Faster Depth-Adaptive Transformers

Yijin Liu, [1] Fandong Meng,[2] Jie Zhou,[2] Yufeng Chen[1] and Jinan Xu[1]
[1]Beijing Jiaotong University, China
[2]Pattern Recognition Center, WeChat AI, Tencent Inc, China
{yijinliu, fandongmeng, withtomzhou}@tencent.com
{chenyf, jaxu}@bjtu.edu.cn

Abstract

Depth-adaptive neural networks can dynamically adjust depths according to the hardness of input words, and thus improve efficiency. The main challenge is how to measure such hardness and decide the required depths (*i.e.,* layers) to conduct. Previous works generally build a halting unit to decide whether the computation should continue or stop at each layer. As there is no specific supervision of depth selection, the halting unit may be under-optimized and inaccurate, which results in suboptimal and unstable performance when modeling sentences. In this paper, we get rid of the halting unit and estimate the required depths in advance, which yields a faster depth-adaptive model. Specifically, two approaches are proposed to explicitly measure the hardness of input words and estimate corresponding adaptive depth, namely 1) mutual information (MI) based estimation and 2) reconstruction loss based estimation. We conduct experiments on the text classification task with 24 datasets in various sizes and domains. Results confirm that our approaches can speed up the vanilla Transformer (up to 7x) while preserving high accuracy. Moreover, efficiency and robustness are significantly improved when compared with other depth-adaptive approaches.

Introduction

In the NLP literature, neural networks generally conduct a fixed number of computations over all words in a sentence, regardless of whether they are easy or difficult. In terms of both efficiency and ease of learning, it is preferable to dynamically vary the numbers of computations according to the hardness of input words (Dehghani et al. 2019).

Graves (2016) firstly proposes adaptive computation time (ACT) to improve efficiency of neural networks. Specifically, ACT employs a halting unit upon each word when processing a sentence, then this halting unit determines a probability that computation should continue or stop layer-by-layer. Its application to sequence processing is attractive and promising. For instance, ACT has been extended to reduce computations either by exiting early or by skipping layers for the ResNet (Figurnov et al. 2017), the vanilla Transformer (Elbayad et al. 2020), and the Universal Transformer (Dehghani et al. 2019).

However, there is no explicit supervision to directly train the halting unit of ACT, and thus how to measure the hardness of input words and decide required depths is the key point. Given a task, previous works generally treat the loss from different layers as a measure to implicitly estimate the required depths, *e.g.,* gradient estimation in ACT, or reinforcement rewards in SkipNet (Wang et al. 2018). Unfortunately, these approaches may lead to inaccurate depth selections with high variances, and thus unstable performance. More recently, the depth-adaptive Transformer (Elbayad et al. 2020) directly trains the halting unit with the supervision of 'pseudo-labels', which is generated by comparing task-specific losses over all layers. Despite its success, the depth-adaptive Transformer still relays on a halting unit, which brings additional computing costs for depth predictions, hindering its potential performance.

In this paper, we get rid of a halting unit when building our model, and thus no additional computing costs need to estimate depth. Instead, we propose two approaches to explicitly estimate the required depths in advance, which yield a faster depth-adaptive Transformer. Specifically, the MI-based approach calculates the mutual dependence between a word and all categorical labels. The larger the MI value of the word is, the more information of labels is obtained through observing this word, thus fewer depths are needed to learn an adequate representation for this word, and vice versa. Due to the MI-based approach is purely conducted in the data preprocessing stage, the computing cost is ignorable when compared with training a neural model in the depth-adaptive Transformer. The reconstruction loss based approach measures the hardness of learning a word by reconstructing it with its contexts in a sentence. The less reconstruction loss of the word is, the more easily its representation is learned. Therefore the index of the layer with minimum reconstruction loss can be regarded as an approximation for required depths. As a by-product, the reconstruction loss based approach is easy to apply to unsupervised scenarios, as it needs no task-specific labels. Both of the above approaches aim to find a suitable depth estimation. Afterward, the estimated depths are directly used to guide our model to conduct corresponding depth for both training and testing.

Without loss of generality, we base our model on the Transformer encoder. Extensive experiments are conducted on the text classification task (24 datasets in various sizes and domains). Results show that our proposed approaches can accelerate the vanilla Transformer up to 7x, while preserving high accuracy. Furthermore, we improve the efficiency and robustness of previous depth-adaptive models.

Our main contributions are as follows[1]:

- We are the first to estimate the adaptive depths in advance and do not rely on a halting unit to predict depths.
- We propose two effective approaches to explicitly estimate the required computational depths for input words. Specifically, the MI-based approach is computing efficient and the reconstruction loss based one is also applicable in unsupervised scenarios.
- Both of our approaches can accelerate the vanilla Transformer up to 7x, while preserving high accuracy. Furthermore, we improve previous depth-adaptive models in terms of accuracy, efficiency, and robustness.
- We provide thorough analyses to offer more insights and elucidate properties of our approaches.

Model

Depth Estimation

In this section, we introduce how to quantify the hardness of learning representations for input words and obtain corresponding estimated depths.

