mechanisms to NLP pre-training language models for efficiency improvements so far.

3 Methodology

Distinct to the above efforts, our approach fusions the adaptation and distillation into a novel speed-up approach, shown in Figure 2, achieving competitive results in both accuracy and efficiency.

3.1 Model architecture

As shown in Figure 2, FastBERT consists of backbone and branches. The backbone is built upon 12-layers Transformer encoder with an additional teacher-classifier, while the branches include student-classifiers which are appended to each Transformer output to enable early outputs.

3.1.1 Backbone

The backbone consists of three parts: the embedding layer, the encoder containing stacks of Transformer blocks (Vaswani et al., 2017), and the teacher classifier. The structure of the embedding layer and the encoder conform with those of BERT

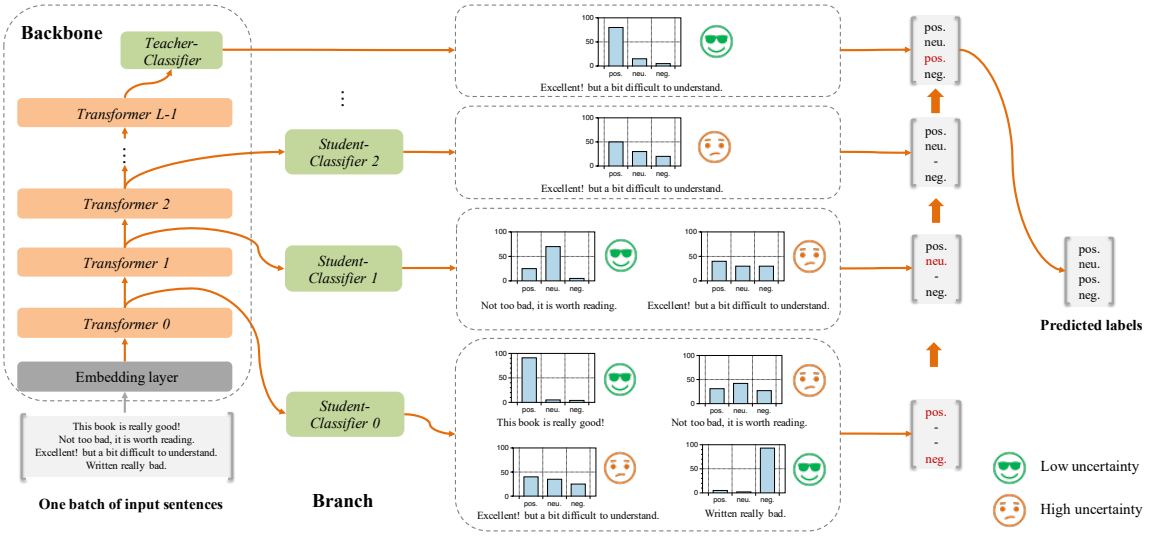

Figure 2: The inference process of FastBERT, where the number of executed layers with each sample varies based on its complexity. This illustrates a sample-wise adaptive mechanism. Taking a batch of inputs ($batch\_size = 4$) as an example, the *Transformer0* and *Student-classifier0* inferred their labels as probability distributions and calculate the individual uncertainty. Cases with low uncertainty are immediately removed from the batch, while those with higher uncertainty are sent to the next layer for further inference.

(Devlin et al., 2019). Given the sentence length n, an input sentence $s = [w_0, w_1, ...w_n]$ will be transformed by the embedding layers to a sequence of vector representations e like (1),

$$e = Embedding(s), \tag{1}$$

where e is the summation of word, position, and segment embeddings. Next, the transformer blocks in the encoder performs a layer-by-layer feature extraction as (2),

$$h_i = Transformer\_i(h_{i-1}), \tag{2}$$

where h_i ($i = -1, 0, 1, ..., L-1$) is the output features at the ith layer, and $h_{-1} = e$. L is the number of Transformer layers.

