Length-Adaptive Transformer: Train Once with Length Drop, Use Anytime with Search

Gyuwan Kim
Clova AI, NAVER Corp.
gyuwan.kim@navercorp.com

Kyunghyun Cho
New York University
kyunghyun.cho@nyu.edu

Abstract

Despite transformers' impressive accuracy, their computational cost is often prohibitive to use with limited computational resources. Most previous approaches to improve inference efficiency require a separate model for each possible computational budget. In this paper, we extend PoWER-BERT (Goyal et al., 2020) and propose *Length-Adaptive Transformer* that can be used for various inference scenarios after one-shot training. We train a transformer with *LengthDrop*, a structural variant of dropout, which stochastically determines a sequence length at each layer. We then conduct a multi-objective evolutionary search to find a length configuration that maximizes the accuracy and minimizes the efficiency metric under any given computational budget. Additionally, we significantly extend the applicability of PoWER-BERT beyond sequence-level classification into token-level classification with *Drop-and-Restore* process that drops word-vectors temporarily in intermediate layers and restores at the last layer if necessary. We empirically verify the utility of the proposed approach by demonstrating the superior accuracy-efficiency trade-off under various setups, including span-based question answering and text classification. Code is available at https://github.com/clovaai/length-adaptive-transformer.

1 Introduction

Pre-trained language models (Peters et al., 2018; Devlin et al., 2018; Radford et al., 2019; Yang et al., 2019; He et al., 2020) have achieved notable improvements in various natural language processing (NLP) tasks. Most of them rely on transformers (Vaswani et al., 2017), and the number of model parameters ranges from hundreds of millions to billions (Shoeybi et al., 2019; Raffel et al., 2019; Kaplan et al., 2020; Brown et al., 2020). Despite this high accuracy, excessive computational overhead during inference, both in terms of time and memory, has hindered its use in real applications. This level of excessive computation has further raised the concern over energy consumption as well (Schwartz et al., 2019; Strubell et al., 2019; Cao et al., 2020).

Recent studies have attempted to address these concerns regarding large-scale transformers' computational and energy efficiency (see §7 for a more extensive discussion.) Among these, we focus on PoWER-BERT (Goyal et al., 2020) which progressively reduces sequence length by eliminating word-vectors based on the attention values as passing layers. PoWER-BERT establishes the superiority of accuracy-time trade-off over earlier approaches (Sanh et al., 2019; Sun et al., 2019; Michel et al., 2019). However, it requires us to train a separate model for each efficiency constraint. In this paper, we thus develop a framework based on PoWER-BERT such that we can train a single model that can be adapted in the inference time to meet any given efficiency target.

In order to train a transformer to cope with a diverse set of computational budgets in the inference time, we propose to train once while reducing the sequence length with a random proportion at each layer. We refer to this procedure as LengthDrop, which was motivated by the nested dropout (Rippel et al., 2014). We can extract sub-models of shared weights with any length configuration without requiring extra post-processing nor additional fine-tuning.

It is not trivial to find an optimal length configuration given the inference-time computational budget, although it is extremely important in order to deploy these large-scale transformers in practice. Once a transformer is trained with the proposed LengthDrop, we search for the length configuration that maximizes the accuracy given a computational budget. Because this search is combinatorial and has multiple objectives (accuracy and efficiency),

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 6501–6511
August 1–6, 2021.

in this work, we propose to use an evolutionary search algorithm, which further allows us to obtain a full Pareto frontier of accuracy-efficiency trade-off of each model.

PoWER-BERT, which forms the foundation of the proposed two-stage procedure, is only applicable to sequence-level classification, because it eliminates some of the word vectors at each layer by design. In other words, it cannot be used for token-level tasks such as span-based question answering (Rajpurkar et al., 2016) because these tasks require hidden representations of the entire input sequence at the final layer. We thus propose to extend PoWER-BERT with a novel Drop-and-Restore process (§3.3), which eliminates this inherent limitation. Word vectors are dropped and set aside, rather than eliminated, in intermediate layers to maintain the saving of computational cost, as was with the original PoWER-BERT. These set-aside vectors are then restored at the final hidden layer and provided as an input to a subsequent task-specific layer, unlike the original PoWER-BERT.

The main contributions of this work are two-fold. First, we introduce LengthDrop, a structured variant of dropout for training a single Length-Adaptive Transformer model that allows us to automatically derive multiple sub-models with different length configurations in the inference time using evolutionary search, without requiring any re-training. Second, we design Drop-and-Restore process that makes PoWER-BERT applicable beyond classification, which enables PoWER-BERT to be applicable to a wider range of NLP tasks such as span-based question answering. We empirically verify Length-Adaptive Transformer works quite well using the variants of BERT on a diverse set of NLP tasks, including SQuAD 1.1 (Rajpurkar et al., 2016) and two sequence-level classification tasks in GLUE benchmark (Wang et al., 2018). Our experiments reveal that the proposed approach grants us fine-grained control of computational efficiency and a superior accuracy-efficiency trade-off in the inference time compared to existing approaches.

