🤔 PALBERT: Teaching ALBERT to Ponder.

Anonymous ACL submission

Abstract

Currently, pre-trained models can be considered the default choice for a wide range of NLP tasks. Despite their SoTA results, there is practical evidence that these models may require a different number of computing layers for different input sequences, since evaluating all layers leads to overconfidence on wrong predictions (namely overthinking). This problem can potentially be solved by implementing adaptive computation time approaches, which were first designed to improve inference speed.

Recently proposed PonderNet may be a promising solution for performing an early exit by treating the exit layer's index as a latent variable. However, the originally proposed exit criterion, relying on sampling from trained posterior distribution on the probability of exiting from i-th layer, introduces major variance in model outputs, significantly reducing the resulting model's performance.

In this paper, we propose Ponder ALBERT (PALBERT) – an improvement to PonderNet with a novel deterministic Q-exit criterion and a revisited model architecture. We compared PALBERT with recent methods for performing an early exit. We observed that the proposed changes can be considered significant improvements on the original PonderNet architecture and outperform PABEE on a wide range of GLUE tasks. In addition, we also performed an in-depth ablation study of the proposed architecture to further understand Lambda layers and their performance.

1 Introduction

These days, fine-tuning pre-trained models on downstream tasks became a de facto standard technique for training NLP models. One model that is widely used in real-world applications is ALBERT (Lan et al., 2020), which is based on the Transformer architecture (Vaswani et al., 2017) with shared layers (i.e., the same layer is evaluated several times to provide an output).

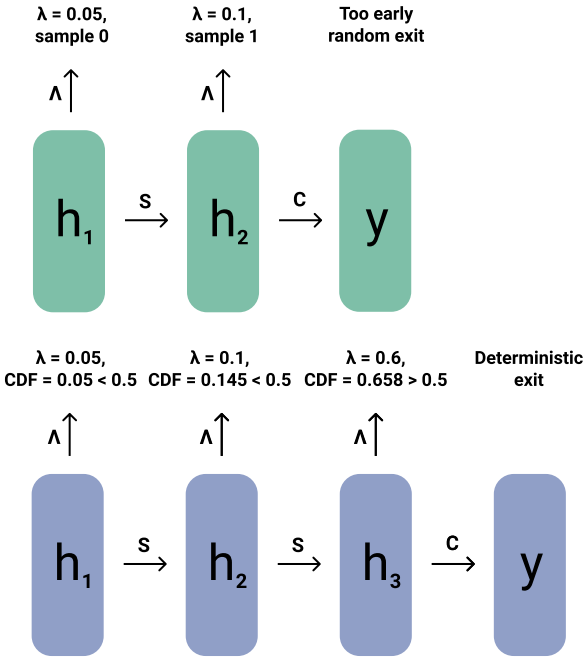

Figure 1: A comparison of the original sampling exit criterion of PonderNet (on the top) and the proposed Q-exit criterion (on the bottom). PonderNet performs sampling from the Bernoulli distribution obtained from the Lambda layer at each step, possibly exiting a model too early or too late. For Q-exit, we evaluate the Cumulative distribution function (CDF) of the probability of exiting at layer i. Once CDF becomes greater than the threshold value (0.5 in this example), we perform an early exit. With such a deterministic criterion, we can perform an early exit from a model more robustly without introducing variance in the exit layer's index during inference.

While ALBERT-Base evaluates the Transformer block 12 times, layer sharing makes it possible to evaluate it an arbitrary number of times. Zhou et al. (2020) showed that running ALBERT-Base block for a fixed number of times (10) could increase the accuracy of the fine-tuned model on specific tasks (e.g., MRPC). This phenomenon is called overthinking. Because of this fact, **making models perform an early exit is not only done to in-**

