DACT-BERT: INCREASING THE EFFICIENCY AND INTERPRETABILITY OF BERT BY USING ADAPTIVE COMPUTATION TIME.

Anonymous authors
Paper under double-blind review

ABSTRACT

Large-scale pre-trained language models have shown remarkable results in diverse NLP applications. Unfortunately, these performance gains have been accompanied by a significant increase in computation time and model size, stressing the need to develop new or complementary strategies to increase the efficiency and interpretability of current large language models, such as BERT. In this paper we propose DACT-BERT, a differentiable adaptive computation time strategy for BERT language model. DACT-BERT adds an adaptive computation mechanism to the regular processing pipeline of BERT. This mechanism controls the number of transformer blocks that BERT needs to execute at inference time. By doing this, the model makes predictions based on the most appropriate intermediate representations for the task encoded by the pre-trained weights. With respect to previous works, our method has the advantage of being fully differentiable and directly integrated to BERT's main processing pipeline. This enables the incorporation of gradient-based transparency mechanisms to improve interpretability. Furthermore, by discarding useless steps, DACT-BERT facilitates the understanding of the underlying process used by BERT to reach an inference. Our experiments demonstrate that our approach is effective in significantly reducing computational complexity without affecting model accuracy. Additionally, they also demonstrate that DACT-BERT helps to improve model interpretability.

1 INTRODUCTION

Recently, the use of pre-trained language models based on large-scale transformers (Vaswani et al., 2017) has experimented a substantial increase, mainly due to its success to support a large variety of NLP tasks (Rogers et al., 2020). In particular, BERT (Devlin et al., 2019) has become one of the most popular tools. The usual pipeline consists of finnetuning BERT by adapting and retraining its classification head to meet the requirements of a specific NLP task. Unfortunately, the benefits of using a powerful model such as BERT are also accompanied by a highly demanding computational load. In effect, current pre-trained language models such as BERT have many layers and millions, or even billions, of parameters, making them computationally intensive both during training and inference. Furthermore, the large number of parameters makes these models hard to analyze, limiting their interpretability.

While high accuracy is usually the ultimate goal, computational efficiency and model interpretability are also desirable objectives. In terms of computational efficiency, the use of a demanding model not only causes longer processing times and limits applicability to low-end devices, but it also has major implications in terms of the environmental impact of AI technologies (Schwartz et al., 2019). As an example, (Strubell et al., 2019) provides an estimation of the carbon footprint of several large NLP models, including BERT, concluding that they are becoming environmentally unfriendly. In terms of interpretability, potential biased or malicious uses of AI technologies, in particular NLP applications, are increasing the need to provide them with the ability to explain their decisions (Gilpin et al., 2018). As an example, (Goodman & Flaxman, 2016) analyzes the implications of a new legislation of the European Union that enforces the right to explanation on algorithmic decision-making.

Fortunately, recent works have shown that behind the immense capacity of BERT, there is considerable redundancy and over-parametrization (Kovaleva et al., 2019; Rogers et al., 2020). Consequently, recent works have explored new strategies to develop efficient and compact versions of BERT. A relevant strategy consists of distilling the knowledge from a pre-trained model into a smaller network. This is the approach followed by works such as Sanh et al. (2020) and Jiao et al. (2020). The main disavantage of this strategy is the inherent complexity of conducting an efficient and effective distillation process. An alternative strategy consists of providing BERT with an adaptive mechanism to control its processing pipeline according to the complexity of the current query. This is the approach followed by works such as Xin et al. (2020b), Liu et al. (2020), and Zhou et al. (2020). As a relevant common limitation of these adaptive models, all of them depend heavily on an external hyperparameter which must be carefully tuned for each model and task.

In this work, we present DACT-BERT, a Differentiable Adaptive Computation Time mechanism (Graves, 2016) to control the complexity of the regular processing pipeline of BERT. Specifically, DACT-BERT controls the number of transformer blocks that BERT needs to execute at inference time. This allows DACT-BERT to make predictions using a variable number of transformer blocks, taking advantage of the different information encoded in each intermediate representation. In other words, the model answers each query by making a prediction based on the most appropriate intermediate representation. As a major novelty, DACT-BERT integrates a totally differentiable module (Eyzaguirre & Soto, 2020) that allows us to train a halting neuron after each transformer block. This neuron indicates the confidence of the model at that point. Based on this, the execution can be stopped when the response predicted by the model stabilizes in a given output, making unnecessary to continue running the processing pipeling.

With respect to previous works, our method has the advantage of being fully differentiable and directly integrated to BERT's main processing pipeline. This avoids complexities associated to the calibration of external hyperparameter that are task dependent. In terms of interpretability, by discarding useless steps, DACT-BERT facilitates the understanding of the underlying process used by BERT to reach each inference. Furthermore, this enables the incorporation of gradient-based transparency mechanisms (Sundararajan et al., 2017) to improve interpretability. Our experiments using tasks from the GLUE benchmark (Wang et al., 2018) demonstrate that our approach is effective in significantly reducing computational complexity without affecting model accuracy. Additionally, they also demonstrate that DACT-BERT helps to improve model interpretability by explaining the relationship between not only predictions and inputs, but also the model confidence score and inputs.