2 Background

In this section, we review some of the building blocks of our main approach. In particular, we review transformers, which are a standard backbone used in natural language processing these days, and PoWER-BERT, which was recently proposed as an effective way to train a large-scale, but highly efficient transformer for sequence-level classification.

2.1 Transformers and BERT

A transformer is a particular neural network that has been designed to work with a variable-length sequence input and is implemented as a stack of self-attention and fully connected layers (Vaswani et al., 2017). It has recently become one of the most widely used models for natural language processing. Here, we give a brief overview of the transformer which is the basic building block of the proposed approach.

Each token x_t in a sequence of tokens $x = (x_1, \ldots, x_N)$, representing input text, is first turned into a continuous vector $h_t^0 \in R^H$ which is the sum of the token and position embedding vectors. This sequence is fed into the first transformer layer which returns another sequence of the same length $h^1 \in R^{N \times H}$. We repeat this procedure L times, for a transformer with L layers, to obtain $h^L = (h_1^L, \ldots, h_N^L)$. We refer to each vector in the hidden sequence at each layer as a *word vector* to emphasize that there exists a correspondence between each such vector and one of the input words.

Although the transformer was first introduced for the problem of machine translation, Devlin et al. (2018) demonstrated that the transformer can be trained and used as a sentence encoder. More specifically, Devlin et al. (2018) showed that the transformer-based masked language model, called BERT, learns a universally useful parameter set that can be fine-tuned for any downstream task, including sequence-level and token-level classification.

In the case of sequence-level classification, a softmax classifier is attached to the word vector h_1^L associated with the special token `[CLS]`, and the entire network, including the softmax classifier and BERT, is fine-tuned. For token-level classification, we use each h_t^L as the final hidden representation of the associated t-th word in the input sequence. This strategy of pre-training followed by fine-tuning, often referred to as transfer learning, is a dominant approach to classification in natural language processing.

2.2 PoWER-BERT

PoWER-BERT keeps only the topmost l_j word vectors at each layer j by eliminating redundant ones based on the significance score which is the total amount of attention imposed by a word on the other words (Goyal et al., 2020). l_j is the hyper-parameter that determines how many vectors to

keep at layer j. PoWER-BERT has the same model parameters as BERT, but the extraction layers are interspersed after the self-attention layer in every transformer block (Vaswani et al., 2017).

PoWER-BERT reduces inference time successfully, achieving better accuracy-time trade-off than DistilBERT (Sanh et al., 2019), BERT-PKD (Sun et al., 2019), and Head-Prune (Michel et al., 2019). Despite the original intention of maximizing the inference efficiency with the minimal loss in accuracy, it is possible to set up PoWER-BERT to be both more efficient and more accurate compared to the original BERT, which was observed but largely overlooked by Goyal et al. (2020).

Training a PoWER-BERT model consists of three steps: (1) fine-tuning, (2) length configuration search, and (3) re-training. The fine-tuning step is just like the standard fine-tuning step of BERT given a target task. A length configuration is a sequence of retention parameters $(l_1, \cdots l_L)$, each of which corresponds to the number of word vectors that are kept at each layer. These retention parameters are learned along with all the other parameters to minimize the original task loss together with an extra term that approximately measures the number of retained word vectors across layers. In the re-training step, PoWER-BERT is fine-tuned with the length configuration fixed to its learned one.

For each computational budget, we must train a separate model going through all three steps described above. Moreover, the length configuration search step above is only approximate, as it relies on the relaxation of retention parameters which are inherently discrete. This leads to the lack of guaranteed correlation between the success of this stage and true run-time. Even worse, it is a delicate act to tune the length configuration given a target computational budget because the trade-off is *implicitly* made via a regularization coefficient. Furthermore, PoWER-BERT has an inherent limitation in that it only applies to sequence-level classification because it eliminates word vectors in intermediate layers.

3 Length-Adaptive Transformer

In this section, we explain our proposed framework which results in a transformer that reduces the length of a sequence at each layer with an arbitrary rate. We call such a resulting transformer a Length-Adaptive Transformer. We train Length-Adaptive Transformer with LengthDrop which randomly samples the number of hidden vectors to be dropped at each layer with the goal of making the final model robust to such drop in the inference time. Once the model is trained, we search for the optimal trade-off between accuracy and efficiency using multi-objective evolutionary search, which allows us to use the model for any given computational budget without fine-tuning nor re-training. At the end of this section, we describe Drop-and-Restore process as a way to greatly increase the applicability of PoWER-BERT which forms a building block of the proposed framework.