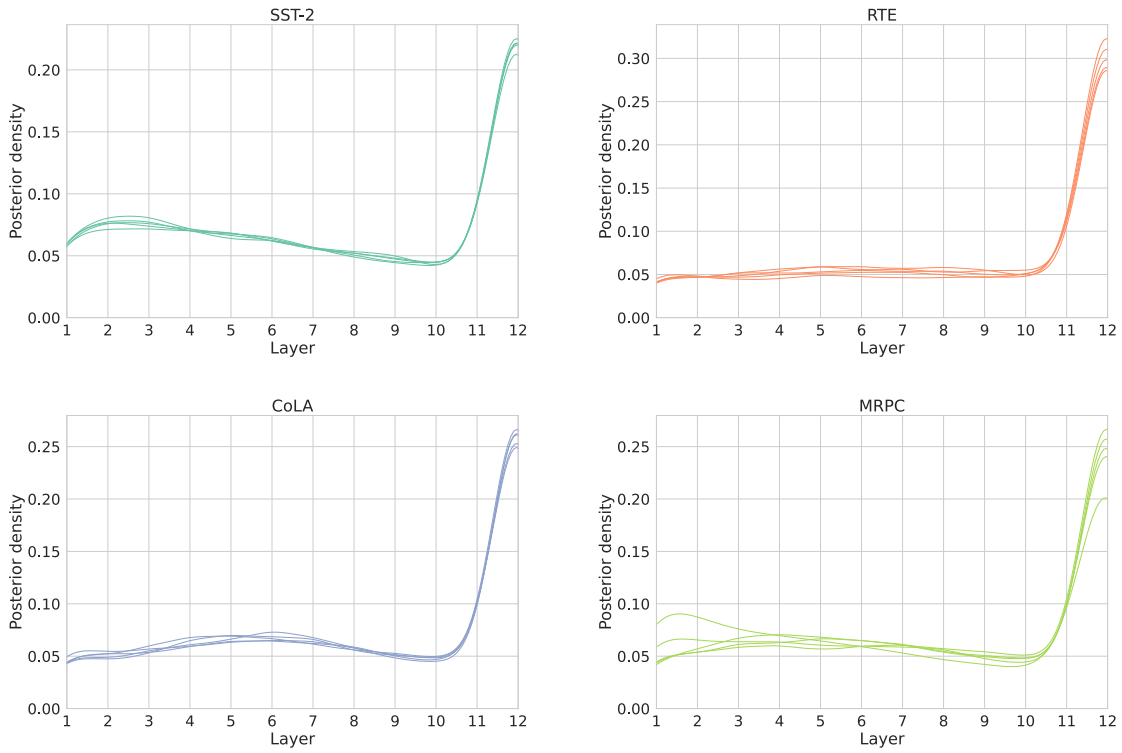

Figure 2: An estimation of $E_{x\sim D}\Big[p(i|x)\Big]$, where $p(i|x)$ is a trained posterior probability of exiting from layer i of PALBERT models across different tasks, and D is the distribution of the training dataset. We took 5 models trained on these tasks and sampled exit layer indices for the training dataset's inputs. We smoothed the obtained probabilities for visibility.

crease inference speed but also to make them more accurate. A recent PABEE (Zhou et al., 2020) solution was designed to overcome this issue by performing an early exit based on the consensus between different classifier heads from different layers. The model stops evaluating when several classifiers in a row produce the same result.

An orthogonal way to perform an early exit from a model is PonderNet (Banino et al., 2021) – a variational approach that treats the exit layer's index as a latent variable. By maximizing the lower bound of the likelihood of the training data, PonderNet trains a model which can predict whether it is necessary to exit from a specific layer during evaluation. However, Banino et al. (2021) proposed to sample from the trained posterior distribution of exiting from each layer during inference, which leads to major variance in model outputs.

This paper proposes **Ponder ALBERT (PALBERT)** – an improvement to PonderNet adapted for ALBERT fine-tuning. Instead of performing an early exit by sampling from the trained posterior distribution during evaluation, we used a novel zero-variance exit criterion, namely **Q-exit**, which evaluates the CDF of the exit layer's probability distribution and perform a deterministic early exit. We also revisited the architectural choices of Lambda layers used to predict the probability of exiting from the current layer in order to make them aware of dynamics in hidden states across previous layers and the number of currently running layers.