2 Related Work

2.1 Efficient Transformers

2.1.1 Static Approaches

Several architectures have been designed to avoid overcomputing in transformer-based models. One such strategy is to use lightweight architectures that are trained from scratch. As an example, ALBERT proposes cross-layer parameter sharing as a way to improve model efficiency. Similar methodologies have also been previously explored by Bai et al. (2019) and Dehghani et al. (2019), demonstrating the effectiveness of weight-tied Transformers. A second strategy is to distill the knowledge of pretrained models into a more compact "student". Models such as PKD-BERT (Sun et al., 2019), TinyBERT (Jiao et al., 2020), and DistilBERT (Sanh et al., 2020) compress the knowledge of large models (teachers) into lighter ones (students). In this way, a more compact model is obtained with a performance that is usually close to the original one. While these approaches effectively reduce the total calculation needed to execute the model (measured in FLOPs or number of parameters), they are limited in the same way as BERT, they do not take into account that some tasks could be less complicated than others and always use the same amount of computation. Furthermore, the application of a suitable distillation process is not a trivial task that introduces extra complications.

2.1.2 Dynamic Approaches

Recently, a series of algorithms have been proposed to reduce computation in Transformer language models based on the concept of model early exiting. Models such as DeeBERT, (Xin et al., 2020a)

FastBert (Liu et al., 2020), and PABEE (Zhou et al., 2020) employ strategies similar to those used by Kaya et al. (2019) on convolutional neural networks. These models introduce intermediate classifiers after each Transformer block. These classifiers are then trained independently from these blocks (not end-to-end). After both training stages, a "halting criterion" is used to dynamically determine the number of blocks needed to perform a specific prediction. Instead of using a brittle confidence approach (Guo et al., 2017) to determine when to stop, recent adaptive Transformer architectures rely on computing the Shannon's entropy of the output probabilities (Xin et al., 2020a; Liu et al., 2020) or patience (Zhou et al., 2020). As a relevant disadvantage, in all the previous cases, the metric used to stop processing is fixed, resulting in a model that is not differentiable and therefore must be trained in stages.

In contrast to previous works, we propose a fully differentiable alternative to achieve adaptive computation in Transformers based architectures. Our method takes inspiration from DACT (Eyzaguirre & Soto, 2020), a technique proposed for visual reasoning tasks that is capable of adapting the number of reasoning steps in a recurrent model. Unlike previous works, instead of relying on heuristics of knowledge from outside experts, during training DACT-BERT learns directly when to halt. This feature allows it to tailor the computation used based on the complexity of the input. Furthermore, as all this is differentiable, we can train the complete model, adapting the weights of the transformer blocks in the process.

2.2 Interpretable transformers

Due to the current widespread use of Transformers, attempting to understand the factors that lead to a prediction is currently an active area of research. Traditionally, attention weights have been used as a naive way to generate explanations, but this is usually only possible in the context of small Transformer models (Stahlberg et al., 2018; Villa et al., 2020). In the case of large scale models, several studies have shown that many of the heads of attention are redundant (Michel et al., 2019; Clark et al., 2019; Kovaleva et al., 2019). Furthermore, not all predictions require the same number of Transformer layers (Lee et al., 2019). In general, the overparameterization of models makes it difficult to identify meaningful relationships between the input tokens and the predictions. This complexity is the source of the intuition that deeper models are harder to explain, an idea that recently received theoretical support (Barceló et al., 2020).

Following the previous intuition, DACT-BERT improves the transparency of BERT models by adaptively reducing the sequence of blocks needed to output a prediction. Furthermore, DACT-BERT improves interpretability by exposing a learned confidence score after each unit of computation (Transformer blocks). This also enables to access the causes for the confidence itself through the inclusion of methods such as Integrated gradients (Sundararajan et al., 2017) and Layer Conductance (Dhamdhere et al., 2018) thanks to its differentiable nature.

3 DACT-BERT: Differentiable Adaptive Computation Time for BERT

Adaptive methods work by signaling the amount of computation necessary to complete a given task. In this work, our signaling module, DACT, produces a prediction based on intermediate results and the confidence it has on them. This mechanism can then be used to stop an execution when stable results are obtained, reducing the total number of steps necessary for a given prediction. The original formulation of DACT (Eyzaguirre & Soto, 2020) applies this module to recurrent models. In our case, we adapt the formulation to the case of Transformer based architectures, namely BERT.

As shown in Figure 1, DACT-BERT introduces additional linear layers after each computational unit, similar to the *off-ramps* in (Xin et al., 2020b) or the student classifiers in the work of Liu et al. (2020). As in both these cases, we define the discrete unit of computation to be a single BERT Transformer block.