Following the final encoding output is a teacher classifier that extracts in-domain features for downstream inferences. It has a fully-connected layer narrowing the dimension from 768 to 128, a self-attention joining a fully-connected layer without changes in vector size, and a fully-connected layer with a $softmax$ function projecting vectors to an N-class indicator p_t as in (3), where N is the task-specific number of classes.

$$p_t = Teacher\_Classifier(h_{L-1}). \tag{3}$$

3.1.2 Branches

To provide FastBERT with more adaptability, multiple branches, i.e. the student classifiers, in the same architecture with the teacher are added to the output of each Transformer block to enable early outputs, especially in those simple cases. The student classifiers can be described as (4),

$$p_{s_i} = Student\_Classifier\_i(h_i). \tag{4}$$

The student classifier is designed carefully to balance model accuracy and inference speed, for simple networks may impair the performance, while a heavy attention module severely slows down the inference speed. Our classifier has proven to be lighter with ensured competitive accuracy, detailed verifications are showcased in Section 4.1.

3.2 Model training

FastBERT requires respective training steps for the backbone and the student classifiers. The parameters in one module is always frozen while the other module is being trained. The model is trained in preparation for downstream inference with three steps: the major backbone pre-training, entire backbone fine-tuning, and self-distillation for student classifiers.

3.2.1 Pre-training

The pre-training of backbone resembles that of BERT in the same way that our backbone resembles BERT. Any pre-training method used for BERT-like models (e.g., BERT-WWM (Cui et al., 2019), RoBERTa (Liu et al., 2019), and ERNIE

(Sun et al., 2019b)) can be directly applied. Note that the teacher classifier, as it is only used for inference, stays unaffected at this time. Also conveniently, FastBERT does not even need to perform pre-training by itself, for it can load high-quality pre-trained models freely.

3.2.2 Fine-tuning for backbone

For each downstream task, we plug in the task-specific data into the model, fine-tuning both the major backbone and the teacher classifier. The structure of the teacher classifier is as previously described. At this stage, all student classifiers are not enabled.

3.2.3 Self-distillation for branch

With the backbone well-trained for knowledge extraction, its output, as a high-quality soft-label containing both the original embedding and the generalized knowledge, is distilled for training student classifiers. As student are mutually independent, their predictions p_s are compared with the teacher soft-label p_t respectively, with the differences measured by KL-Divergence in (5),

$$D_{KL}(p_s, p_t) = \sum_{i=1}^{N} p_s(i) \cdot \log \frac{p_s(i)}{p_t(j)}. \quad (5)$$

As there are $L-1$ student classifiers in the FastBERT, the sum of their KL-Divergences is used as the total loss for self-distillation, which is formulated in (6),

$$Loss(p_{s_0}, ..., p_{s_{L-2}}, p_t) = \sum_{i=0}^{L-2} D_{KL}(p_{s_i}, p_t), \quad (6)$$

where p_{s_i} refers to the probability distribution of the output from *student-classifier i*.

Since this process only requires the teacher's output, we are free to use an unlimited number of unlabeled data, instead of being restricted to the labeled ones. This provides us with sufficient resources for self-distillation, which means we can always improve the student performance as long as the teacher allows. Moreover, our method differs from the previous distillation method, for the teacher and student outputs lie within the same model. This learning process does not require additional pre-training structures, making the distillation entirely a learning process by self.

3.3 Adaptive inference

With the above steps, FastBERT is well-prepared to perform inference in an adaptive manner, which means we can adjust the number of executed encoding layers within the model according to the sample complexity.

At each Transformer layer, we measure for each sample on whether the current inference is credible enough to be terminated.

Given an input sequence, the uncertainty of a student classifier's output p_s is computed with a normalized entropy in (7),

$$Uncertainty = \frac{\sum_{i=1}^{N} p_s(i) \log p_s(i)}{\log \frac{1}{N}}, \quad (7)$$

where p_s is the distribution of output probability, and N is the number of labeled classes.

With the definition of the uncertainty, we make an important hypothesis.