Mutual Information Based Estimation. Mutual Information (MI) is a general concept in information theory. It measures the mutual dependence between two random variables X and Y. Formally, the MI value is calculated as:

$$\mathrm{MI}(X;Y) = \sum_{y\in Y}\sum_{x\in X} p_{(X,Y)} \cdot \log\left(\frac{p_{(X,Y)}(x,y)}{p_X(x)\cdot p_Y(y)}\right) \quad (1)$$

where $p_{(X,Y)}$ is the joint probability of X and Y, and p_X and p_Y are the probability functions of X and Y respectively.

MI has been widely used for feature selection in the statistic machine learning literature (Peng, Long, and Ding 2005). In our case of text classification, X is the set of all words, and Y is the set of predefined labels. Given a word $x \in X$ and a label $y \in Y$, the value of $\mathrm{MI}(x, y)$ measures the degree of dependence between them. The larger $\mathrm{MI}(x, y)$ is, the greater certainty between this word x and label y is, and thus fewer computations are needed to learn an adequate representation for x to predict y. For example, the word 'terrible' can decide a 'negative' label with high confidence in sentiment analysis tasks, and thus it is unnecessary to conduct a very deep transformation when processing words with high MI values, and vice versa. Namely, we force our models not to merely focus on a few 'important' words and pay more attention to overview contexts when learning the representation of a sentence. In this way, our models avoid overfilling limited 'important' words, which also takes an effect of regularization, and thus improve generalization and robustness. Based on the above assumptions, it is intuitive and suitable to choose MI to quantify the difficulty of learning a word.

Formally, given a dataset with vocab X and label set Y, the MI value $\mathrm{MI}(x)$ for word x is calculated as:

$$\begin{aligned}\mathrm{MI}(x) = \sum_{y\in\{Y\}}\sum_{i_x\in\{0,1\}}\sum_{i_y\in\{0,1\}} & P(i_x, i_y) \\ \cdot \log & \left(\frac{P(i_x, i_y)}{P(i_x)\cdot P(i_y)}\right)\end{aligned} \quad (2)$$

where i_x is a boolean indicator whether word x exists in a sentence. Similarly, i_y indicates the existence of label y. In practice, the probability formulas $P(\cdot)$ in Equation (2) are calculated by frequencies of words, labels, or their combinations. A smooth factor (0.1 in our experiments) is introduced to avoid zero division. To avoid injecting information of golden labels when testing, we only use the training set to calculate MI values,

Once the MI value $\mathrm{MI}(x)$ for each word is obtained, we proceed to generate the pesudo-label of depth distribution $d(x)$ accordingly. As the histogram of MI values shown in Figure 1 (upper part), there is an obvious long tail phenomenon, which manifests that the distribution is extremely imbalanced. To alleviate this issue, we perform a logarithmic scaling for the original $\mathrm{MI}(x)$ as:

$$\mathrm{MI}_{log}(x) = -\log\left(\mathrm{MI}(x)\right) \quad (3)$$

Next, according to the scaled $\mathrm{MI}_{log}(x)$, we uniformly divide all words into N bins [2] with fixed-width margin, where N denotes a predefined number of bins (*i.e.,* maximum depth). Consequently, the estimated depth value $d(x)$ for word x is the index of corresponding bins.

The MI-based approach is purely calculated at the data preprocessing stage, thus it is highly efficient in computation and does not rely on additional trainable parameters.

Reconstruction Loss Based Estimation. Generally in a sentence, several words may bring redundant information that has been included by their contexts. Thus if we mask out these trivial words, it would be easier to reconstruct them than others. Namely, The less reconstruction loss of a word is, the more easily its representation is learned. Based on the above principle, we utilize this property of reconstruction loss to quantify the hardness of learning the representation for input words and then estimate their required depths. Firstly, we finetune BERT (Devlin et al. 2019) with a masked language model task (MLM) on datasets of downstream tasks. Note that we modify BERT to make predictions at any layers with a shared classifier, which is also known as *anytime prediction* (Huang et al. 2017; Elbayad et al. 2020). The losses from all layers are summed up [3] to

[1]Codes will appear at https://github.com/Adaxry/Adaptive-Transformer

[2]We set N to 12 for the compatibility of BERT.

[3]We experimented with different weights (*e.g.,* random sample, or linearly decaying with the number of layers) for different layers, and finally choose the simple equal weights.