As Figure 1 shows, the n-th DACT module computes both an output vector y_n with the predicted class probabilities, as well as an accompanying scalar confidence score (or halting value) h_n. Following Devlin et al. (2019), both, y_n and h_n, are estimated by using the classification token ($[CLS]$) that is included in BERT as part of the output representation of each layer. During training all the output vectors and halting values are combined to obtain the final predicted probabilities following

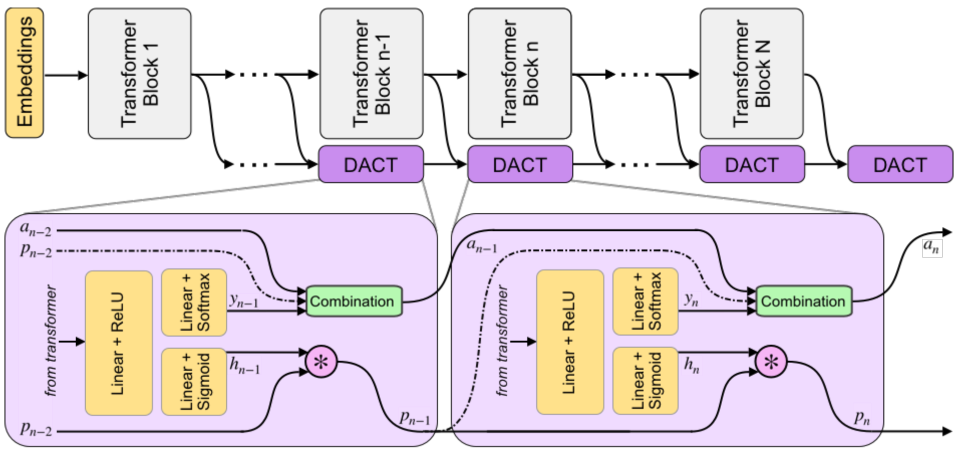


Figure 1: DACT-BERT adds an additional classification layer after each Transformer block, along with a sigmoidal *confidence function*. DACT-BERT combines the Transformer hidden state and the outputs and confidences of all earlier layers into an accumulated answer a_n. Later, during inference, the model is halted once $a_n \approx a_N$.

an expression that can be rewritten as the weighted average of all intermediate outputs y_n multiplied by a function of the confidences of earlier blocks. Then, during inference, the confidence scores can be used to reduce computation.

Auxiliary accumulator variables a_n are used to build the full output of the model (a_N with $N = 12$ for BERT-base and RoBERTA-base). This is performed by accumulating the intermediate outputs y_n:

$$a_n = \begin{cases} \overrightarrow{0} & \text{if} \quad n = 0 \\ y_n p_{n-1} + a_{n-1}(1 - p_{n-1}) & \text{otherwise} \end{cases} \tag{1}$$

where p_n is a monotonically decreasing function of the confidence scores defined as follows:

$$p_n = \prod_{i=1}^{n} h_i = h_n p_{n-1} \tag{2}$$

The model is trained to reduce the classification loss of the final output along with a regularizer that induces a bias towards reduced computation. In contrast to the regularizer used in Eyzaguirre & Soto (2020), we use the following:

$$\hat{L}(\boldsymbol{x}, \boldsymbol{y}) = L(\boldsymbol{x}, \boldsymbol{y}) + \tau \sum_{i=1}^{n} h_i \tag{3}$$

where τ is a hyper-parameter used to moderate the trade-off between complexity and error. We find empirically that this regularizer helps training convergence and further binarizes the halting probabilities.

Notably, the whole formulation is end-to-end differentiable. This allows us to fine-tune the weights of the underlying backbone (*i.e.* the Transformer blocks and embedding layers) using a joint optimization with the process that trains the intermediate classifiers. This stands in contrast to all existing methods for dynamic Transformers, where training is done in two stages first pre-training the backbone (and not the intermediate classifiers) and later freezing the backbone and only modifying the weights of the classifiers.

3.1 Generality

Note that the formulation of DACT is highly general and any differentiable function of the intermediate hidden representations can be used to obtain the confidence scores. In particular, in this work we choose to learn the *confidence function* and therefore use a perceptron as a universal function approximator, followed by a sigmoid activation function.

We highlight that both DeeBERT and FastBERT can be expressed and trained with our model. For DeeBERT this can be achieved by hamstringing DACT and training it in three stages. The first two stages consist of the same training regime as DeeBERT, first training only the transformer blocks and the final classification layer, and subsequently freezing the transformer blocks and training the intermediate classifiers independently. An appropriate *confidence function* to force the model to adapt based on entropy is a simple logistic regression that receives the output entropy and consists only of a bias to encode the entropy threshold, and a single weight to scale the layer response. Training at this point will yield different learnt entropy thresholds for each layer (which is positive) but, in order to fully represent DeeBERT, we need to further handicap the model by tying the biases in all the *confidence functions*. FastBERT can be obtained similarly by simply replacing the second training stage with the distillation process described by Liu et al. (2020).