We experimented with PALBERT on the GLUE Benchmark datasets (Wang et al., 2018). The ablation study showed that PALBERT produced significantly better results than the original PonderNet architecture adapted for ALBERT fine-tuning. Furthermore, PALBERT outperformed PABEE and is comparable to plain ALBERT fine-tuning, while also exceeding it in speeds. We also analyzed the trained model and provided insights on further improvement of the variational approach for early exiting.

2 Related Work

Most of the approaches used to perform an early exit from a model are based on the probability distribution of predictions: BranchyNet (Teerapittayanon et al., 2016), FastBERT (Liu et al., 2020),

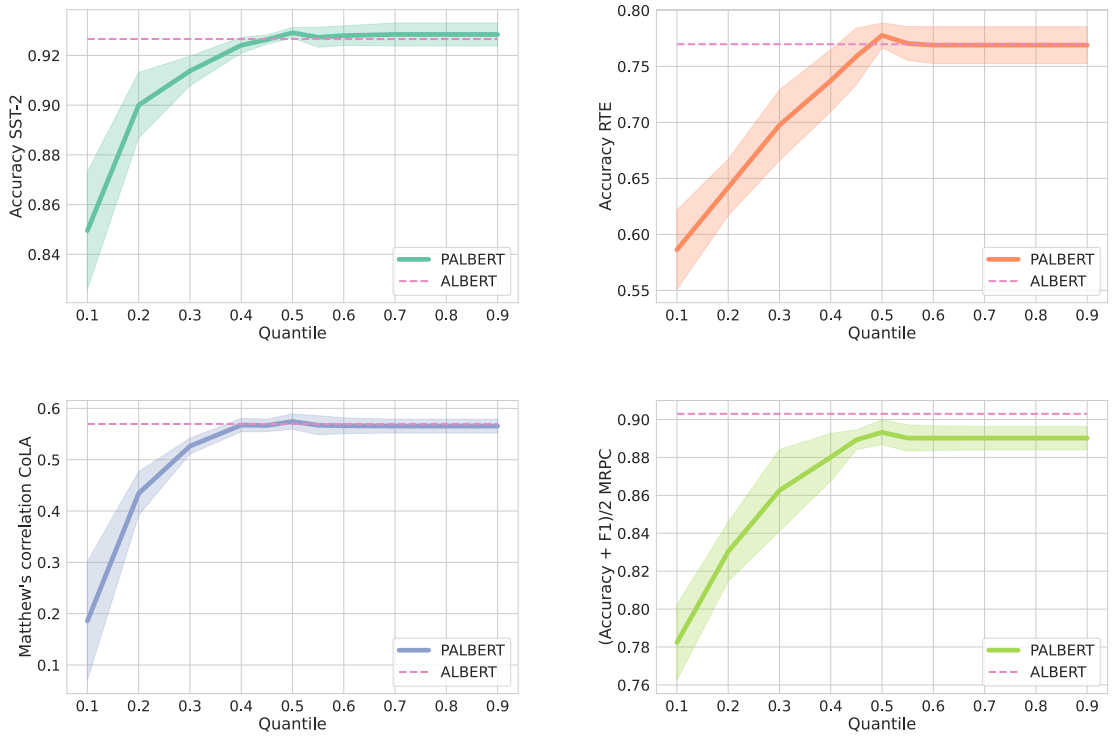

Figure 3: PALBERT score dependency on the Q-exit threshold. We report the mean and std values of task metrics across 5 trained models. See section 4.2 for more details

DeeBERT (Xin et al., 2020), which can be seen as an entropy criterion. However, there is strong practical evidence that classification models' overthinking causes a reduction in predictions' entropy, making these methods difficult to use (Zhou et al., 2020). Furthermore, it is unclear how to adapt entropy methods for regression tasks (Zhou et al., 2020).