In short, we train a Length-Adaptive Transformer once with LengthDrop and Drop-and-Restore, and use it with an automatically determined length configuration for inference with any target computational budget, on both sequence-level and token-level tasks.

3.1 LengthDrop

Earlier approaches to efficient inference with transformers have focused on a scenario where the target computational budget for inference is known in advance (Sanh et al., 2019; Goyal et al., 2020). This greatly increases the cost of deploying transformers, as it requires us to train a separate transformer for each scenario. Instead, we propose to train one model that could be used for a diverse set of target computational budgets without re-training.

Before each SGD update, LengthDrop randomly generates a length configuration by sequentially sampling a sequence length l_{i+1} at the $(i+1)$-th layer based on the previous layer's sequence length l_i, following the uniform distribution $\mathcal{U}((1-p)l_i, l_i)$, where l_0 is set to the length of the input sequence, and p is the LengthDrop probability. This sequential sampling results in a length configuration $(l_1, \cdots, l_L)$. Length-Adaptive Transformer can be thought of as consisting of a full model and many sub-models corresponding to different length configurations, similarly to a neural network trained with different dropout masks (Srivastava et al., 2014).

LayerDrop From the perspective of each word vector, the proposed LengthDrop could be thought of as skipping the layers between when it was set aside and the final layer where it was restored. The word vector however does not have any information based on which it can determine whether it would be dropped at any particular layer. In our

preliminary experiments, we found that this greatly hinders optimization. We address this issue by using LayerDrop (Fan et al., 2019) which skips each layer of a transformer uniformly at random. The LayerDrop encourages each word vector to be agnostic to skipping any number of layers between when it is dropped and when it is restored, just like dropout (Srivastava et al., 2014) prevents hidden neurons from co-adapting with each other by randomly dropping them.

Sandwich Rule and Inplace Distillation We observed that standard supervised training with LengthDrop does not work well in the preliminary experiments. We instead borrow a pair of training techniques developed by Yu and Huang (2019) which are sandwich rule and inplace distillation, for better optimization as well as final generalization. At each update, we update the full model without LengthDrop as usual to minimize the supervised loss function. We simultaneously update n_s randomly-sampled sub-models (which are called sandwiches) and the smallest-possible sub-model, which corresponds to keeping only $(1-p)l_i$ word vectors at each layer i, using knowledge distillation (Hinton et al., 2015) from the full model. Here, sub-models mean models with length reduction. They are trained to their prediction close to the full model's prediction (inplace distillation).

3.2 Evolutionary Search of Length Configurations

After training a Length-Adaptive Transformer with LengthDrop, we search for appropriate length configurations for possible target computational budgets that will be given at inference time. The length configuration determines the model performance in terms of both accuracy and efficiency. In order to search for the optimal length configuration, we propose to use evolutionary search, similarly to Cai et al. (2019) and Wang et al. (2020a). This procedure is efficient, as it only requires a single pass through the relatively small validation set for each length configuration, unlike re-training for a new computational budget which requires multiple passes through a significantly larger training set for each budget.

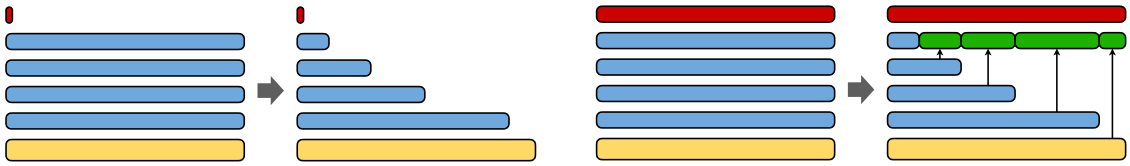
We initialize the population with constant-ratio configurations. Each configuration is created by $l_{i+1} = (1-r)l_i$ for each layer i with r so that the amount of computation within the initial population is uniformly distributed between those of the smallest and full models. At each iteration, we evolve the population to consist only of configurations lie on a newly updated efficiency-accuracy Pareto frontier by mutation and crossover. Mutation alters an original length configuration $(l_1, \cdots, l_L)$ to $(l'_1, \cdots, l'_L)$ by sampling l'_i from the uniform distribution $\mathcal{U}(l'_{i-1}, l_{i+1})$ with the probability p_m or keeping the original length $l'_i = l_i$, sweeping the layers from $i = 1$ to $i = L$. A crossover takes two length configurations and averages the lengths at each layer. Both of these operations are performed while ensuring the monotonicity of the lengths over the layers. We repeat this iteration G times while maintaining n_m mutated configurations and n_c crossover'd configurations. Repeating this procedure pushes the Pareto frontier further to identify the best trade-off between two objectives, efficiency and accuracy, without requiring any continuous relaxation of length configurations nor using a proxy objective function.