In consequence, DACT-BERT can be seen as a generalization of existing dynamic Transformer architectures, allowing us to train these in an end-to-end fashion while eliminating the need for manual tuning of the entropy hyperparameter. In this sense, in our experiments, we note that the training of only the *confidence functions* results in a optimization problem that produces highly stable solutions, suggesting that SGD is able to find suitable optimum entropy thresholds [1].

3.2 Dynamic computation at inference

By construction the DACT algorithm allows us to calculate upper and lower bounds for each of the output classes after any computation step (*ie.* transformer block). During inference, execution halts once the predicted probabilities for the topmost class are shown to remain higher than that of the *runner-up* class (and by extension, of any other class). That is, the model is stopped from running additional blocks once it is found that doing so will not change the class with maximum probability in the final output because the difference between the top class and the rest is insurmountable.

Mathematically, we prove that the difference is insurmountable by comparing the lower bound for the probability of the top class predicted after n blocks ($\Pr(c^*, n)$) with the upper bound of the probability of the runner-up class ($\Pr(c^{ru}, n)$). Therefore, the *halting condition* remains the same as in the original DACT formulation (Eyzaguirre & Soto, 2020):

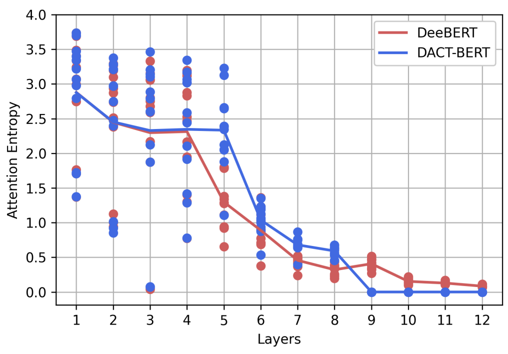
$$\Pr(c^*, n)(1 - p_n)^d \geq \Pr(c^{ru}, n) + p_n d \tag{4}$$

3.3 Training

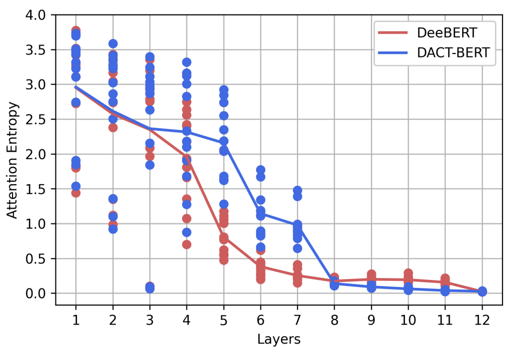
	S	MRPC		QNLI		RTE	
		F1	Saved	Acc	Saved	Acc	Saved
		BERT-base					
Base	-	88.2%	0.0%	91.0%	0.0%	69.9%	0.0%
	0.25	87.3%	30.8%	89.1%	40.5%	69.8%	5.3%
DeeBERT	0.50	85.2%	49.1%	85.5%	58.2%	68.1%	21.2%
	0.75	79.9%	91.7%	59.6%	91.7%	52.0%	91.7%
DACT BERT	-	85.4%	48.9%	87.5%	46.7%	62.7%	50.0%
		RoBERTa-base					
Base	-	90.4%	0.0%	92.4%	0.0%	67.5%	0.0%
	0.25	73.4%	0.0%	90.2%	33.3%	66.9%	36.1%
DeeBERT	0.50	73.4%	0.0%	88.0%	42.8%	66.0%	39.6%
	0.75	0.0%	91.7%	59.5%	91.7%	49.8%	91.7%
DACT BERT	-	88.9%	51.7%	91.9%	44.1%	54.5%	63.1%

Table 1: Results of the performance comparison between DACT-BERT and DeeBERT methods in different tasks of the GLUE benchmark. BERT and RoBERTa vanilla were included as base models. The column S indicates the entropy value used to reduce the computation on the DeeBERT model. The column *Saved* indicates the percentage of blocks saved by using the corresponding method.

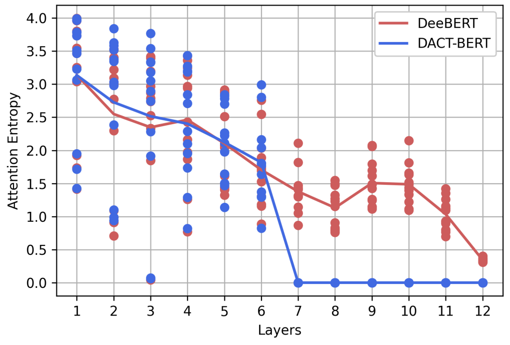
[1] for a small enough learning rate.



(a) MRPC



(b) QNLI



(c) RTE

Figure 2: Attention entropy distribution per layer in the backbone for DACT-BERT (blue) and DeeBERT (red) for three different GLUE tasks. Each point represent the entropy for one attention head in each layer and the line shows the mean entropy for all the attentions in a given layer.