Hypothesis 1. *LUHA: the Lower the Uncertainty, the Higher the Accuracy.*

Definition 1. *Speed: The threshold to distinguish high and low uncertainty.*

LUHA is verified in Section 4.4. Both *Uncertainty* and *Speed* range between 0 and 1. The adaptive inference mechanism can be described as: At each layer of FastBERT, the corresponding *student classifier* will predict the label of each sample with measured *Uncertainty*. Samples with *Uncertainty* below the *Speed* will be sifted to early outputs, while samples with *Uncertainty* above the *Speed* will move on to the next layer.

Intuitively, with a higher *Speed*, fewer samples will be sent to higher layers, and overall inference speed will be faster, and vice versa. Therefore, *Speed* can be used as a halt value for weighing the inference accuracy and efficiency.

Table 1: FLOPs of each operation within the FastBERT (M = Million, N = the number of labels).

Operation	Sub-operation	FLOPs	Total
Transformer	Self-attention ($768 \to 768$)	603.0M	1809.9M
	Feedforward ($768 \to 3072 \to 768$)	1207.9M	
Classifier	Fully-connect ($768 \to 128$)	25.1M	46.1M
	Self-attention ($128 \to 128$)	16.8M	
	Fully-connect ($128 \to 128$)	4.2M	
	Fully-connect ($128 \to N$)	-	

Table 2: Comparison of accuracy (Acc.) and FLOPs (speedup) between FastBERT and Baselines in six Chinese datasets and six English datasets.

Dataset/ Model	ChnSentiCorp		Book review		Shopping review		LCQMC		Weibo		THUCNews	
	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)
BERT	95.25	21785M (1.00x)	86.88	21785M (1.00x)	96.84	21785M (1.00x)	86.68	21785M (1.00x)	97.69	21785M (1.00x)	96.71	21785M (1.00x)
DistilBERT (6 layers)	88.58	10918M (2.00x)	83.31	10918M (2.00x)	95.40	10918M (2.00x)	84.12	10918M (2.00x)	97.69	10918M (2.00x)	95.54	10918M (2.00x)
DistilBERT (3 layers)	87.33	5428M (4.01x)	81.17	5428M (4.01x)	94.84	5428M (4.01x)	84.07	5428M (4.01x)	97.58	5428M (4.01x)	95.14	5428M (4.01x)
DistilBERT (1 layers)	81.33	1858M (11.72x)	77.40	1858M (11.72x)	91.35	1858M (11.72x)	71.34	1858M (11.72x)	96.90	1858M (11.72x)	91.13	1858M (11.72x)
FastBERT (speed=0.1)	95.25	10741M (2.02x)	86.88	13613M (1.60x)	96.79	4885M (4.45x)	86.59	12930M (1.68x)	97.71	3691M (5.90x)	96.71	3595M (6.05x)
FastBERT (speed=0.5)	92.00	3191M (6.82x)	86.64	5170M (4.21x)	96.42	2517M (8.65x)	84.05	6352M (3.42x)	97.72	3341M (6.51x)	95.64	1979M (11.00x)
FastBERT (speed=0.8)	89.75	2315M (9.40x)	85.14	3012M (7.23x)	95.72	2087M (10.04x)	77.45	3310M (6.57x)	97.69	1982M (10.09x)	94.97	1854M (11.74x)

Dataset/ Model	Ag.news		Amz.F		Dbpedia		Yahoo		Yelp.F		Yelp.P	
	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)
BERT	94.47	21785M (1.00x)	65.50	21785M (1.00x)	99.31	21785M (1.00x)	77.36	21785M (1.00x)	65.93	21785M (1.00x)	96.04	21785M (1.00x)
DistilBERT (6 layers)	94.64	10872M (2.00x)	64.05	10872M (2.00x)	99.10	10872M (2.00x)	76.73	10872M (2.00x)	64.25	10872M (2.00x)	95.31	10872M (2.00x)
DistilBERT (3 layers)	93.98	5436M (4.00x)	63.84	5436M (4.00x)	99.05	5436M (4.00x)	76.56	5436M (4.00x)	63.50	5436M (4.00x)	93.23	5436M (4.00x)
DistilBERT (1 layers)	92.88	1816M (12.00x)	59.48	1816M (12.00x)	98.95	1816M (12.00x)	74.93	1816M (12.00x)	58.59	1816M (12.00x)	91.59	1816M (12.00x)
FastBERT (speed=0.1)	94.38	6013M (3.62x)	65.50	21005M (1.03x)	99.28	2060M (10.57x)	77.37	16172M (1.30x)	65.93	20659M (1.05x)	95.99	6668M (3.26x)
FastBERT (speed=0.5)	93.14	2108M (10.33x)	64.64	10047M (2.16x)	99.05	1854M (11.74x)	76.57	4852M (4.48x)	64.73	9827M (2.21x)	95.32	3456M (6.30x)
FastBERT (speed=0.8)	92.53	1858M (11.72x)	61.70	2356M (9.24x)	99.04	1853M (11.75x)	75.05	1965M (11.08x)	60.66	2602M (8.37x)	94.31	2460M (8.85x)