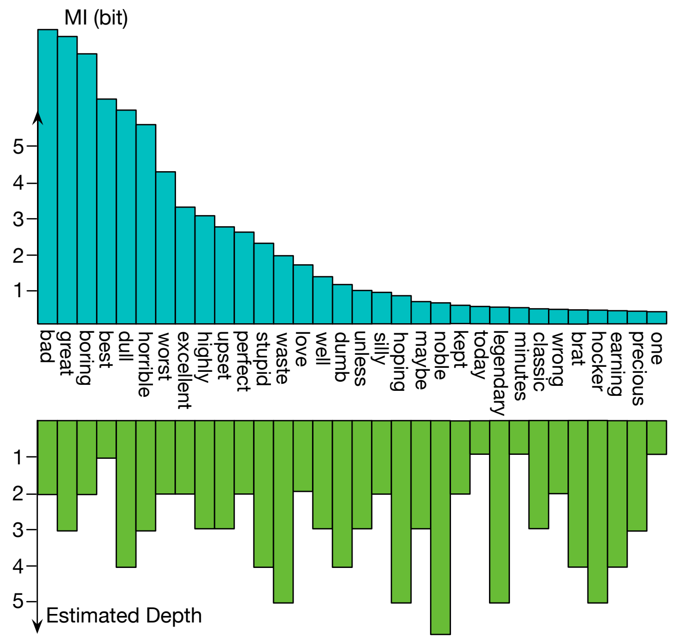

Figure 1: The histogram of MI values of partial words from IMDB (upper part), and corresponding depths of these words by using the reconstruction loss based estimation (the bottom part).

the final loss. After finetuning the MLM, given an input sentence $\boldsymbol{x}$ with $|\boldsymbol{x}|$ words, we sequentially replace each word $\boldsymbol{x}_t$ ($t \in [1, |\boldsymbol{x}|]$) with a special symbol <MASK>, and then feed the sentence with a <MASK> into the MLM. Finally, the index of a layer with the minimum loss is selected as the estimated depth value $d(\boldsymbol{x}_t)$:

$$d(\boldsymbol{x}_t) = \arg\min_n (loss_n - \lambda n) \tag{4}$$

where $n \in N$ is the index of layer, $loss_n$ is the loss of <MASK> in the n-th layer, and λn is the penalty factor to encourage a lower selection [4]. Specifically, we train MLMs following the experimental setup of BERT (Devlin et al. 2019) with two major differences: 1) We make predictions at every layer with a shared classifier instead of only at the final layer in BERT; 2) We remove the next sentence prediction task following RoBERTa (Liu et al. 2019).

Comparisons Between the Two Approaches. Although the above approaches perform differently, they both serve as a measure to estimate required depths for input words from the perspective of learning their representations. We proceed to make a detailed comparison between the two approaches.

In the term of computational cost, the MI-based approach calculates MI values, and then stores the word-depth pairs that resemble word embeddings. The above procedures merely happen at the stage of data preprocessing, which requires trivial computational cost and does not rely on additional trainable parameters, and thus the MI-based approach is highly efficient in computation. In contrast, the reconstruction loss based approach needs to train several MLMs with *anytime prediction*, which yields extra computational costs. Considering the MLMs are dependent on the main model, the calculation of depths can be conducted in advance in a piplined manner.

[4]We elaborate the effect and choice of λ in the following analytical Section.

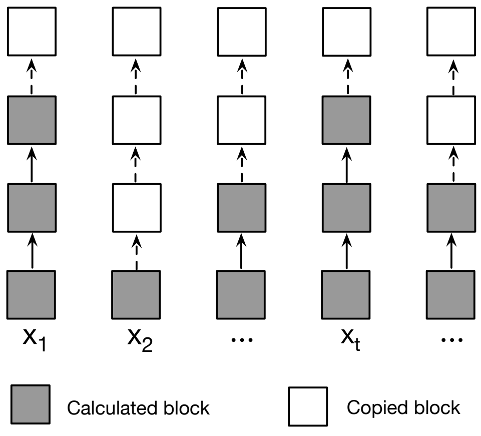

Figure 2: The overview of our depth-adaptive Transformer. Once a word $\boldsymbol{x}_t$ achieve its own depth $d(\boldsymbol{x}_t)$, it will simply copy states to upper layers.

As the histogram shown in Figure 1, we observe different preferences between the two estimations. Firstly, the MI-based approach (upper part) tends to assign higher MI values to label-relevant words (*e.g.,* opinion words 'perfect' and 'horrible' in IMDB). After the scaling function described by Equation (3), these opinion words are assigned a lower number of depths, namely fewer computational steps. Such operations make our models not only focus on a few 'important' words, but also pay more attention to the overview contexts, which takes an effect of regularization, and thus improve generalization and robustness.

Unlike the bias for label-related words in the MI-based approach, the reconstruction based approach (bottom part in Figure 1) purely relies on the unsupervised context to measures the hardness of learning, which is good at recognizing common words (*e.g.,* 'today', 'one' and 'me'), and assigns a smaller number of computations, and vice versa. As a by-product, the reconstruction loss based approach is applicable to unsupervised scenarios, as it needs no task-specific labels.

Depth-Adaptive Mechanism

As the overview shown in Figure 2, we stack N layers of the Transformer encoder to model a sentence. The Transformer encoder consists of two sub-layers in each layer. The first sub-layer is a multi-head dot-product self-attention and the second one is a position-wise fully connected feed-forward network. We refer readers to the original paper (Vaswani et al. 2017) for more details.