The training of the module follows a two step process. First the underlying Transformer model must be tuned to the relevant task. This ensures a good starting point onto which the DACT module can then be adapted to and speeds up convergence.

This is followed by a second fine-tuning phase where both the DACT module as well as the underlying Transformer are jointly trained for the relevant task. This latter training phase, not only makes it possible to use the module to work adaptatively, but also modifies the Transformer layers making them more suited to work together, while also generating new representations in the Transformer which are useful for the new lower computation scenario.

4 Results

4.1 Experimental Setup

We tested our method using BERT and RoBERTa, evaluating both models on three different tasks from the GLUE benchmark (Wang et al., 2018). Our model was developed using PyTorch (Paszke et al., 2017) on top of the implementation released by Xin et al. (2020a) as well as the HuggingFace Transformers library (Wolf et al., 2019) [2].

In our experiments, we fine-tune the backbone model for the GLUE tasks using the default values of the hyper-parameters. For the second stage we use $\tau = 5 \cdot 10^{-3}$ for the regularizer to balance computation time with precision. Other values for this parameter result in different tradeoffs between computation and accuracy.

[2]Code will be released upon publication.

To analyze the results, DeeBERT (Xin et al., 2020b) was chosen as the representative baseline for entropy-based dynamic Transformers since the trained weights were published in the HuggingFace model hub[3]. Three entropy thresholds S were selected to test this model for a fair quality-efficiency comparison. The same results are also computed for both BERT and RoBERTa models for comparison purpose.

4.2 Computational efficiency

We first tested our method to understand its accuracy to computation characteristics. We show both metrics for both values as the efficiency value almost always comes at the expense of a performance drop on the task in question. For these experiments, efficiency was measured as the percentage of Transformer layers saved out of the total number of layers executed without an efficiency mechanism (equivalent to the total number of layers in their static counterparts). The specific metrics for performance are those suggested in the GLUE paper (Wang et al., 2018) for each task.

The results of this evaluation can be seen in Table 1. Both results for the original BERT and RoBERTa performance were included. Both results for the original BERT and RoBERTa performance were included. It can be observed that both DACT-BERT and DeeBERT models perform similarly when using comparable amounts of computation. We interpret this result as an informal proof of the efficacy of using entropy to quantify model uncertainty, and as validation of out learnt *confidence function* and the training process that produced it.

Importantly, because our model learns to regulate itself, it shows a remarkable stability in the amount of computation saved. By contrast, DeeBERT proves to be extremely sensible to the chosen value for the entropy hyperparameter, exhibiting important fluctuations in both computation and performance indicators for small changes in its value (see RTE column in Table 1). Additionally, we observe that the amount of computation saved by DACT remains close to 50% regardless of the task, which again contrasts with DeeBERT, as entropy-based models require a painstaking calibration process in order to find the entropy threshold that will lead to a particular efficiency level. This robustness seems to come from training the efficiency mechanism instead of relying on a somewhat arbitrary quantity for its control.

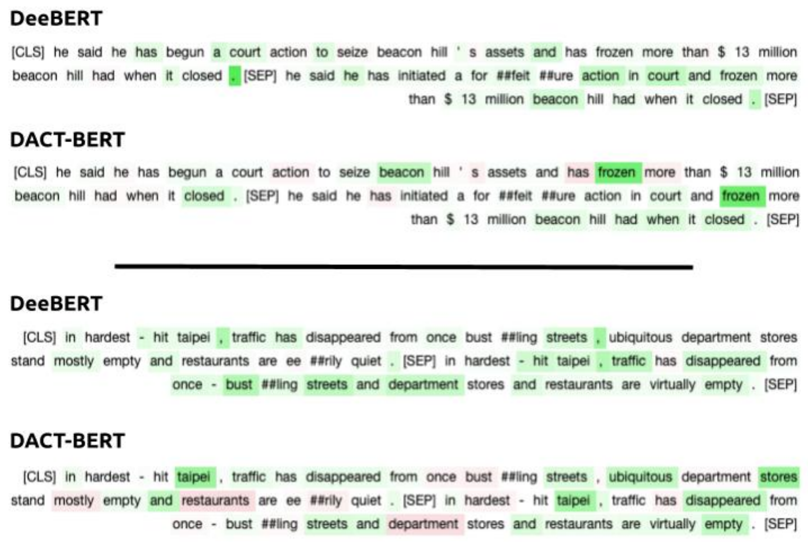
Additional advantages of our model can be observed in Figure 2 which shows the average entropy of each head's attention distribution for DACT-BERT and DeeBERT (as sugested by Rogers et al. (2020)). First, it can be easily seen that our approach uses less layers (exact frequencies are shown in Figure 5). That is, even when using on average the same number of layers (as is the case in Figure 2a), DACT-BERT completely disregards the outputs from the last blocks, enabling us to prune whole layers without changing the model accuracy for reduced model size. On the one hand, we explain this difference by noting that the entropy will remain high throughout the whole model for the case of difficult questions as it will be uncertain about the answer. On the other hand, any layer in DACT-BERT is capable of quitting computation if it believes future layers cannot answer with more certainty than its own (regardless of how certain the model actually is). Figure 2c support this hypothesis, showing that difficult tasks, where the model performs worse, final layers are used more frequently.