3.3 Drop-and-Restore Process

The applicability of the PoWER-BERT, based on which our main contribution above was made, is limited to sequence-level classification because it eliminates word vectors at each layer. In addition to our main contribution above, we thus propose to extend the PoWER-BERT so that it is applicable to token-level classification, such as span-based question-answering. Our proposal, to which we refer as Drop-and-Restore, does not eliminate word vectors at each layer according to the length configuration but instead sets them aside until the final hidden layer. At the final hidden layer, these word vectors are brought back to form the full hidden sequence, as illustrated graphically in Figure 1.

4 Experiment Setup

Datasets We test the proposed approach on both sequence-level and token-level tasks, the latter of which could not have been done with the original PoWER-BERT unless for the proposed Drop-and-Restore. We use MNLI-m and SST-2 from GLUE benchmark (Wang et al., 2018), as was done to test PoWER-BERT earlier, for sequence-level classification. We choose them because consistent accuracy scores from standard training on them due to their sufficiently large training set imply that they are reliable to verify our approach. We use SQuAD 1.1 (Rajpurkar et al., 2016) for token-level classification.



(a) *PoWER* (b) *Drop-and-Restore*

Figure 1: Illustration of (a) word-vector elimination process in PoWER-BERT (Goyal et al., 2020) and (b) Drop-and-Restore process in Length-Adaptive Transformer. Yellow box and blue boxes imply the output of embedding layer and transformer layers, respectively. Green boxes mean vectors dropped in lower layers and restored at the last layer. Red box is the task-specific layer. Though word-vectors in the middle could be eliminated (or dropped), remaining vectors are left-aligned for the better illustration. In this case, the number of transformer layers is four.

Evaluation metrics We use the number of floating operations (FLOPs) as a main metric to measure the inference efficiency given any length configuration, as it is agnostic to the choice of the underlying hardware, unlike other alternatives such as hardware-aware latency (Wang et al., 2020a) or energy consumption (Henderson et al., 2020). We later demonstrate that FLOPs and wall-clock time on GPU and CPU correlate well with the proposed approach, which is not necessarily the case for other approaches, such as unstructured weight pruning (Han et al., 2015; See et al., 2016).

Pre-trained transformers Since BERT was introduced by Devlin et al. (2018), it has become a standard practice to start from a pre-trained (masked) language model and fine-tune it for each downstream task. We follow the same strategy in this paper and test two pre-trained transformer-based language models; $\text{BERT}_{\text{Base}}$ (Devlin et al., 2018) and DistilBERT (Sanh et al., 2019), which allows us to demonstrate that the usefulness and applicability of our approach are not tied to any specific architectural choice, such as the number of layers and the maximum input sequence length. Although we focus on BERT-based masked language models here, the proposed approach is readily applicable to any transformer-based models.

Learning We train a Length-Adaptive Transformer with LengthDrop probability and LayerDrop probability both set to 0.2. We use $n_s = 2$ randomly sampled intermediate sub-models in addition to the full model and smallest model for applying the sandwich learning rule.

We start fine-tuning the pre-trained transformer without Drop-and-Restore first, just as Goyal et al. (2020) did with PoWER-BERT. We then continue fine-tuning it for another five epochs *with* Drop-and-Restore. This is unlike the recommended three epochs by Devlin et al. (2018), as learning progresses slower due to a higher level of stochasticity introduced by LengthDrop and LayerDrop. We use the batch size of 32, the learning rate of $5e-5$ for SQuAD 1.1 and $2e-5$ for MNLI-m and SST, and the maximum sequence length of 384 for SQuAD 1.1 and 128 for MNLI-m and SST.

Search We run up to $G = 30$ iterations of evolutionary search, using $n_m = 30$ mutated configurations with mutation probability $p_m = 0.5$ and $n_c = 30$ crossover'd configurations, to find the Pareto frontier of accuracy and efficiency.

5 Results and Analysis

Efficiency-accuracy trade-off We use SQuAD 1.1 to examine the effect of the proposed approach on the efficiency-accuracy trade-off. When the underlying classifier was not trained with LengthDrop, as proposed in this paper, the accuracy drops even more dramatically as more word vectors are dropped at each layer. The difference between standard transformer and Length-Adaptive Transformer is stark in Figure 2. This verifies the importance of training a transformer in a way that makes it malleable for inference-time re-configuration.

When the model was trained with the proposed LengthDrop, we notice the efficacy of the proposed approach of using evolutionary search to find the optimal trade-off between inference efficiency and accuracy. The trade-off curve from the proposed search strategy has a larger area-under-curve (AUC) than when constant-rate length reduction was used to meet a target computational budget. It demonstrates the importance of using both LengthDrop and evolutionary search.