To make sure all hidden states of the same layer are avail-

Dataset	Classes	Type	Average Lenghts	Max Lengths	Train Sample	Test Sample
TREC (Li and Roth 2002)	6	Question	12	39	5,952	500
AG's News (Zhang, Zhao, and LeCun 2015)	4	Topic	44	221	120,000	7,600
DBPedia (Zhang, Zhao, and LeCun 2015)	14	Topic	67	3,841	560,000	70,000
Subj (Pang and Lee 2004)	2	Sentiment	26	122	10,000	CV
MR (Pang and Lee 2005)	2	Sentiment	23	61	10,622	CV
Amazon-16 (Liu, Qiu, and Huang 2017)	2	Sentiment	133	5,942	31,880	6,400
IMDB (Maas et al. 2011)	2	Sentiment	230	2,472	25,000	25,000
Yelp Polarity (Zhang, Zhao, and LeCun 2015)	2	Sentiment	177	2,066	560,000	38,000
Yelp Full (Zhang, Zhao, and LeCun 2015)	5	Sentiment	179	2,342	650,000	50,000

Table 1: Dataset statistics. 'CV' refers to 5-fold cross-validation. There are 16 subsets in Amazon-16.

able to compute self-attention, once a word $\boldsymbol{x}_t$ reaches its own maximal layer $d(\boldsymbol{x}_t)$, it will stop computation, and simply copy its states to the next layer until all words stop or the maximal layer N is reached. Formally, at the n-th layer, for the word x_t, its hidden state $\boldsymbol{h}_i^n$ are updated as follows:

$$\boldsymbol{h}_t^n = \begin{cases} \boldsymbol{h}_t^{n-1} & \text{if } n > d(x_t) \\ \text{Transformer}(\boldsymbol{h}_t^{n-1}) & \text{else} \end{cases} \tag{5}$$

where $n \in [1, N]$ refers to the index of the layer. Especially, $\boldsymbol{h}_t^0$ is initialized by the BERT embedding.

Task-specific Settings

After dynamic steps of computation for each word position, we make task-specific predictions upon the maximal stop layer $n_{max} \in [1, N]$ among all word positions. The feature vector $\boldsymbol{v}$ consists of mean and max pooling of output hidden states $\boldsymbol{h}^{n_{max}}$, and is activated by ReLU. Finally, a softmax classifier are built on $\boldsymbol{v}$. Formally, the above-mentioned procedures are computed as follows:

$$\begin{aligned} \boldsymbol{v} &= \text{ReLU}([\max(\boldsymbol{h}^{n_{max}}); \text{mean}(\boldsymbol{h}^{n_{max}})]) \\ P(\widetilde{y}|\boldsymbol{v}) &= \text{softmax}(\boldsymbol{W}\boldsymbol{v} + \boldsymbol{b}) \end{aligned} \tag{6}$$

where $\boldsymbol{W} \in \mathbb{R}^{d_{model} \times |S|}$ and $\boldsymbol{b} \in \mathbb{R}^{|S|}$ are parameters of the classifier, $|S|$ is the size of the label set, and $P(\widetilde{y}|\boldsymbol{v})$ is the probability distribution. At the training stage, we use the cross-entropy loss computed as:

$$Loss = -\sum_{i=1}^{|S|} y_i log(P_i(\widetilde{y}|\boldsymbol{v})) \tag{7}$$

where y_i is the golden label. For testing, the most probable label $\hat{y}$ is chosen from above probability distribution described by Equation (6):

$$\hat{y} = \arg\max P(\widetilde{y}|\boldsymbol{v}) \tag{8}$$

Experiments

Task and Datasets

Text classification aims to assign a predefined label to text (Zhang, Zhao, and LeCun 2015), which is a classic task for natural language processing and is generally evaluated by accuracy score. Generally, The number of labels may range from two to more, which corresponds to binary and fine-grained classification. We conduct extensive experiments on the 24 popular benchmarks collected from diverse domains (*e.g., topic*, *sentiment*) ranging from modestly sized to large-scaled. The statistics of these datasets are listed in Table 1.

Implementation Details

For the MI-based estimation approach, we calculate word-depth pairs on the training set in advance and then calculate depths for words in the test set. For the reconstruction based approach, we calculate word-depth pairs for both train and test set without using label information. The penalty factor λ in the reconstruction loss based approach is set to 0.1. Dropout (Srivastava et al. 2014) is applied to word embeddings, residual connection , and attention scores with a rate of 0.1. Models are optimized by the Adam optimizer (Kingma and Ba 2014) with gradient clipping of 5 (Pascanu, Mikolov, and Bengio 2013). $BERT_{base}$ is used to initialize the Transformer encoder. Long sentences exceed 512 words are clipped.

Main Results

Results on Amazon-16. Amazon-16 consists of consumer comments from 16 different domains (*e.g.,* Apparel). We compare our approaches with different baseline models in Table 2. The Multi-Scale Transformer (Guo et al. 2019b) is designed to capture features from different scales, and the Star-Transformer (Guo et al. 2019a) is a lightweight Transformer with a star-shaped topology. Due to the absence of a powerful contextual model (*e.g.,* BERT), their results underperform others by a margin. The Transformer model is finetuned on BERT and conducts fixed 12 layers for every instance, which yields a strong baseline model. Following the setup of depth-adaptive Transformer, we add a halting unit on the bottom layer of the Transformer encoder, and generate a 'pesudo-label' for the halting unit by classification accuracy. Our approaches (the last two columns in Table 2) achieve better or comparable performance over these strong baseline models. The MI-based approach also takes a regularization effect, and thus it achieves better performance than the reconstruction counterpart.