4 Experimental results

In this section, we will verify the effectiveness of FastBERT on twelve NLP datasets (six in English and six in Chinese) with detailed explanations.

4.1 FLOPs analysis

Floating-point operations (FLOPs) is a measure of the computational complexity of models, which indicates the number of floating-point operations that the model performs for a single process. The FLOPs has nothing to do with the model's operating environment (CPU, GPU or TPU) and only reveals the computational complexity. Generally speaking, the bigger the model's FLOPs is, the longer the inference time will be. With the same accuracy, models with low FLOPs are more efficient and more suitable for industrial uses.

We list the measured FLOPs of both structures in Table 1, from which we can infer that, **the calculation load (FLOPs) of the Classifier is much lighter than that of the Transformer**. This is the basis of the speed-up of FastBERT, for although it adds additional classifiers, it achieves acceleration by reducing more computation in Transformers.

4.2 Baseline and dataset

4.2.1 Baseline

In this section, we compare FastBERT against two baselines:

- **BERT**[1] The 12-layer BERT-base model was pre-trained on Wiki corpus and released by Google (Devlin et al., 2019).
- **DistilBERT**[2] The most famous distillation method of BERT with 6 layers was released by Huggingface (Sanh et al., 2019). In addition, we use the same method to distill the DistilBERT with 3 and 1 layer(s), respectively.

4.2.2 Dataset

To verify the effectiveness of FastBERT, especially in industrial scenarios, six Chinese and six English datasets pressing closer to actual applications are used. The six Chinese datasets include

[1] https://github.com/google-research/bert

[2] https://github.com/huggingface/transformers/tree/master/examples/distillation