We make a minor observation that the proposed approach ends up with a significantly higher accuracy than DistillBERT when enough computational

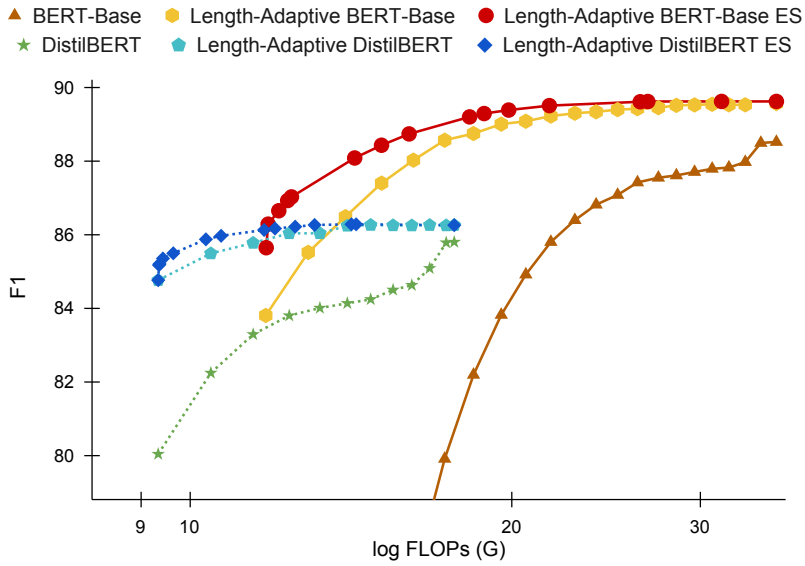
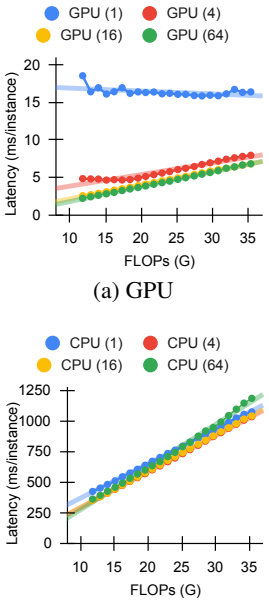


Figure 2: Pareto curves of F1 score and FLOPs on SQuAD 1.1 (Rajpurkar et al., 2016). We apply the proposed method to $\text{BERT}_{\text{Base}}$ (solid lines) and DistilBERT (dotted lines). For each model, we draw three curves using (1) standard fine-tuned transformer with constant-rate length reduction, (2) Length-Adaptive Transformer with constant-rate length reduction, and (3) Length-Adaptive Transformer with length configurations obtained from the evolutionary search.



(a) GPU

(b) CPU

Figure 3: Correlation between FLOPs and latency with different length configurations.

budget is allowed for inference ($\log \text{FLOPs} > 10$). This makes our approach desirable in a wide array of scenarios, as it does not require any additional pre-training stage, as does DistilBERT. With a severe constraint on the computational budget, the proposed approach could be used on DistilBERT to significantly improve the efficiency without compromising the accuracy.

Maximizing inference efficiency We consider all three tasks, SQuAD 1.1, MNLI-m, and SST-2, and investigate how much efficiency can be gained by the proposed approach with minimal sacrifice of accuracy. First, we look at how much efficiency could be gained without losing accuracy. That is, we use the length configuration that maximizes the inference efficiency (i.e., minimize the FLOPs) while ensuring that the accuracy is above or the same as the accuracy of the standard approach without any drop of word vectors. The results are presented in the rows marked with Length-Adaptive† from Table 1. For example, in the case of $\text{BERT}_{\text{Base}}$, the proposed approach reduces FLOPs by more than half across all three tasks.

From Figure 2, we have observed that the proposed Length-Adaptive Transformer generalizes better than the standard, base model in some cases. Thus, we try to maximize both the inference efficiency and accuracy in order to see whether it is possible for the proposed algorithm to find a length configuration that both maximizes inference efficiency and improves accuracy. We present the results in the rows marked with Length-Adaptive* from Table 1. For all cases, Length-Adaptive Transformer achieves higher accuracy than a standard transformer does while reducing FLOPs significantly. Although it is not apparent from the table, tor MNLI-m and SST-2, the accuracy of the smallest sub-model is already greater than or equal to that of a standard transformer.