Zhou et al. (2020) proposed PABEE – a method to perform an early exit based on several classifiers from the different levels of a model. Once several classifiers in a row (the number of these classifiers is determined by the patience hyperparameter t) produce the same result, we can perform an early exit. LeeBERT (Zhu, 2021) also uses the idea of a consensus-based exiting strategy augmenting the training algorithm with the self-distillation technique and cross-level optimization. Self-distillation is orthogonal to the early exit approach and can be combined with PALBERT. Because of this, we did not include LeeBERT in our experiments and only used PABEE as a consensus-based method.

An alternative way to perform an early exit is the Ponder architecture (Banino et al., 2021), which uses auxiliary Lambda layers to predict whether a model should exit from a specific layer during the runtime. Inputs to Lambda layers used in PonderNet are hidden states from the current layer of a model. PonderNet can be seen as a model with the latent variable in the face of the exit layer index, which is trained by maximizing the lower bound of the marginalized likelihood of the data.

During inference, PonderNet authors proposed to sample from the trained posterior distribution of exit layer probabilities. However, this exit criterion can lead to uncertainty in outputs for the same input. Even if the Lambda layer produced probability equal to 0.1 of exiting from the first layer, we could still exit a model too early in one of ten, cases even though the probability was small. We also hypothesize that predicting exiting from a layer based entirely on a single hidden state could be sub-optimal since performing early exit could also depend on the dynamics in hidden states across layers (i.e., Lambda layer should know how hidden states change during the evaluation).

3 Ponder ALBERT

The usual ALBERT evaluation can be defined as a computation of n hidden states $h_i = S(h_{i-1})$ from the input embeddings h_0 of an input sequence x, where $i \in [1; n]$. Once h_n is obtained, it is passed

to a classifier block $C(h_n)$ to get the parameters of an output distribution $p(y|x)$. A common way to fine-tune this architecture on downstream tasks is to initialize the embeddings and the S layer by using ALBERT (pre-trained on Masked Language Modelling) while initializing C randomly and then optimizing all parameters by maximizing the likelihood of the training data.

While plain ALBERT performs a fixed number of computational steps, it is possible to perform an arbitrary number of evaluations of the layer S. Banino et al. (2021) proposed to extend each Transformer layer with a shared Lambda layer. More precisely, for each layer i, after S outputs a new h_i, it is then passed to the classifier and Lambda layers to get parameters $C(h_i)$ of output distributions $p(y|x,i)$ and the probability of exiting from the i-th layer $\lambda_i = \Lambda(h_i)$, which induces a generalized geometric distribution on probability of exiting from layer i equal to

$$p(i|x) = \lambda_i \prod_{j=1}^{i-1}(1-\lambda_j). \tag{1}$$

Then, having the probability distribution from each layer $p(y|x,i)$, the parameters of the model are optimized to maximize

$$\begin{aligned} L(x,y) = E_{i\sim p(i|x)}\Big[p(y|x,i)\Big] - \\ -\beta KL\Big(p(\cdot|x)||p(\cdot|\lambda)\Big) \leq p(y|x) \end{aligned} \tag{2}$$

Here, $p(\cdot|\lambda)$ is a prior distribution of exiting from each layer, parametrized by the hyperparameter λ, and $E_{i\sim p(i|x)}\Big[p(y|x,i)\Big]$ is evaluated analytically by averaging likelihoods from different layers with posterior exit probabilities. If we treat the exit layer index as a latent variable, then optimizing L from the Equation 2 could be seen as maximizing the lower bound of marginalized likelihood $p(y|x)$ (Kingma and Welling, 2014).

Note that the probability of exiting from the last layer n is normalized as $p(n|\lambda) = 1 - \sum_{i=1}^{n-1} p(i|\lambda)$ in order to make $p(i|\lambda)$ sum into 1 with a finite number of steps. The same is true for $p(i|x)$.

