LittleBird: Efficient Faster & Longer Transformer for Question Answering

Minchul Lee and **Kijong Han** and **Myeong Cheol Shin**
Kakao Enterprise Corp., South Korea
{phil.i,mat.h,index.sh}@kakaoenterprise.com

Abstract

BERT has shown a lot of sucess in a wide variety of NLP tasks. But it has a limitation dealing with long inputs due to its attention mechanism. Longformer, ETC and BigBird addressed this issue and effectively solved the quadratic dependency problem. However we find that these models are not sufficient, and propose LittleBird, a novel model based on BigBird with improved speed and memory footprint while maintaining accuracy. In particular, we devise a more flexible and efficient position representation method based on Attention with Linear Biases (ALiBi). We also show that replacing the method of global information represented in the BigBird with pack and unpack attention is more effective. The proposed model can work on long inputs even after being pre-trained on short inputs, and can be trained efficiently reusing existing pre-trained language model for short inputs. This is a significant benefit for low-resource languages where large amounts of long text data are difficult to obtain. As a result, our experiments show that LittleBird works very well in a variety of languages, achieving high performance in question answering tasks, particularly in KorQuAD2.0, Korean Question Answering Dataset for long paragraphs.

1 Introduction

Transformer (Vaswani et al., 2017) and pre-trained language models (Devlin et al., 2019; Liu et al., 2019) based on it have shown a lot of success in a wide variety of NLP tasks. However, the quadratic dependency problem that comes from the attention mechanism makes it impractical to process long documents. Many techniques have been studied to overcome this problem and BigBird (Zaheer et al., 2020) showed robust and state-of-the-art performance on various NLP downstream tasks.

In this study, we propose a new model LittleBird by analyzing and improving the shortcomings of BigBird. LittleBird shows improved speed and memory footprint compared to BigBird while maintaining the overall accuracy of the question answering (QA) benchmarks and showing better accuracy in some of them.

In this study, we propose three major improvements compared to BigBird. The first is the method for position representation. In BigBird, trainable positional embedding is used similar to BERT (Devlin et al., 2019), and in ETC, relative positional encoding is used similar to T5 (Raffel et al., 2020). However, trainable positional embedding cannot handle longer inputs than those used for training and the relative position encoding is relatively slow and uses extra memory and parameters (Ma et al., 2021). Press et al. (2021) introduced the attention with linear biases (ALiBi) method that resolves these problems, but it was designed for causal language modeling, not autoencoding language modeling, which is typically useful for QA tasks. Thus, we devise a new method based on the ALiBi that is fast, flexible, and also effective in QA tasks.

The second is the method of capturing global information. BigBird introduces two ways of capturing global information, the random sparse attention and global tokens (Ainslie et al., 2020) which attend to and be attended by all other tokens. However, the random attention method is practically slow compared to its time complexity because it requires to repeat gather and scatter operations at random positions. In addition, a relatively large number of (∼hundreds) global tokens are required to achieve the reported performance using only global tokens without a random attention method in ETC. We show that replacing them with modified pack and unpack attention (Ma et al., 2021) is more effective.

The last is the efficient way to train a model for long sequences. We introduce a simple but effective method, Padding Insertion, which makes the model robust to long inputs while training on short inputs. We also propose a distillation method that

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 5261–5277

can maximize the reuse of the pre-trained model for a short length and show that our model can be effectively pre-trained using these methods.

Our model shows a 12∼29% reduction in peak memory usage and a 6∼46% reduction in latency compared to various BigBird and ETC model settings reported in the paper for 4K length document inference while showing better accuracy on several English QA benchmarks dev sets (Kwiatkowski et al., 2019; Welbl et al., 2018). Our model achieves new state-of-the-art performance on KorQUAD 2.0 (Kim et al., 2019), a Korean long document QA benchmark. In addition, these results are obtained with LittileBird pre-trained with only 2K sequence length. It shows that our novel positional representation method works well when the model is applied to the QA downstream task with a document longer than the sequence used in the pre-training phase.

2 Background and Related Work

2.1 Transformers for Longer Input

Various methods have been studied to maintain reasonable performance without using the quadratic operation of Transformer to handle long documents. Child et al. (2019) introduced sparse factorizations of the attention matrix which reduce the complexity to $O(n\sqrt{n})$. Reformer (Kitaev et al., 2019) reduced complexity to $O(n \log n)$ using locality-sensitive hashing.