FLOPs vs. Latency As has been discussed in recent literature (see, e.g., Li et al. (2020); Chin et al. (2020)), the number of FLOPs is not a perfect indicator of the real latency measured in wall-clock time, as the latter is affected by the combination of hardware choice and network architecture. To understand the real-world impact of the proposed approach, we study the relationship between FLOPs, obtained by the proposed procedure, and wall-clock time measured on both CPU and GPU by measuring them while varying length configurations. As shown in Figure 3, FLOPs and latency exhibit near-linear correlation on GPU, when the minibatch size is ≥ 16, and regardless of the minibatch size, on CPU. In other words, the reduction in FLOPs with

Model		SQuAD 1.1		MNLI-m		SST-2	
Pretrained Transformer	Method	F1	FLOPs	Acc	FLOPs	Acc	FLOPs
$\text{BERT}_{\text{Base}}$	Standard	88.5	1.00x	84.4	1.00x	92.8	1.00x
	Length-Adaptive*	89.6	0.89x	85.0	0.58x	93.1	0.36x
	Length-Adaptive†	88.7	0.45x	84.4	0.35x	92.8	0.35x
DistilBERT	Standard	85.8	1.00x	80.9	1.00x	90.6	1.00x
	Length-Adaptive*	86.3	0.81x	81.5	0.56x	92.0	0.55x
	Length-Adaptive†	85.9	0.59x	81.3	0.54x	91.7	0.54x

Table 1: Comparison results of standard Transformer and Length-Adaptive Transformer. Among length configurations on the Pareto frontier of Length-Adaptive Transformer, we pick two representative points: Length-Adaptive* and Length-Adaptive† as the most efficient one while having the highest accuracy and the accuracy higher than (or equal to) standard Transformer, respectively.

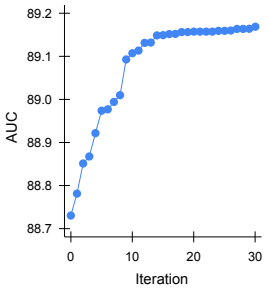


Figure 4: Example of area under Pareto curve as the evolutionary search of length configurations proceeds.

the proposed approach directly implies the reduction in wall-clock time.

Convergence of search Although the proposed approach is efficient in that it requires only one round of training, it needs a separate search stage for each target budget. It is important for evolutionary search to converge quickly in the number of forward sweeps of a validation set. As exemplified in Figure 4, evolutionary search converges after about fifteen iterations.

6 Comparison with Other Works

Our framework allows a novel method for anytime prediction with adaptive sequence length given any transformers. Thus, our goal is not state-of-the-art classification accuracy, although our experimental results (§5) demonstrate that our method still attains a good accuracy level.

We emphasize that other adaptive computation works (§7) are orthogonal with ours, meaning that various adaptive dimensions (sequence length, depth, attention head, hidden dimension, etc.) can be jointly used. In other words, even if other adaptive methods show better curves than ours, our method and theirs can boost each other when combined. We provide some comparison results with PoWER-BERT (not anytime prediction method) and DynaBERT (Hou et al., 2020) (concurrent adaptive computation method) as follows.

Comparison with PoWER-BERT According to Goyal et al. (2020), PoWER-BERT achieves 2.6x speedup for MNLI-m and 2.4x speedup for SST-2 by losing 1% of their accuracy. Length-Adaptive Transformer obtains a 2.9x speedup in terms of FLOPs without losing accuracy on MNLI-m and SST-2. Considering Figure 3, our speedup in execution time would be close to 2.9x in the same setting of PoWER-BERT where the time measurement is done with a batch size of 128 on GPU. It indicates that our model offers a better trade-off than PoWER-BERT, even with a single model.

Comparison with DynaBERT According to Hou et al. (2020), DyanBERT obtains a gain of $+1.0$, $+0.1$, $+0.4$ for the best accuracy in SQuAD 1.1, MNLI-m, and SST-2, respectively, while Length-Adaptive Transformer achieves a gain of $+1.1$, $+0.6$, $+0.3$. These results imply that Length-Adaptive Transformer can give a comparable (or better) performance with DynaBERT.

7 Related Work

The main purpose of the proposed algorithm is to improve the inference efficiency of a large-scale transformer. This goal has been pursued from various directions, and here we provide a brief overview of these earlier and some concurrent attempts in the context of the proposed approach.

Weight pruning Weight pruning (Han et al., 2015) focuses on reducing the number of parameters that directly reflects the memory footprint of a model and indirectly correlates with inference speed. However, their actual speed-up in runtime is usually not significant, especially while executing a model with parallel computation using GPU devices (Tang et al., 2018; Li et al., 2020).

Adaptive architecture There are three major axes along which computation can be reduced in a neural network; (1) input size/length, (2) network depth, and (3) network width. The proposed approach, based on PoWER-BERT, adaptively reduces the input length as the input sequence is pro-

cessed by the transformer layers. In our knowledge, Goyal et al. (2020) is the first work in this direction for transformers. Funnel-Transformer (Dai et al., 2020) and multi-scale transformer language models (Subramanian et al., 2020) also successfully reduce sequence length in the middle and rescale to full length for the final computation. However, their inference complexity is fixed differently with PoWER-BERT because they are not designed to control efficiency. More recently, TR-BERT (Ye et al., 2021) introduces a policy network trained via reinforcement learning to decide which vectors to skip.