4.3 Interpretability

Our model shows an increased interpretability when compared to both baselines in two different ways. First, we increase interpretability by reducing the number of layers and therefore parameters that contribute to an answer. We hypothesize that this will lead to a more transparent model, as Barceló et al. (2020) proved that interpretability decreases with the number of layers.

To test this hypothesis, we take advantage of the Integrated Gradients technique (Sundararajan et al., 2017) to explore the attributions the input tokens have over the final prediction both for our model as well as our baseline. Two samples from the MRPC dataset explored in this way are shown in Figure 3. Here, it can be seen that DACT-BERT shows a greater amount of attribution towards entities which are key to infer that both phrases are semantically equal, while the baseline shows more uninformative flatter attributions.

[3]`https://huggingface.co/ji-xin`


DeeBERT

[CLS] he said he has begun a court action to seize beacon hill ' s assets and has frozen more than $ 13 million beacon hill had when it closed . [SEP] he said he has initiated a for ##feit ##ure action in court and frozen more than $ 13 million beacon hill had when it closed . [SEP]

DACT-BERT

[CLS] he said he has begun a court action to seize beacon hill ' s assets and has frozen more than $ 13 million beacon hill had when it closed . [SEP] he said he has initiated a for ##feit ##ure action in court and frozen more than $ 13 million beacon hill had when it closed . [SEP]

DeeBERT

[CLS] in hardest - hit taipei , traffic has disappeared from once bust ##ling streets , ubiquitous department stores stand mostly empty and restaurants are ee ##rily quiet . [SEP] in hardest - hit taipei , traffic has disappeared from once - bust ##ling streets and department stores and restaurants are virtually empty . [SEP]

DACT-BERT

[CLS] in hardest - hit taipei , traffic has disappeared from once bust ##ling streets , ubiquitous department stores stand mostly empty and restaurants are ee ##rily quiet . [SEP] in hardest - hit taipei , traffic has disappeared from once - bust ##ling streets and department stores and restaurants are virtually empty . [SEP]

Figure 3: Output prediction attribution to each input token. Two samples from the MRPC task are shown with their attributions obtained both for DeeBERT and DACT-BERT. The color intensity shows the degree on how much each token contributes to the final output. Green is used for tokens that have positive attribution to the output, while red means tokens with negative attribution to the output.

A second source of interpretability provided by DACT-BERT comes from adding a fully differentiable module which manages the number of steps to be taken. In doing this, there is an additional source of insight that can be explored to understand the inner working of the model. In particular to understand which layers and neurons are important for the halting prediction. This gives us further insights on what is the model looking at in order to make the prediction decision. Figure 4 illustrates this idea by showing an example of each layer's halting value, with the corresponding attributions of each token to its value. The closest analogue is to calculate the attributions of the entropy in DeeBERT which appeared to be meaningless in our experiments.

5 Conclusions and Future Work

This work introduced DACT-BERT, a model capable of both leveraging modern pre-trained Transformer architectures and differentiable adaptive computation. By using this mechanism to allow the model to adapt the number of blocks needs to execute at inference time, we find that the model makes predictions that are more computationally efficient and interpretable than the base architecture. Moreover, we proved that this approach is competitive with current state-of-the-art dynamic Transformers when using comparable amounts of computation, while also retaining a series of other advantages such as being more robust, end-to-end trainable, more compressible (by eliminating unused blocks) and interpretable.

Furthermore, we expect future work will benefit from the proposed architecture. While this model laid the groundwork in making BERT more interpretable, we foresee upcoming investigations in both understanding BERT and modifying BERT to make it more transparent will benefit from the techniques presented here. Additionally, there is still a lot of performance to be gained by using DACT-BERT, either by tuning hyperparameters for the GLUE tasks shown here (instead of using the same for every task), or training on other datasets that will benefit more from its end-to-end differentiability and the possibility of fine-tuning Transformer weights.

Finally, we hope that approaching DACT-BERT as a generalization and extension of other dynamic Transformer algorithms will lead to further research on alternative training regimes and *confidence functions* that will continue to increase interpretability and reduce computation.

REFERENCES

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dÁlché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 690–701. Curran Associates, Inc., 2019. URL `http://papers.nips.cc/paper/8358-deep-equilibrium-models.pdf`.

Pablo Barceló, Mikael Monet, Jorge Pérez, and Bernardo Subercaseaux. Model interpretability through the lens of computational complexity. In *Advances in Neural Information Processing Systems 33*. Curran Associates, Inc., 2020.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? An analysis of BERT's attention. *CoRR*, abs/1906.04341, 2019.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HyzdRiR9Y7`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://www.aclweb.org/anthology/N19-1423`.