Data / Model	Multi-Scale Transformer	Star-Transformer	Transformer	Transformer w/ halting unit	Transformer w/ MI estimation	Transformer w/ reconstruction
Apparel	86.5	88.7	**91.9**	91.6	91.4	91.8
Baby	86.3	88.0	88.8	88.1	**90.6**	88.4
Books	87.8	86.9	89.5	88.3	**89.6**	89.5
Camera	89.5	91.8	91.8	92.7	**93.8**	92.9
Dvd	86.5	87.4	88.3	**91.7**	91.4	91.5
Electronics	84.3	87.2	**90.8**	89.3	90.6	90.2
Health	86.8	89.1	**91.7**	91.3	91.6	88.4
Imdb	85.0	85.0	88.3	89.8	89.5	**90.6**
Kitchen	85.8	86.0	87.6	88.1	89.2	86.8
Magazines	91.8	91.8	94.2	**94.8**	94.6	94.7
Mr	78.3	79.0	**83.7**	81.6	82.3	82.5
Music	81.5	84.7	**89.9**	89.3	89.5	87.2
Software	87.3	90.9	91.2	92.9	92.3	**93.8**
Sports	85.5	86.8	87.1	89.2	88.4	**89.8**
Toys	87.8	85.5	89.7	90.3	**90.9**	89.7
Video	88.4	89.3	93.4	**94.3**	93.1	93.5
Avg	86.2	87.4	89.9	90.2	**90.5**	90.1

Table 2: Accuracy scores (%) on the Amazon-16 datasets. Best results on each dataset are bold. The results of Multi-Scale Transformer (Guo et al. 2019b) is cited from the original paper, and other results are our implementations with several recent advanced techniques (*e.g.,* BERT initialization) under the unified setting.

Models / Dataset	TREC	MR	Subj	IMDB	AG.	DBP.	Yelp P.	Yelp F.	Avg.
RCRN (Tay, Tuan, and Hui 2018)	96.20	–	–	92.80	–	–	–	–	–
Cove (McCann et al. 2017)	95.80	–	–	91.80	–	–	–	–	–
Text-CNN (Kim 2014)	93.60	81.50	93.40	–	–	–	–	–	–
Multi-QT (Logeswaran and Lee 2018)	92.80	82.40	94.80	–	–	–	–	–	–
AdaSent (Zhao, Lu, and Poupart 2015)	92.40	83.10	95.50	–	–	–	–	–	–
CNN-MCFA (Amplayo et al. 2018)	94.20	81.80	94.40	–	–	–	–	–	–
Capsule-B (Yang et al. 2018)	92.80	82.30	93.80	–	92.60	–	–	–	–
DNC+CUW (Le, Tran, and Venkatesh 2019)	–	–	–	–	93.90	–	96.40	65.60	–
Region-Emb (Qiao et al. 2018)	–	–	–	–	92.80	98.90	96.40	64.90	–
Char-CNN (Zhang, Zhao, and LeCun 2015)	–	–	–	–	90.49	98.45	95.12	62.05	–
DPCNN (Johnson and Zhang 2017)	–	–	–	–	93.13	99.12	97.36	69.42	–
DRNN (Wang 2018)	–	–	–	–	94.47	99.19	97.27	69.15	–
SWEM-concat (Shen et al. 2018)	92.20	78.20	93.00	–	92.66	98.57	95.81	63.79	–
Star-Transformer (Guo et al. 2019a) †	93.00	79.76	93.40	94.52	92.50	98.62	94.20	63.21	88.65
BERT (Devlin et al. 2019)	–	–	–	95.49	–	99.36	98.11	70.68	–
XLNet (Yang et al. 2019)	–	–	–	**96.80**	95.55	**99.40**	**98.63**	72.95	–
Transformer (Vaswani et al. 2017) †	96.00	83.75	96.00	95.58	95.13	99.22	98.09	69.80	91.69
w/ Halting unit (Elbayad et al. 2020) †	95.80	83.23	96.00	95.80	95.50	99.30	98.25	69.75	91.70
w/ MI estimation (ours) †	**96.50**	**84.20**	96.00	96.72	**95.90**	99.32	98.10	**72.98**	**92.46**
w/ Reconstruction estimation (ours) †	96.20	83.90	**96.30**	96.60	95.65	99.25	98.00	69.58	91.93

Table 3: Accuracy scores (%) on modestly sized and large-scaled datasets. 'AG.', 'DBP.', 'Yelp P.' and 'Yelp F.' are the abbreviations of 'AG's News', 'DBPedia', 'Yelp Polarity' and 'Yelp Full', respectively. † is our implementations with several recent advanced techniques and *analogous* parameter sizes. 'Transformer' is initialized by $BERT_{base}$ with 12 fixed layers.

Results on Larger Benchmarks. Although the Amazon-16 benchmark is challenging, its small data size makes the results prone to be unstable, therefore we conduct experiments on larger benchmarks for a more convincing conclusion. In this paragraph, we only focus on the classification accuracy listed in Table 3, and more detailed results about computing speed and model robustness will be discussed in the next section.