3.1 Exit Criterion

During inference, Banino et al. (2021) proposed to sample the exit layer index from $p(i|x)$ (i.e., by sampling iteratively from a Bernoulli distribution with parameter λ_i). While a sampling-based exit criterion correlates with the variational view of PonderNet's training objective (it can be seen as performing a single sample Monte-Carlo estimation of $E_{i\sim p(\cdot|x)}\Big[p(y|x,i)\Big]$); such estimation has major variance, which introduces the randomness in the inference process of PonderNet (see Figure 2).

To overcome the issue of randomness, we propose **Q-exit**[1]: a novel deterministic criterion of performing early exit, which we used for PALBERT. Instead of sampling from the distribution $p(i|x)$ during inference, we evaluate its CDF by accumulating $p(i|x)$ from each layer. Once the CDF is greater than the threshold hyperparameter q, we perform an early exit. See Figure 1 for a schematic comparison of the sampling criterion with Q-exit. Threshold q can be seen as a trade-off between underthinking and overthinking. Therefore, q should be selected during the validation of the trained model in order to choose the best-performing value.

Based on our experiments, we found that the proposed criterion produced significantly better accuracy on various tasks compared to the original sampling criterion (see Sections 4.1, 4.4), while also being more practical than the original sampling criterion.

3.2 Lambda Layer Architecture

While the original PonderNet used a single layer MLP to obtain logit of exiting probability, we hypothesize that making the Lambda layer understand the dynamics of changing ALBERT hidden states is crucial for achieving good performance. To do so, instead of passing a single hidden state h_i from the i-th layer in Λ, we concatenate it with h_{i-1}. I.e., for PALBERT, we evaluate the probability of exiting from i-th layer as

$$\lambda_i = \Lambda([h_i, h_{i-1}]). \tag{3}$$

We used a 3 layer MLP with $\tanh$ activation for the Lambda layer to operate with more complex input. Based on the ablation study, we observed that increasing the capacity improves the accuracy of the trained model (See section 4.1). We also found it beneficial to fine-tune the Lambda layer with a different learning rate than all other parameters.

[1] **Q**-exit stands for Quantile

Method	Speed-up	SST-2	RTE	QNLI	CoLA	MRPC	MNLI	QQP	STS-B	Macro
Dev set										
ALBERT	×1.0	92.7	76.5	91.5	56.6	**90.5**	**84.8**	88.9	**90.6**	84.0
PABEE	×1.41	92.7	76.9	**91.5**	55.6	88.3	84.5	88.9	89.9	83.5
PonderNet	×1.48	91.3	74.0	88.3	51.3	87.1	81.7	87.7	88.2	81.2
PALBERT	×1.29	**93.1**	**78.3**	91.0	**58.1**	89.3	84.7	**88.9**	89.9	**84.2**
Test set										
ALBERT	×1.0	**93.4**	70.0	**92.1**	**50.5**	85.6	79.0	**84.7**	**87.4**	80.3
PABEE	×1.39	92.7	71.1	91.3	46.0	84.3	79.2	83.7	86.5	79.3
PALBERT	×1.26	93.0	**73.5**	91.7	48.6	**87.1**	**79.8**	84.4	86.5	**80.6**

Table 1: A comparison of PALBERT with recent approaches on the GLUE benchmark. Each result for the dev set is a median task score across 5 runs. We report the two metrics' mean for the MRPC, QQP, and STS-B tasks. For the MNLI task, we report the mean accuracy across matched and mismatched datasets. For the test set, we used the best model according to the dev score. In the Macro column, we present the average results across tasks. We bolded the best results and underlined the second-best results.

4 Experiments

4.1 Ablation Study

We performed an ablation study of the proposed changes in PonderNet architecture. We experimented with adding the proposed Q-exit criterion, Lambda layer architecture, and fine-tuning strategies. We also compared the proposed changes with fine-tuning vanilla ALBERT. These methods were benchmarked on SST-2, RTE, and CoLA tasks from the GLUE Benchmarking dataset (Wang et al., 2018).

For evaluation, we performed a grid hyperparameter search on an appropriate metric score on the dev split for each dataset. Following the PABEE training setup, we trained all models with a fixed learning rate until validation metrics stopped increasing for 5 epochs. We used a fixed $q = 0.5$ for all experiments on models with the Q-exit criterion.