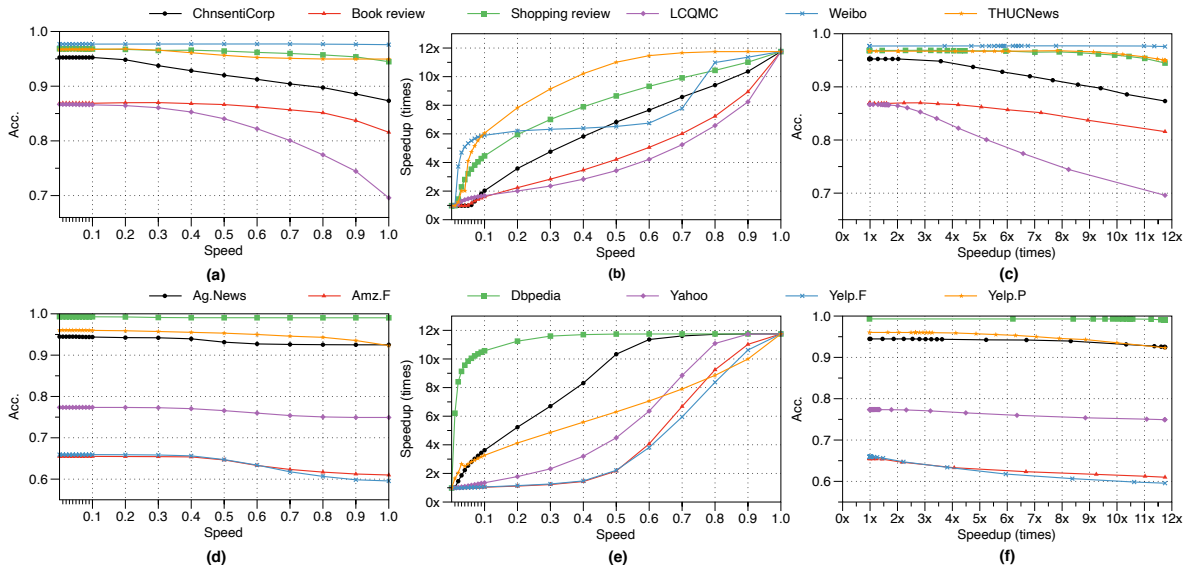

Figure 3: The trade-offs of FastBERT on twelve datasets (six in Chinese and six in English): **(a)** and **(d)** are *Speed-Accuracy* relations, showing changes of *Speed* (the threshold of *Uncertainty*) in dependence of the accuracy; **(b)** and **(e)** are *Speed-Speedup* relations, indicating that the *Speed* manages the adaptibility of FastBERT; **(c)** and **(f)** are the *Speedup-Accuracy* relations, i.e. the trade-off between efficiency and accuracy.

the sentence classification tasks (ChnSentiCorp, Book review(Qiu et al., 2018), Shopping review, Weibo and THUCNews) and a sentences-matching task (LCQMC(Liu et al., 2018)). All the Chinese datasets are available at the FastBERT project. The six English datasets (Ag.News, Amz.F, DBpedia, Yahoo, Yelp.F, and Yelp.P) are sentence classification tasks and were released in (Zhang et al., 2015).

4.3 Performance comparison

To perform a fair comparison, BERT / DistilBERT / FastBERT all adopt the same configuration as follows. In this paper, $L = 12$. The number of self-attention heads, the hidden dimension of embedding vectors, and the max length of the input sentence are set to 12, 768 and 128 respectively. Both FastBERT and BERT use pre-trained parameters provided by Google, while DistilBERT is pre-trained with (Sanh et al., 2019). We fine-tune these models using the AdamW (Loshchilov and Hutter) algorithm, a 2×10^{-5} learning rate, and a 0.1 warmup. Then, we select the model with the best accuracy in 3 epochs. For the self-distillation of FastBERT, we increase the learning rate to 2×10^{-4} and distill it for 5 epochs.

We evaluate the text inference capabilities of these models on the twelve datasets and report their accuracy (Acc.) and sample-averaged FLOPs under different *Speed* values. The result of comparisons are shown in Table 2, where the *Speedup* is obtained by using BERT as the benchmark. It can be observed that with the setting of $Speed = 0.1$, FastBERT can speed up 2 to 5 times without losing accuracy for most datasets. If a little loss of accuracy is tolerated, FastBERT can be 7 to 11 times faster than BERT. Comparing to DistilBERT, FastBERT trades less accuracy to catch higher efficiency. Figure 3 illustrates FastBERT's tradeoff in accuracy and efficiency. The speedup ratio of FastBERT are free to be adjusted between 1 and 12, while the loss of accuracy remains small, which is a very attractive feature in the industry.

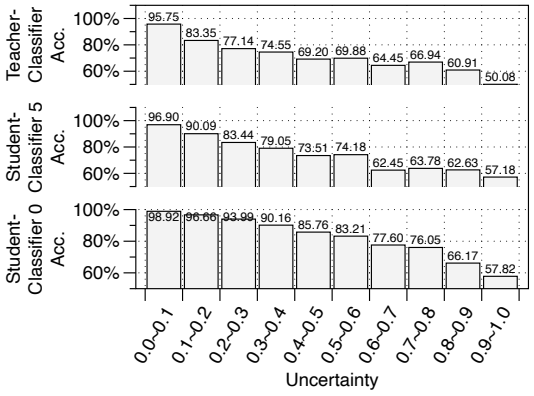

Figure 4: The relation of classifier accuracy and average case uncertainty: Three classifiers at the bottom, in the middle, and on top of the FastBERT were analyzed, and their accuracy within various uncertainty intervals were calculated with the Book Review dataset.