The upper part of Table 3 lists several high-performance baseline models. Their detailed descriptions are omitted here. In terms of accuracy, our approaches achieve comparable performance with these state-of-the-art models. At the bottom part of Table 3, we finetune BERT as our strong baseline model. Results show that this baseline model performs on par with the state-of-the-art XLNet (Yang et al. 2019). Then we build a halting unit at the bottom of the baseline model under the same setup with the depth-adaptive Transformer. Results show that applying a halting unit has no

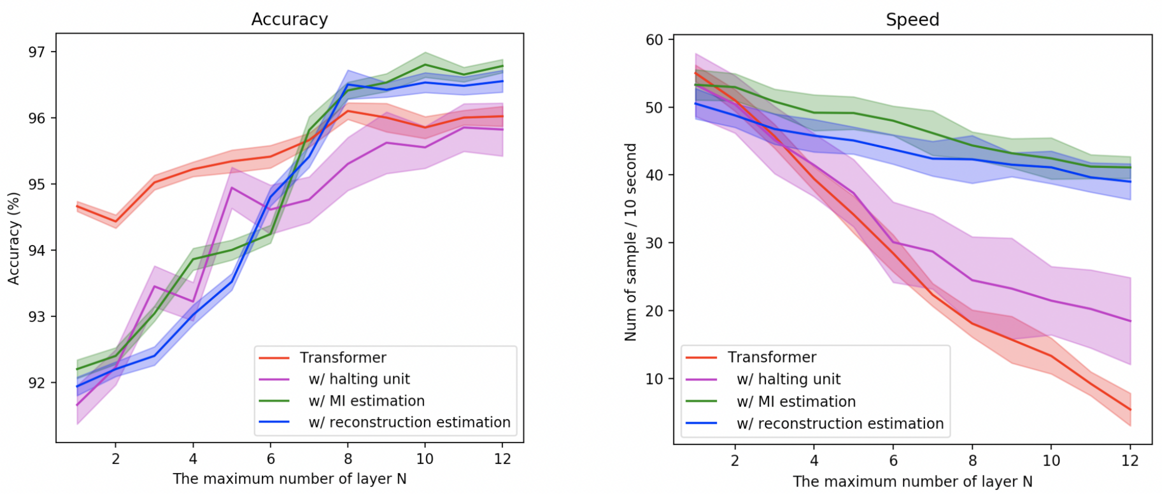

Figure 3: Accuracy scores (a) and speed (b) for each model on IMDB when $N \in [1, 12]$. The solid line indicates the mean performance, and the size of the colored area indicates variance (used to measure robustness). 'speed' is the number of samples calculated in ten-second on one Tesla P40 GPU with the batch size of 1.

obvious impact on accuracy. The last two rows list results of our estimation approaches, where the MI-based approach brings in consistent improvements over the baseline and the Transformer w/ a halting unit, by +0.77% and +0.76% on average, respectively. We speculate the improvements mainly come from the additional deep supervision and regularization effect of the MI-based approach. In contrast, the reconstruction based approach only show improvements over the baseline model (+0.24%) and the Transformer w/ a halting unit (+0.23%) by a small margin.

Analysis

We conduct analytical experiments on the modestly sized IMDB to offer more insights and elucidate the properties of our approaches.

Effect of the Maximum Number of Layers

Firstly, we train several fixed-layer Transformers with N ranging from one to twelve, and then build a halting unit on the above Transformers to dynamically adjust the actual number of layers to conduct. Meanwhile, we respectively utilize our two approaches on the fixed-layer Transformer to activate dynamic layers. Note that each model is trained with different random initialization three times and we report the mean and variance. Here, we take the variance value to measure the robustness against the random initialization and different depth selections. As drawn in Figure 3, solid lines are the mean performance, and the size of the colored areas indicate variances.

Accuracy and Robustness. Results of accuracy and robustness are drawn in Figure 3 (a). In the lower layers ($N \in [1, 6]$), as the searching space for depth selection is small, the depth-adaptive models perform worse than the Transformer baseline. In contrast, when $N \in [6, 12]$, the depth-adaptive models come up with the baseline. Due to the additional depth supervision and the regularization effect, the application of our approaches can further significantly improve accuracy and robustness over both the Transformer and w/ a halting unit. (green and blue lines vs. purple line in Figure 3 (a))

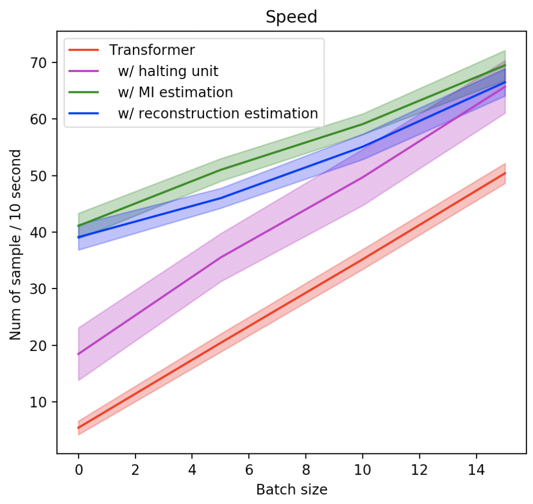

Figure 4: Speed for each model on IMDB when $batchsize \in [1, 15]$. The solid line indicates the mean performance, and the size of the colored area indicates variance (used to measure robustness). 'speed' is the number of samples calculated in ten seconds on one Tesla P40 GPU.