LayerDrop (Fan et al., 2019) drops random layers during the training to be robust to pruning inspired by Huang et al. (2016). Word-level adaptive depth in Elbayad et al. (2019) and Liu et al. (2020b) might seemingly resemble with length reduction, but word vectors that reached the maximal layer are used for self-attention computation without updating themselves. Escaping a network early (Teerapittayanon et al., 2016; Huang et al., 2017) based on the confidence of the prediction (Xin et al., 2020, 2021; Schwartz et al., 2020; Liu et al., 2020a; Li et al., 2021) also offers a control over accuracy-efficiency trade-off by changing a threshold, but it is difficult to tune a threshold for a desired computational budget because of the example-wise adaptive computation.

Slimmable neural networks (Yu et al., 2018; Lee and Shin, 2018) reduce the hidden dimension for the any-time prediction. DynaBERT (Hou et al., 2020) can run at adaptive width (the number of attention heads and intermediate hidden dimension) and depth. Hardware-aware Transformers (Wang et al., 2020a) construct a design space with arbitrary encoder-decoder attention and heterogeneous layers in terms of different numbers of layers, attention heads, hidden dimension, and embedding dimension. SpAtten (Wang et al., 2020b) performs cascade token and head pruning for an efficient algorithm-architecture co-design.

Structured dropout A major innovation we introduce over the existing PoWER-BERT is the use of stochastic, structured regularization to make a transformer robust to the choice of length configuration in the inference time. Rippel et al. (2014) proposes a nested dropout to learn ordered representations. Similar to LengthDrop, it samples an index from a prior distribution and drops all units with a larger index than the sampled one.

Search There have been a series of attempts at finding the optimal network configuration by solving a combinatorial optimization problem. In computer vision, Once-for-All (Cai et al., 2019) use an evolutionary search (Real et al., 2019) to find a better configuration in dimensions of depth, width, kernel size, and resolution given computational budget. Similarly but differently, our evolutionary search is *mutli-objective* to find length configurations on the Pareto accuracy-efficiency frontier to cope with any possible computational budgets. Moreover, we only change the sequence length of hidden vectors instead of architectural model size like dimensions.

Sequence Length Shortformer (Press et al., 2020) initially trained on shorter subsequences and then moved to longer ones achieves improved perplexity than a standard transformer with normal training while reducing overall training time. Novel architectures with the efficient attention mechanism (Kitaev et al., 2020; Beltagy et al., 2020; Zaheer et al., 2020; Ainslie et al., 2020; Choromanski et al., 2020; Peng et al., 2021) are suggested to reduce the transformer's quadratic computational complexity in the input sequence length. Tay et al. (2020b) and Tay et al. (2020a) provide a survey of these efficient transformers and their benchmark comparison, respectively.

8 Conclusion and Future Work

In this work, we propose a new framework for training a transformer once and using it for efficient inference under any computational budget. With the help of training with LengthDrop and Drop-and-Restore process followed by the evolutionary search, our proposed Length-Adaptive Transformer allows any given transformer models to be used with any inference-time computational budget for both sequence-level and token-level classification tasks. Our experiments, on SQuAD 1.1, MNLI-m and SST-2, have revealed that the proposed algorithmic framework significantly pushes a better Pareto frontier on the trade-off between inference efficiency and accuracy. Furthermore, we have observed that the proposed Length-Adaptive Transformer could achieve up to 3x speed-up over the standard transformer without sacrificing accuracy, both in terms of FLOPs and wallclock time.

Although our approach finds *an* optimal length configuration of a trained classifier per computational budget, it leaves a open question whether the proposed approach could be further extended

to support per-instance length configuration by for instance training a small, auxiliary neural network for each computational budget. Yet another aspect we have not investigated in this paper is the applicability of the proposed approach to sequence generation, such as machine translation. We leave both of these research directions for the future.

Our approach is effective, as we have shown in this paper, and also quite simple to implement on top of existing language models. We release our implementation at https://github.com/clovaai/length-adaptive-transformer, which is based on HuggingFace's *Transformers* library (Wolf et al., 2019), and plan to adapt it for a broader set of transformer-based models and downstream tasks, including other modalities (Dosovitskiy et al., 2020; Touvron et al., 2020; Gulati et al., 2020).

Acknowledgments

The authors appreciate Clova AI members and the anonymous reviewers for their constructive feedback. Specifically, Dongyoon Han and Byeongho Heo introduced relevant works and gave insights from the view of the computer vision community. We use Naver Smart Machine Learning (Sung et al., 2017; Kim et al., 2018) platform for the experiments.