Kedar Dhamdhere, Mukund Sundararajan, and Qiqi Yan. How important is a neuron? *arXiv preprint arXiv:1805.12233*, 2018.

Cristobal Eyzaguirre and Alvaro Soto. Differentiable adaptive computation time for visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12817–12825, 2020.

L.H. Gilpin, D. Bau, B. Yuan, A. Bajwa, M. Specter, and L. Kagal. Explaining explanations: An overview of interpretability of machine learning. In *Int. Conf. on Data Science and Advanced Analytics (DSAA)*, 2018.

B. Goodman and S. Flaxman. EU regulations on algorithmic decision-making and a "right to explanation". *ArXiv*, abs/1606.08813, 2016.

A. Graves. Adaptive computation time for recurrent neural networks. *ArXiv*, abs/1603.08983, 2016.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks, 2017.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tiny{bert}: Distilling {bert} for natural language understanding, 2020. URL `https://openreview.net/forum?id=rJx0Q6EFPB`.

Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *International Conference on Machine Learning*, pp. 3301–3310. PMLR, 2019.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4365–4374, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1445. URL `https://www.aclweb.org/anthology/D19-1445`.

Jaejun Lee, Raphael Tang, and Jimmy Lin. What would elsa do? freezing layers during transformer fine-tuning, 2019.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. FastBERT: a self-distilling BERT with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6035–6044, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.537. URL `https://www.aclweb.org/anthology/2020.acl-main.537`.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 14014–14024. Curran Associates, Inc., 2019. URL `http://papers.nips.cc/paper/9551-are-sixteen-heads-really-better-than-one.pdf`.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

A. Rogers, O. Kovaleva, and A. Rumshisky. A primer in BERTology: What we know about how BERT works. *ArXiv*, abs/2002.12327, 2020.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

R. Schwartz, J. Dodge, N.A. Smith, and O. Etzioni. Green ai. *ArXiv*, abs/1907.10597, 2019.

Felix Stahlberg, Danielle Saunders, and Bill Byrne. An operation sequence model for explainable neural machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 175–186, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5420. URL `https://www.aclweb.org/anthology/W18-5420`.

E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in NLP. *ArXiv*, abs/1906.02243, 2019.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for BERT model compression. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4323–4332, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1441. URL `https://www.aclweb.org/anthology/D19-1441`.

M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *ICML*, 2017.

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

Andrés Villa, Vladimir Araujo, Francisca Cattan, and Denis Parra. Interpretable contextual team-aware item recommendation: Application in multiplayer online battle arena games. In *Fourteenth ACM Conference on Recommender Systems*, RecSys '20, pp. 503–508, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375832. doi: 10.1145/3383313.3412211. URL `https://doi.org/10.1145/3383313.3412211`.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL `https://www.aclweb.org/anthology/W18-5446`.

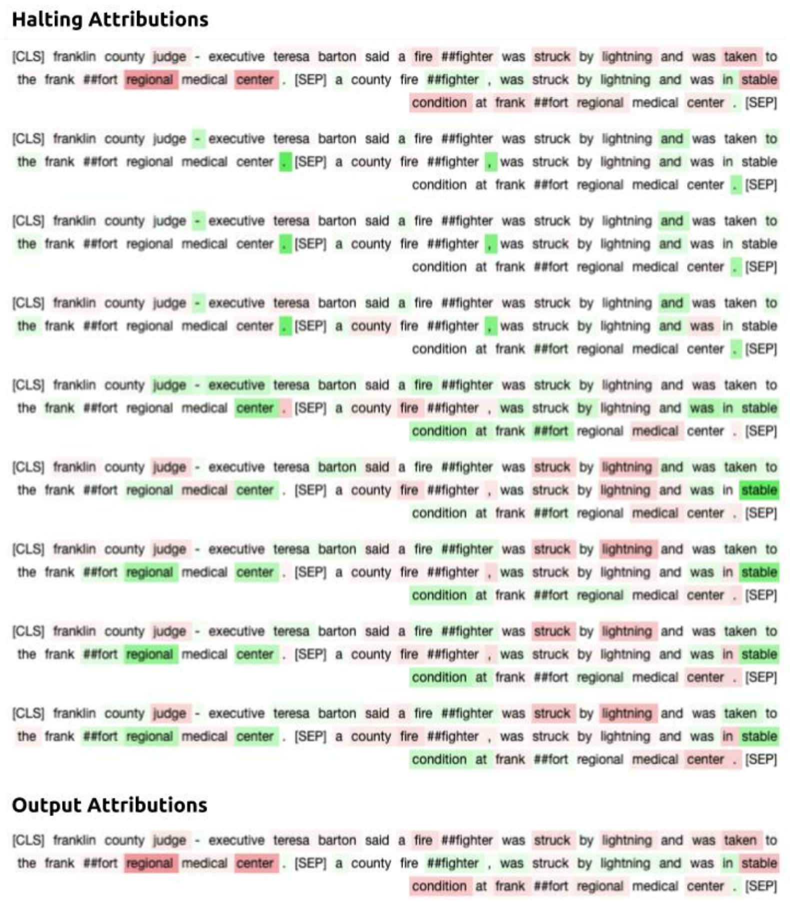
Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2246–2251, Online, July 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.204. URL `https://www.aclweb.org/anthology/2020.acl-main.204`.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2246–2251, Online, July 2020b. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.204. URL `https://www.aclweb.org/anthology/2020.acl-main.204`.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: Fast and robust inference with early exit, 2020.