Speed and Robustness. Figure 3 (b) shows the speed and robustness of each model. The speed of vanilla Transformer almost linearly decays with the growth of the number of

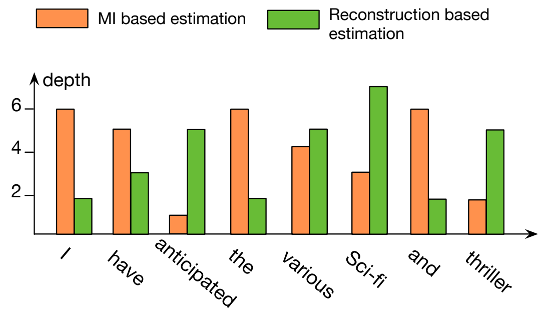

Figure 5: The histogram of the depth distribution of a case from IMDB, which is estimated by our approaches.

layers. As $N \in [1, 3]$, due to the additional prediction for depths, models w/ halting unit runs a bit slower than the baseline. However, the superiority of adaptive depths becomes apparent with the growth of the number of layers. In particular, as $N = 12$, the model w/ halting run 3.4x faster than the fixed-layer baseline (pure lines vs. red line in Figure 3 (b)). As our approaches free the dependency for depth prediction and can further speed up the model, both of our approaches run about 7x faster than the fixed-layer baseline (green and blue lines vs. red line in Figure 3 (b)). In addition, our approaches perform more robust in the term of speed gains than the Transformer w/ a halting unit.

Speed on Different Batch Sizes

The depth-adaptive models conduct dynamic computations at each word position, and thus the actually activated depths are decided by the maximal depth value. As a result, when the batch size gets larger, the final activated depth may potentially become larger as well, which may hurt the effectiveness of depth-adaptive models. In this section, we fix the maximal number of layer N to 12, and then compare the speed of each model. As shown in Figure 4, the speed gain of the depth-adaptive models (green, blue and purple lines in Figure 4) grows slower than the vanilla Transformer (red line in Figure 4). However, the absolute speed of depth-adaptive models is still much faster than the vanilla Transformer. We leave the further improvement of depth-adaptive models on larger batch sizes to future works.

Effect of Penalty Factor λ

If no constraints are applied on the depth selection, the reconstruction loss based approach tends to choose a layer as deep as possible, and thus an extra penalty factor λ is necessary to encourage a lower choice. We simply search $\lambda \in [0, 0.2]$, and finally set it to 0.1 for a good accuracy-speed trade-off. The detailed results are list in Table 4.

Case Study

We choose a random sentence from the IMDB dataset, and show the estimated depths outputted by both approaches in Figure 5 (upper part). We observe that the MI-based estimation tends to assign a smaller number of depths for opinion words, *e.g.,* 'anticipated' and 'thriller'. While the reconstruction loss based estimation is prone to omit common words, e.g., 'and'.

λ	0	0.05	0.10	0.15	0.20
accuracy	96.54	96.31	96.55	96.27	96.29
speed	23	33	48	54	58
average depth	9.5	6.3	4.5	3.9	3.6

Table 4: Effect of penalty factor λ. The definition of 'speed' is same as that in Figure 3. 'average depth' is the average predicted depth of words in test set.

Related Work

Our work is mainly inspired by ACT (Graves 2016), and we further explicitly train the halting union with the supervision of estimated depths. Unlike Universal Transformer (Dehghani et al. 2019) iteratively applies ACT on the same layer, we dynamically adjust the amount of both computation and model capacity.

A closely related work named 'Depth-Adaptive Transformer' (Elbayad et al. 2020) uses task-specific loss as an estimation of depth selection. Our approaches are different from it in three major aspects: 1) We get rid of the halting unit and remove the additional computing cost for depths, thus yield a faster depth-adaptive Transformer; 2) our MI-based estimation does not need to train an extra module, and is highly efficient in computation; 3) our reconstruction loss based estimation is unsupervised, and can be easily applied on general unlabeled texts. Another group of works also aims to improve efficiency of neural network through reducing the entire layers, *e.g.,* DynaBERT (Hou et al. 2020), LayerDrop (Fan, Grave, and Joulin 2019) and MobileBERT (Sun et al. 2020). In contrast, our approaches perform adaptive depths in the fine-grained word level.

Conclusion

We get rid of the halting unit and remove the additional computing cost for depths, thus yield a faster depth-adaptive Transformer. Specifically, we propose two effective approaches 1) mutual information based estimation and 2) reconstruction loss based estimation. Experimental results confirm that our approaches can speed up the vanilla Transformer (up to 7x) while preserving high accuracy. Moreover, we significantly improve previous depth-adaptive models in terms of accuracy, efficiency, and robustness. We will further explore the potential improvement of the depth-adaptive Transformer when facing larger batch size in future work.