References

Joshua Ainslie, Santiago Ontanón, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. Etc: Encoding long and structured inputs in transformers. *arXiv preprint arXiv:2004.08483*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Han Cai, Chuang Gan, and Song Han. 2019. Once for all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*.

Qingqing Cao, Aruna Balasubramanian, and Niranjan Balasubramanian. 2020. Towards accurate and reliable energy measurement of nlp models. *arXiv preprint arXiv:2010.05248*.

Ting-Wu Chin, Ruizhou Ding, Cha Zhang, and Diana Marculescu. 2020. Towards efficient model compression via learned global ranking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1518–1528.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc V Le. 2020. Funnel-transformer: Filtering out sequential redundancy for efficient language processing. *arXiv preprint arXiv:2006.03236*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2019. Depth-adaptive transformer. *arXiv preprint arXiv:1910.10073*.

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

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. Power-bert: Accelerating bert inference via progressive word-vector elimination. In *International Conference on Machine Learning*, pages 3690–3699. PMLR.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. 2020. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*.

Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the systematic reporting of the energy and carbon footprints of machine learning. *arXiv preprint arXiv:2002.05651*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Lu Hou, Lifeng Shang, Xin Jiang, and Qun Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. *arXiv preprint arXiv:2004.04037*.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. 2017. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. 2016. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Hanjoo Kim, Minkyu Kim, Dongjoo Seo, Jinwoong Kim, Heungseok Park, Soeun Park, Hyunwoo Jo, KyungHyun Kim, Youngil Yang, Youngkwan Kim, et al. 2018. Nsml: Meet the mlaas platform with a real-world case study. *arXiv preprint arXiv:1810.09957*.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Hankook Lee and Jinwoo Shin. 2018. Anytime neural prediction via slicing networks vertically. *arXiv preprint arXiv:1807.02609*.

Xiaonan Li, Yunfan Shao, Tianxiang Sun, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2021. Accelerating bert inference for sequence labeling via early-exit. *arXiv preprint arXiv:2105.13878*.

Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E Gonzalez. 2020. Train large, then compress: Rethinking model size for efficient training and inference of transformers. *arXiv preprint arXiv:2002.11794*.

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

Yijin Liu, Fandong Meng, Jie Zhou, Yufeng Chen, and Jinan Xu. 2020b. Explicitly modeling adaptive depths for transformer. *arXiv preprint arXiv:2004.13542*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, pages 14014–14024.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A Smith, and Lingpeng Kong. 2021. Random feature attention. *arXiv preprint arXiv:2103.02143*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Ofir Press, Noah A Smith, and Mike Lewis. 2020. Shortformer: Better language modeling using shorter inputs. *arXiv preprint arXiv:2012.15832*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

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*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789.

Oren Rippel, Michael Gelbart, and Ryan Adams. 2014. Learning ordered representations with nested dropout. In *International Conference on Machine Learning*, pages 1746–1754.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2019. Green ai. *arXiv preprint arXiv:1907.10597*.

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

Abigail See, Minh-Thang Luong, and Christopher D Manning. 2016. Compression of neural machine translation models via pruning. *arXiv preprint arXiv:1606.09274*.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*.

Sandeep Subramanian, Ronan Collobert, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2020. Multi-scale transformer language models. *arXiv preprint arXiv:2005.00581*.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for bert model compression. *arXiv preprint arXiv:1908.09355*.

Nako Sung, Minkyu Kim, Hyunwoo Jo, Youngil Yang, Jingwoong Kim, Leonard Lausen, Youngkwan Kim, Gayoung Lee, Donghyun Kwak, Jung-Woo Ha, et al. 2017. Nsml: A machine learning platform that enables you to focus on your models. *arXiv preprint arXiv:1712.05902*.

Raphael Tang, Ashutosh Adhikari, and Jimmy Lin. 2018. Flops as a direct optimization objective for learning sparse neural networks. *arXiv preprint arXiv:1811.03060*.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2020a. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020b. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.

Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. 2016. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2020. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*.

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 neural information processing systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. 2020a. Hat: Hardware-aware transformers for efficient natural language processing. *arXiv preprint arXiv:2005.14187*.

Hanrui Wang, Zhekai Zhang, and Song Han. 2020b. Spatten: Efficient sparse attention architecture with cascade token and head pruning. *arXiv preprint arXiv:2012.09852*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

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

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. Berxit: Early exiting for bert with better fine-tuning and extension to regression. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 91–104.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

Deming Ye, Yankai Lin, Yufei Huang, and Maosong Sun. 2021. Tr-bert: Dynamic token reduction for accelerating bert inference. *arXiv preprint arXiv:2105.11618*.

Jiahui Yu and Thomas S Huang. 2019. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1803–1811.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. 2018. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*.