We trained each model 5 times with the best hyperparameters and reported the mean and std values. A full list of the methods' hyperparameter ranges can be found in Table 3.

See Table 2 for the full list of the results of our ablation study. Based on these experiments, PonderNet architecture is seen as performing significantly worse than vanilla ALBERT fine-tuning. At the same time, the deterministic Q-exit criterion dramatically improves PonderNet accuracy when compared to a random sampling of the exit layer. A more complex Lambda layer that can handle hidden state changes' dynamics can further improve model accuracy when compared to the original PonderNet.

4.2 Understanding the Threshold of Q-exit

As noted previously in Section 3.1, we treat the threshold value q of the Q-exit criterion as a trade-off between underthinking and overthinking, where increasing q forces a model to evaluate more layers, and vice versa.

Therefore, it is necessary to find the best-performing threshold for each task where a model has the highest accuracy. To do so, we evaluated trained PALBERT models from the ablation study (see Section 4.1) on dev splits of tasks with different values of q. We then averaged obtained metrics and reported the mean and std values for various thresholds (See Figure 3 for the results).

We observed that exiting models with $q = 0.5$ shows the best overall performance for different tasks. Making q greater than 0.5 leads to a reduction in accuracy and can often force models to evaluate all 12 layers of ALBERT-Base.

We associate such behavior of trained models with the fact that the huge probability mass of trained posterior probability $p(i|x)$ is concentrated near the last layers of models (see Figure 2). We hypothesize that the reason for this is that the parameterization of prior probability $p(i|\lambda)$ as geometric distribution with normalized last layer, proposed with PonderNet (Banino et al., 2021), leads to a huge prior probability of exiting on the last layers (See Section 3). For MRPC, we observe a huge variance in the probabilities of exiting from different models on the first layers, which we believe leads to poor performance on this task. Note that these plots could be seen as an estimation of proba-