Acknowledgments

The research work described in this paper has been supported by the National Key R&D Program of China (2020AAA0108001) and the National Nature Science Foundation of China (No. 61976015, 61976016, 61876198 and 61370130). This work was done when Yijin Liu was interning at Pattern Recognition Center, WeChat AI, Tencent Inc, China. Jinan Xu is the corresponding author of the paper.

References

Amplayo, R. K.; Lee, K.; Yeo, J.; and Hwang, S.-w. 2018. Translations as additional contexts for sentence classification. In *Proceedings of IJCAI*.

Dehghani, M.; Gouws, S.; Vinyals, O.; Uszkoreit, J.; and Łukasz Kaiser. 2019. Universal Transformers. In *Proceedings of ICLR*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL*.

Elbayad, M.; Gu, J.; Grave, E.; and Auli, M. 2020. Depth-Adaptive Transformer. In *ICLR*.

Fan, A.; Grave, E.; and Joulin, A. 2019. Reducing Transformer Depth on Demand with Structured Dropout. In *International Conference on Learning Representations*.

Figurnov, M.; Collins, M. D.; Zhu, Y.; Zhang, L.; Huang, J.; Vetrov, D.; and Salakhutdinov, R. 2017. Spatially Adaptive Computation Time for Residual Networks. In *Proceedings of CVPR*.

Graves, A. 2016. Adaptive computation time for recurrent neural networks. *arXiv* .

Guo, Q.; Qiu, X.; Liu, P.; Shao, Y.; Xue, X.; and Zhang, Z. 2019a. Star-transformer. In *Proceedings of NAACL*.

Guo, Q.; Qiu, X.; Liu, P.; Xue, X.; and Zhang, Z. 2019b. Multi-Scale Self-Attention for Text Classification. *arXiv preprint arXiv:1912.00544* .

Hou, L.; Shang, L.; Jiang, X.; and Liu, Q. 2020. DynaBERT: Dynamic BERT with Adaptive Width and Depth. *arXiv preprint arXiv:2004.04037* .

Huang, G.; Chen, D.; Li, T.; Wu, F.; van der Maaten, L.; and Weinberger, K. Q. 2017. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844* .

Johnson, R.; and Zhang, T. 2017. Deep Pyramid Convolutional Neural Networks for Text Categorization. In *Proceedings of ACL*.

Kim, Y. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv* .

Le, H.; Tran, T.; and Venkatesh, S. 2019. Learning to Remember More with Less Memorization. In *Proceedings of ICLR*.

Li, X.; and Roth, D. 2002. Learning question classifiers. In *Proceedings of COLING*, 1–7.

Liu, P.; Qiu, X.; and Huang, X. 2017. Adversarial Multi-task Learning for Text Classification. In *Proceedings of ACL*.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* .

Logeswaran, L.; and Lee, H. 2018. An efficient framework for learning sentence representations. *arXiv* .

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *Proceedings of ACL*, 142–150.

McCann, B.; Bradbury, J.; Xiong, C.; and Socher, R. 2017. Learned in Translation: Contextualized Word Vectors. In *Proceedings of NeurIPS*.

Pang, B.; and Lee, L. 2004. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In *Proceedings of ACL*.

Pang, B.; and Lee, L. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *Proceedings of ACL*.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of ICML*.

Peng, H.; Long, F.; and Ding, C. 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence* 27(8): 1226–1238.

Qiao, C.; Huang, B.; Niu, G.; Li, D.; Dong, D.; He, W.; Yu, D.; and Wu, H. 2018. A New Method of Region Embedding for Text Classification. In *Proceedings of ICLR*.

Shen, D.; Wang, G.; Wang, W.; Min, M. R.; Su, Q.; Zhang, Y.; Li, C.; Henao, R.; and Carin, L. 2018. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *Proceedings of ACL*.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The journal of machine learning research* .

Sun, Z.; Yu, H.; Song, X.; Liu, R.; Yang, Y.; and Zhou, D. 2020. MobileBERT: a Compact Task-Agnostic BERT for Resource-Limited Devices. In *ACL*.

Tay, Y.; Tuan, L. A.; and Hui, S. C. 2018. Recurrently Controlled Recurrent Networks. In *Proceedings of NeurIPS*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NIPS*.

Wang, B. 2018. Disconnected Recurrent Neural Networks for Text Categorization. In *Proceedings of ACL*.

Wang, X.; Yu, F.; Dou, Z.-Y.; Darrell, T.; and Gonzalez, J. E. 2018. SkipNet: Learning Dynamic Routing in Convolutional Networks. In *The European Conference on Computer Vision (ECCV)*.

Yang, M.; Zhao, W.; Ye, J.; Lei, Z.; Zhao, Z.; and Zhang, S. 2018. Investigating Capsule Networks with Dynamic Routing for Text Classification. In *Proceedings of EMNLP*. Brussels, Belgium.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R. R.; and Le, Q. V. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NIPS*.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, 649–657.

Zhao, H.; Lu, Z.; and Poupart, P. 2015. Self-adaptive hierarchical sentence model. In *Proceedings of IJCAI*.