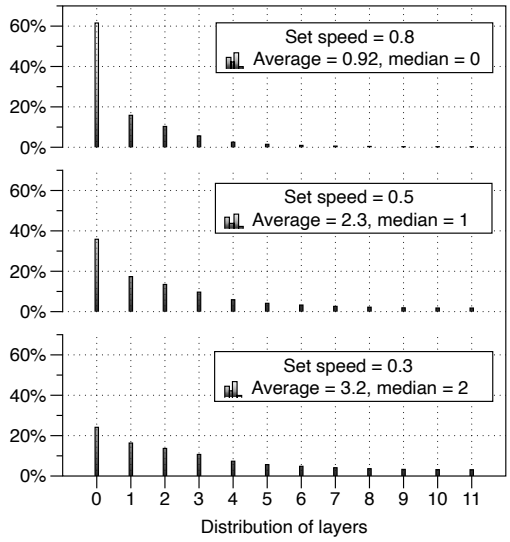

Figure 5: The distribution of executed layers on average in the Book review dataset, with experiments at three different speeds (0.3, 0.5 and 0.8).

4.4 LUHA hypothesis verification

As is described in the Section 3.3, the adaptive inference of FastBERT is based on the LUHA hypothesis, i.e., "*the Lower the Uncertainty, the Higher the Accuracy*". Here, we prove this hypothesis using the book review dataset. We intercept the classification results of *Student-Classifier0*, *Student-Classifier5*, and *Teacher-Classifier* in FastBERT, then count their accuracy in each uncertainty interval statistically. As shown in Figure 4, the statistical indexes confirm that the classifier follows the LUHA hypothesis, no matter it sits at the bottom, in the middle or on top of the model.

From Figure 4, it is easy to mistakenly conclude that *Student*s has better performance than *Teacher* due to the fact that the accuracy of *Student* in each uncertainty range is higher than that of *Teacher*. This conclusion can be denied by analysis with Figure 6(a) together. For the *Teacher*, more samples are located in areas with lower uncertainty, while the *Student*s' samples are nearly uniformly distributed. Therefore the overall accuracy of the *Teacher* is still higher than that of *Student*s.

4.5 In-depth study

In this section, we conduct a set of in-depth analysis of FastBERT from three aspects: the distribution of exit layer, the distribution of sample uncertainty, and the convergence during self-distillation.

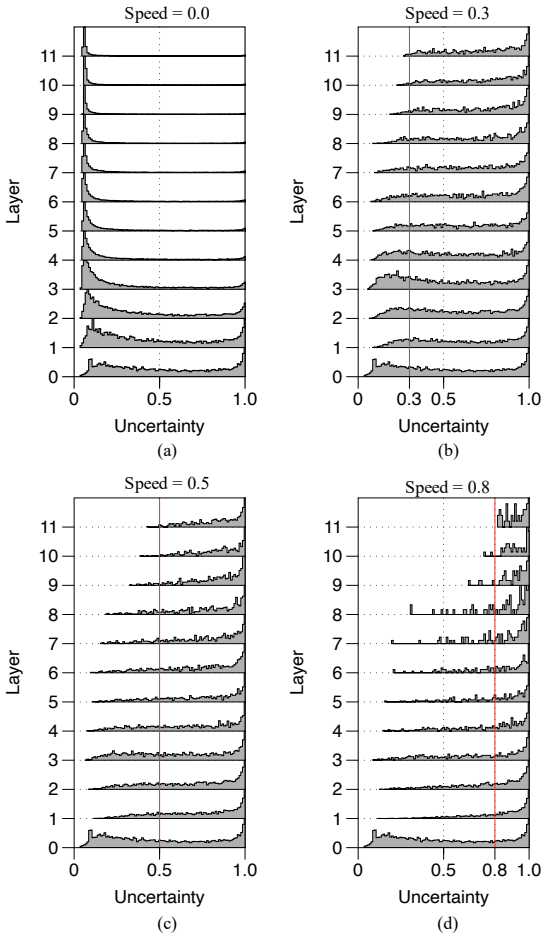

Figure 6: The distribution of *Uncertainty* at different layers of FastBERT in the Book review dataset: **(a)** The *speed* is set to 0.0, which means that all samples will pass through all the twelve layers; **(b)** $\sim$ **(d)**: The *Speed* is set to 0.3, 0.5, and 0.8 respectively, iand only the samples with *Uncertainty* higher than *Speed* will be sent to the next layer.