A HALTING AND OUTPUT ATTRIBUTIONS



Halting Attributions

[CLS] franklin county judge - executive teresa barton said a fire ##fighter was struck by lightning and was taken to the frank ##fort regional medical center . [SEP] a county fire ##fighter , was struck by lightning and was in stable condition at frank ##fort regional medical center . [SEP]

[CLS] franklin county judge - executive teresa barton said a fire ##fighter was struck by lightning and was taken to the frank ##fort regional medical center . [SEP] a county fire ##fighter , was struck by lightning and was in stable condition at frank ##fort regional medical center . [SEP]

[CLS] franklin county judge - executive teresa barton said a fire ##fighter was struck by lightning and was taken to the frank ##fort regional medical center . [SEP] a county fire ##fighter , was struck by lightning and was in stable condition at frank ##fort regional medical center . [SEP]

[CLS] franklin county judge - executive teresa barton said a fire ##fighter was struck by lightning and was taken to the frank ##fort regional medical center . [SEP] a county fire ##fighter , was struck by lightning and was in stable condition at frank ##fort regional medical center . [SEP]

[CLS] franklin county judge - executive teresa barton said a fire ##fighter was struck by lightning and was taken to the frank ##fort regional medical center . [SEP] a county fire ##fighter , was struck by lightning and was in stable condition at frank ##fort regional medical center . [SEP]

[CLS] franklin county judge - executive teresa barton said a fire ##fighter was struck by lightning and was taken to the frank ##fort regional medical center . [SEP] a county fire ##fighter , was struck by lightning and was in stable condition at frank ##fort regional medical center . [SEP]

[CLS] franklin county judge - executive teresa barton said a fire ##fighter was struck by lightning and was taken to the frank ##fort regional medical center . [SEP] a county fire ##fighter , was struck by lightning and was in stable condition at frank ##fort regional medical center . [SEP]

[CLS] franklin county judge - executive teresa barton said a fire ##fighter was struck by lightning and was taken to the frank ##fort regional medical center . [SEP] a county fire ##fighter , was struck by lightning and was in stable condition at frank ##fort regional medical center . [SEP]

[CLS] franklin county judge - executive teresa barton said a fire ##fighter was struck by lightning and was taken to the frank ##fort regional medical center . [SEP] a county fire ##fighter , was struck by lightning and was in stable condition at frank ##fort regional medical center . [SEP]

Output Attributions

[CLS] franklin county judge - executive teresa barton said a fire ##fighter was struck by lightning and was taken to the frank ##fort regional medical center . [SEP] a county fire ##fighter , was struck by lightning and was in stable condition at frank ##fort regional medical center . [SEP]

Figure 4: Halting (top) and output (bottom) attribution to each input token. For one sample from the MRPC task. The tokens attributions towards the halting value are computed for every halting neuron, after each transformer block. For both, the color intensity shows the degree on how much each token contributes to the final output. Green is used for tokens that have positive attribution to the output, while red means on tokens with negative attribution to the halt or output.

B Layer Frequency

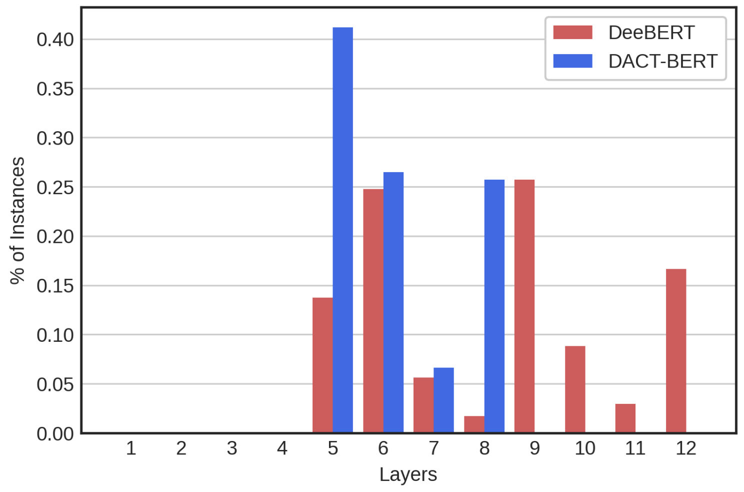


Figure 5: The number of times DACT-BERT (blue) and DeeBERT (red) quit computation at a specific layer in the MRPC task.