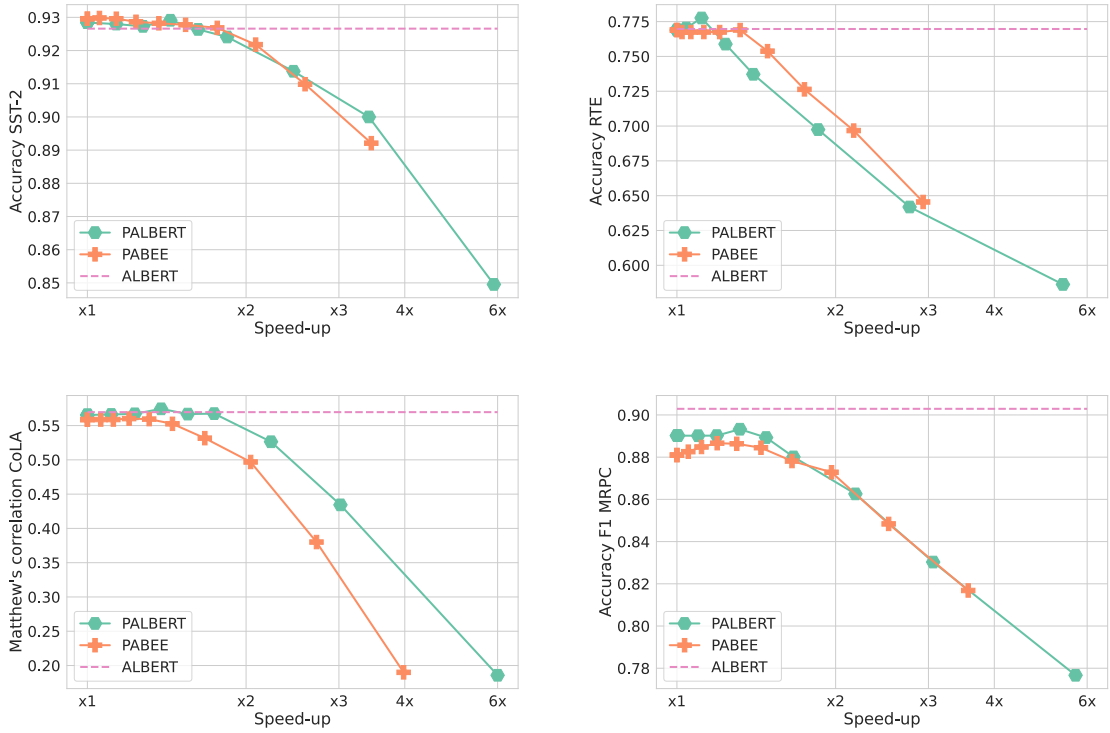

Figure 4: A comparison between PALBERT and PABEE models on CoLA and SST-2 tasks. We varied the threshold value of Q-exit for PALBERT and the patience hyperparameter for PABEE to obtain the plots of task scores of inference increasing in speed. 1x stands for plain ALBERT inference without performing an early exit. The horizontal line corresponds to plain ALBERT fine-tuning. See Section 4.3 for the analysis of these plots.

bilities of exiting from each layer with vanilla PonderNet sampling exit criterion. For the RTE task, layers $i \in [1; 10]$ have approximately the same probability of exiting with total probability mass close to 0.5, introducing huge variance in model outputs.

It is also notable that PALBERT, with a large threshold value q that performs constant exit on the last layer, has better accuracy than vanilla ALBERT fine-tuning for the SST-2 task.

4.3 Speed Analysis

While making $q < 0.5$ improves inference speed, it can also lead to underthinking and lower accuracy (see Figure 4). We compared PALBERT using different threshold values q to PABEE with different patience values t, which stands for the number of layers necessary to output the same result in a row to perform an early exit. We trained a PABEE model following the setup from the ablation study (see Section 4.1). We evaluated task scores for the specified hyperparameters as well as the increase in speed when compared to vanilla ALBERT inference of a full model with 12 layers.

Overall, we observed that PALBERT produced higher scores on different tasks while also being slightly faster than PABEE. For the CoLA and MRPC datasets, PALBERT performed significantly better. The proposed method outperformed PABEE by a large margin while achieving the same increase in speed.

We observed questionable results for the SST-2 dataset: the best score for the PABEE model is slightly higher than for PALBERT. However, it was obtained with a negligible increase in speed, because the best-performing patience for this setup is 11 layers (while the whole model has only 12 layers).

Furthermore, unlike PALBERT, PABEE performed significantly worse than plain ALBERT fine-tuning on the CoLA and RTE tasks. We hypothesize that the reason for this is that separated classifiers for each layer C_i in PABEE were not able to train well enough on such small datasets as CoLA and RTE. Therefore, we can assume that performing an early exit to avoid overthinking is not the main feature of fine-tuning a well-performing

Method				SST-2	RTE	CoLA
ALBERT				92.7 ± 0.3	77.0 ± 1.9	57.0 ± 2.1
PonderNet				91.1 ± 0.6	73.5 ± 1.9	50.8 ± 2.2
Q-exit	Lambda LR	3-Layer Lambda	hidden concat.			
+	-	-	-	92.2 ± 0.3	77.3 ± 1.4	55.7 ± 0.9
+	+	-	-	92.7 ± 0.4	77.3 ± 1.4	56.5 ± 1.2
+	+	+	-	92.6 ± 0.3	77.0 ± 1.4	56.3 ± 2.4
+	+	-	+	**93.0 ± 0.3**	76.5 ± 1.6	56.9 ± 1.9
+	+	+	+	92.9 ± 0.2	**77.8 ± 1.2**	**57.4 ± 1.7**

Table 2: An ablation study of the proposed PALBERT architecture. "Lambda LR" corresponds to fine-tuning the Lambda layer with its own learning rate, "3-layer Lambda" refers to making the Lambda layer have three MLP layers instead of one, and "hidden concat." stands for concatenation of two hidden states as input to the Lambda layer.

model. Instead, it might be possible to simply focus on improving the training process (e.g., by adding auxiliary tasks on each layer).