4.5.1 Layer distribution

In FastBERT, each sample walks through a different number of Transformer layers due to varied complexity. For a certain condition, fewer executed layers often requires less computing resources. As illustrated in Figure 5, we investigate the distribution of exit layers under different constraint of *Speed*s (0.3, 0.5 and 0.8) in the book review dataset. Take *Speed* $= 0.8$ as an example, at the first layer *Transformer0*, 61% of the samples is able to complete the inference. This significantly eliminates unnecessary calculations in the next eleven layers.

4.5.2 Uncertainty distribution

The distribution of sample uncertainty predicted by different student classifiers varies, as is illustrated in Figure 6. Observing these distributions help us to

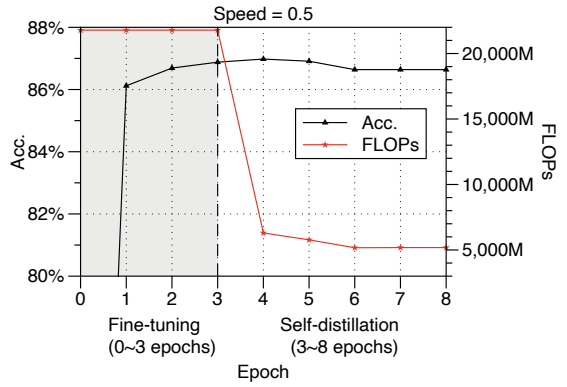

Figure 7: The change in accuracy and FLOPs of FastBERT during fine-tuning and self-distillation with the Book review dataset. The accuracy firstly increases at the fine-tuning stage, while the self-distillation reduces the FLOPs by six times with almost no loss in accuracy.

further understand FastBERT. From Figure 6**(a)**, it can be concluded that the higher the layer is posited, the lower the uncertainty with given *Speed* will be, indicating that the high-layer classifiers more decisive than the lower ones. It is worth noting that at higher layers, there are samples with uncertainty below the threshold of *Uncertainty* (i.e., the *Speed*), for these high-layer classifiers may reverse the previous judgments made by the low-layer classifiers.

4.5.3 Convergence of self-distillation

Self-distillation is a crucial step to enable FastBERT. This process grants student classifiers with the abilities to infer, thereby offloading work from the teacher classifier. Taking the Book Review dataset as an example, we fine-tune the FastBERT with three epochs then self-distill it for five more epochs. Figure 7 illustrates its convergence in accuracy and FLOPs during fine-tune and self-distillation. It could be observed that the accuracy increases with fine-tuning, while the FLOPs decrease during the self-distillation stage.

4.6 Ablation study

Adaptation and self-distillation are two crucial mechanisms in FastBERT. We have preformed ablation studies to investigate the effects brought by these two mechanisms using the Book Review dataset and the Yelp.P dataset. The results are presented in Table 3, in which 'without self-distillation' implies that all classifiers, including both the teacher and the students, are trained in the fine-tuning; while 'without adaptive inference' means that the number of executed layers of each sample is fixated to two or six.

Table 3: Results of ablation studies on the Book review and Yelp.P datasets.