Longformer (Beltagy et al., 2020) and ETC (Ainslie et al., 2020) proposed a method that utilizes several global attention tokens and local windowed attention and reduced the complexity to $O(n)$. In addition, these works showed performance improvement in downstream tasks. BigBird (Zaheer et al., 2020), an extended study related to ETC, propose random sparse attention method and provides detailed theoretical background and experimental results for more downstream tasks.

Recently, LUNA (Ma et al., 2021) introduced the method that approximates softmax attention with two nested linear attention functions called Pack and Unpack Attention, which has only linear time complexity. This method recorded improved performance in both speed and score in the Long Range Arena (LRA) benchmark (Tay et al., 2020).

2.2 Positional Encoding of Transformers

The attention mechanism of Transformers is defined as:

$$\text{Attn}\left(\mathbf{X}, \mathbf{C}\right) = \sigma\left(\frac{Q(\mathbf{X})K(\mathbf{C})^\intercal}{\sqrt{d}}\right)V(\mathbf{C})$$

where $\mathbf{X} \in \mathbb{R}^{l\times d}$ is the query sequence with length l, $\mathbf{C} \in \mathbb{R}^{m\times d}$ is the context sequence with length m, $\sigma(\cdot)$ is a softmax activation and Q, K, $V : \mathbb{R}^d \to \mathbb{R}^d$ is a linear transformation function projecting inputs into the space of query, key and value respectively. Since the attention function is ignorant of the position information of sequence, the Transformer model uses a method that added a special embedding to token embeddings on input of the first layer, called Positional Embedding, to inject position information. Vaswani et al. (2017) proposed Sinusoidal Positional Embedding, which is a non-trainable constant embedding computed from trigonometric functions.

On the other hand, BERT (Devlin et al., 2019) uses trainable positional embeddings instead of constant embeddings. It is adaptable to training data, but has limitations such as being unable to handle longer inputs than those used for training and not being translation-invariant.

Relative position methods have been studied, for solving these problems (Shaw et al., 2018; Raffel et al., 2020). It learns parameters representing the relative distance between tokens and utilizes them to calculate the attention score. However, It is slower than the sinusoidal approach and uses extra memory and parameters (Press et al., 2021).

Press et al. (2021) pointed out that previous methods are vulnerable to extrapolation and proposes ALiBi, a modified attention function for self-attention as follows:

$$\text{ALiBi}\left(\mathbf{X}\right) = \sigma\left(\frac{Q(\mathbf{X})K(\mathbf{X})^\intercal}{\sqrt{d}} - \mathbf{D}^\intercal\right)V(\mathbf{X})$$

$$\mathbf{D}_{i,j} = \begin{cases} m \times (i-j), & \text{for } i \geq j \\ \infty, & \text{for } i < j \end{cases}$$

where m is a head-specific positive real-number hyperparameter and $\mathbf{D} \in \mathbb{R}^{l\times l}$ is a distance matrix.

2.3 Pretraining objectives for Question Answering

To pretrain a language model, an appropriate training objective that fully exploits the language understanding should be defined. Masked LM (Devlin

et al., 2019), for example, replaces 15% of the input tokens with a mask token or a random token and forces the model to denoise it. After pre-training is complete, the last hidden representations of the model contains information to restore the replaced token to the original one and it is useful to transfer this information for other NLP tasks as well, such as question answering.

However, Masked LM (MLM) is suboptimal for extractive QA task. Joshi et al. (2020) proposed SpanBERT, which is pretrained by a span-level masking scheme whose lengths follows geometric distribution and it outperformed BERT with MLM in the most of tasks, especially extractive QA. They proved that training objective predicting spans rather than tokens generates better representations especially for span selection tasks.

Ram et al. (2021) introduced Recurring Span Selection (RSS), a novel pre-training objective which is better aligned to QA tasks. In RSS, each recurring text span, except for one to be used as the golden answer span, is masked with a special token, [QUESTION], and a model is trained to point to the position of the golden answer span using the representations from each [QUESTION] token. Because this pre-training task is so similar to the real QA task, the model trained in this objective outperforms models with other pre-training objectives in both the few-shot and high-resource settings for QA.

2.4 Datasets of Question Answering for Longer Documents

The most widely used English QA dataset is SQuAD (Rajpurkar et al., 2016), but it's insufficient to test understanding of long contexts because of its short paragraph. Thus, for QA of longer documents, other datasets are considered. Typical examples are Natural Questions (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017), which provide a whole Wikipedia page as the document. NarrativeQA (Kočiský et al., 2018), whose documents consist of movie scripts and books, is another example. Recently, Pang et al. (2022) introduced QuALITY, a multiple-choice QA dataset comprised of around 5000 tokens of documents gathered from various sources such as Project Gutenberg and Open American National Corpus.