4.4 GLUE Experiments

Finally, we compared PALBERT with different baseline models on all GLUE tasks.

We re-implemented PABEE according to the original work (Zhou et al., 2020) and used a fixed patience value $t = 6$. We also compared PALBERT with PonderNet architecture adapted for ALBERT fine-tuning. We trained 5 models with the best hyperparameters across the hyperparameter search and reported the median task score on the dev set. We evaluated the test scores on the best models, selected based on their dev scores.

See Table 1 for the full list of results. We observed that PALBERT significantly outperformed PABEE on a wide range of tasks. Vanilla PonderNet with the sampling exit criterion performed the worst. Vanilla ALBERT outperformed PABEE on most tasks and is comparable to PALBERT, while the latter has the higher score averaged across all tasks (see Macro column in Table 1).

PABEE showed the highest increase in speed and is faster than vanilla ALBERT fine-tuning $\times 1.41$ times. PALBERT is still $\times 1.29$ times faster than vanilla ALBERT, while also significantly outperforming PABEE on most tasks.

Note that for tasks with a small dataset (e.g., CoLA, RTE), PABEE is performing poorly. We hypothesize that this is caused by several independent classifiers at each layer C_i failing to train well enough, whereas PALBERT was capable of utilizing knowledge sharing between layers.

Parameter	Values range
Learning rate	[1e-5, 2e-5, 3e-5, 5e-5]
Batch size	[16, 32, 128]
Lambda learning rate	[1e-5, 2e-5, 3e-5]
β	[0.5]
λ	[0.1]
Optimizer	[Adam]
Classifier dropout	[0.1]

Table 3: Hyperparameter search ranges used in all of our experiments. Vanilla ALBERT and PABEE only used batch size and learning rate parameters, while the PonderNet model avoids finding the best Lambda layer learning rate. Weight β of KL used in Equation 2 has a fixed value of 0.5, while prior exit probability distribution parameter λ is fixed to 0.1 and following original PonderNet (Banino et al., 2021).

5 Conclusion and Future Work

In this paper, we proposed improving the PonderNet architecture in order to perform an early exit using a fine-tuned ALBERT model with the novel Q-exit criterion and a revisited Lambda layer architecture. While PALBERT outperformed some recent State-of-The-Art methods used for early exit, there is a clear direction for further improvement of this method, as it was not capable of outperforming plain ALBERT on some GLUE tasks.

We believe that PALBERT could benefit from the development of new parameterization of the prior distribution on exiting from each layer since it directly affects the resulting posterior distribution used to perform an early exit (see Figure 2).

In addition, adding more auxiliary tasks could also make it possible to improve PALBERT further. This way, training of PALBERT can be made more

PABEE-like by making independent classifiers on each layer of the model or adding self-distillation across layers.

Finally, there is still no theoretical justification for the Q-exit threshold value. Although we observed that $q = 0.5$ performed best, it is without a clear explanation as to why that is so. We hypothesize that bringing more insights into developing deterministic exit criteria could further improve the proposed method.

References

Andrea Banino, Jan Balaguer, and Charles Blundell. 2021. Pondernet: Learning to ponder. In *8th ICML Workshop on Automated Machine Learning (AutoML)*.

Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. Fastbert: a self-distilling BERT with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6035–6044. Association for Computational Linguistics.

Surat Teerapittayanon, Bradley McDanel, and H. T. Kung. 2016. BranchyNet: Fast inference via early exiting from deep neural networks. In *Proceedings of the 23rd International Conference on Pattern Recognition*, pages 2464–2469. IEEE.

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, Amapreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. Cite arxiv:1804.07461Comment: https://gluebenchmark.com/.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems*, volume 33, pages 18330–18341. Curran Associates, Inc.

Wei Zhu. 2021. LeeBERT: Learned early exit for BERT with cross-level optimization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2968–2980, Online. Association for Computational Linguistics.