Config.	**Book review**		**Yelp.P**	
	Acc.	FLOPs (speedup)	Acc.	FLOPs (speedup)
FastBERT				
speed=0.2	86.98	9725M (2.23x)	95.90	52783M (4.12x)
speed=0.7	85.69	3621M (6.01x)	94.67	2757M (7.90x)
FastBERT without self-distillation				
speed=0.2	86.22	9921M (2.19x)	95.55	4173M (5.22x)
speed=0.7	85.02	4282M (5.08x)	94.54	2371M (9.18x)
FastBERT without adaptive inference				
layer=6	86.42	11123M (1.95x)	95.18	11123M (1.95x)
layer=2	82.88	3707M (5.87x)	93.11	3707M (5.87x)

From Table 3, we have observed that: **(1)** At almost the same level of speedup, FastBERT without self-distillation or adaption performs poorer; **(2)** When the model is accelerated more than five times, downstream accuracy degrades dramatically without adaption. It is safe to conclude that both the adaptation and self-distillation play a key role in FastBERT, which achieves both significant speedups and favorable low losses of accuracy.

5 Conclusion

In this paper, we propose a fast version of BERT, namely FastBERT. Specifically, FastBERT adopts a self-distillation mechanism during the training phase and an adaptive mechanism in the inference phase, achieving the goal of gaining more efficiency with less accuracy loss. Self-distillation and adaptive inference are first introduced to NLP model in this paper. In addition, FastBERT has a very practical feature in industrial scenarios, i.e., its inference speed is tunable.

Our experiments demonstrate promising results on twelve NLP datasets. Empirical results have shown that FastBERT can be 2 to 3 times faster than BERT without performance degradation. If we slack the tolerated loss in accuracy, the model is free to tune its speedup between 1 and 12 times. Besides, FastBERT remains compatible to the parameter settings of other BERT-like models (e.g., BERT-WWM, ERNIE, and RoBERTa), which means these public available models can be readily loaded

for FastBERT initialization.

6 Future work

These promising results point to future works in **(1)** linearizing the *Speed-Speedup* curve; **(2)** extending this approach to other pre-training architectures such as XLNet (Yang et al., 2019) and ELMo (Peters et al., 2018); **(3)** applying FastBERT on a wider range of NLP tasks, such as named entity recognition and machine translation.

Acknowledgments

This work is funded by 2019 Tencent Rhino-Bird Elite Training Program. Work done while this author was an intern at Tencent.

References

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese BERT. *arXiv preprint arXiv:1906.08101*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of ACL*, pages 4171–4186.

Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. 2017. Spatially adaptive computation time for residual networks. In *Proceedings of CVPR*, pages 1790–1799.

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

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

Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In *Advances in NeurIPS*, pages 1135–1143.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *Computer Science*, 14(7):38–39.

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

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*, pages 1746–1751.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of EMNLP-IJCNLP*, pages 4356–4365.

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

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: Enabling language representation with knowledge graph. In *Proceedings of AAAI*.

Xin Liu, Qingcai Chen, Chong Deng, Huajun Zeng, Jing Chen, Dongfang Li, and Buzhou Tang. 2018. Lcqmc: A large-scale chinese question matching corpus. In *Proceedings of the ICCL*, pages 1952–1962.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Yuanyuan Qiu, Hongzheng Li, Shen Li, Yingdi Jiang, Renfen Hu, and Lijiao Yang. 2018. Revisiting correlations between intrinsic and extrinsic evaluations of word embeddings. In *Proceedings of CCL*, pages 209–221. Springer.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *Technical report, OpenAI*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC2 Workshop*.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019a. Patient knowledge distillation for bert model compression. In *Proceedings of EMNLP-IJCNLP*, pages 4314–4323.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019b. ERNIE: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *Proceedings of ICPR*, pages 2464–2469.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in NeurIPS*, pages 5998–6008.

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 EMNLP*, pages 353–355.

Baoxin Wang. 2018. Disconnected recurrent neural networks for text categorization. In *Proceedings of ACL*, pages 2311–2320.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in NeurIPS*, pages 649–657.

Zhe Zhao, Hui Chen, Jinbin Zhang, Xin Zhao, Tao Liu, Wei Lu, Xi Chen, Haotang Deng, Qi Ju, and Xiaoyong Du. 2019. UER: An open-source toolkit for pre-training models. In *Proceedings of EMNLP-IJCNLP 2019*, page 241.