For Korean QA datasets, the most standard is KorQuAD 1.0 and KorQuAD 2.0, which is comparable to SQuAD in English. The construction and characteristics of the dataset in KorQuAD 1.0 (Lim et al., 2019) are nearly identical to those of SQuAD, except that it is in Korean. Therefore, like SQuAD, KorQuAD 1.0 is not suitable for evaluating QA for long documents. To evaluate understanding of longer documents, KorQuAD 2.0 (Kim et al., 2019) is often used. Since it provides the whole Wikipedia page as a single context without trimming and the page includes not only text but also HTML components such as tables and lists, structural understanding of long HTML documents is required to conquer it.

3 LittleBird Architecture

In this section, we describe the architecture of LittleBird model. Basically, the model can be viewed as a composition of several key ideas including sliding window attention from BigBird (Zaheer et al., 2020), linear bias to attention from ALiBi (Press et al., 2021) and pack and unpack attention from LUNA (Ma et al., 2021).

3.1 Bidirectional ALiBi

Since pre-trained language models (PLM) perform best when using data of the same length as the data used for pretraining in general, a new PLM suitable for the length must be built to perform inference on longer data, which is inefficient. To avoid this, we consider the main idea of ALiBi (Press et al., 2021), which is more efficient than relative positional encoding used at T5. However, because ALiBi was designed for causal language modeling, not autoencoding language modeling, each query can attend to keys to the left of itself only, not keys further away or to the right in ALiBi.

Therefore, we devised BiALiBi (Bidirectional ALiBi), which is improved version of ALiBi to suit the autoencoding language model. BiALiBi has the same attention function as ALiBi, but differs only in the method of calculating the distance matrix as follows:

$$\mathbf{D}_{i,j} = \begin{cases} 0, & \text{for } i = j \\ \alpha & \text{for } i = 0 \text{ or } j = 0 \\ \beta(i-j), & \text{for } i > j \\ \gamma(j-i), & \text{for } i < j \end{cases}$$

where α, β and γ are head-specific slopes like m in ALiBi. α is a value for the [CLS] token, which usually appears at position 0. Because this token should be global, it has the same bias regardless of distance. β and γ are involved in the attention

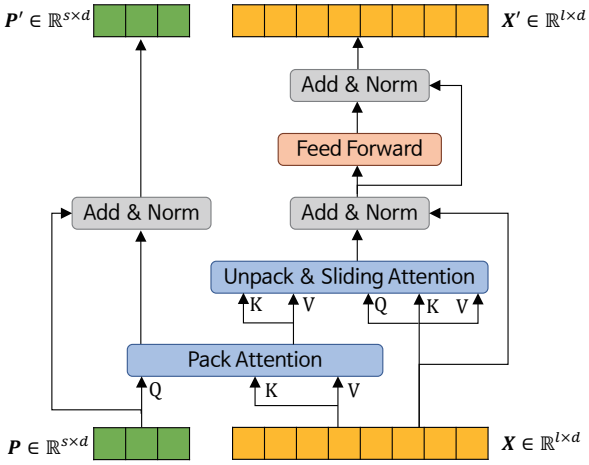


Figure 1: LittleBird Layer

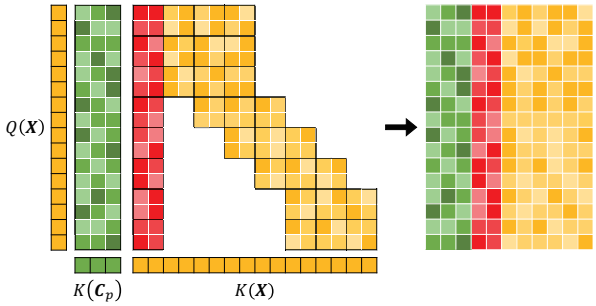


Figure 2: Unpack & Sliding Window Attention of LittleBird

intensity for tokens on the left and right, respectively. Unlike ALiBi, we set α, β and γ as learnable parameters to have more flexibility.

3.2 Sliding Window Attention

Attention module of BigBird (Zaheer et al., 2020) consists of three types of attentions: Global, Window and Random. Global tokens can attend to all other tokens and also can be attended from all other tokens. On the other hand, non-global tokens can only attend to all global tokens, some nearby tokens (Window) and random tokens (Random).

For efficient computation, this attention module is implemented using blocked sliding window. But there is still an overhead where random attention needs repeating gather and scatter operations at random positions. Since it is known that full attention can be substituted well without random attention when global tokens are sufficient (Ainslie et al., 2020), we completely eliminated random attention from our model. We also reduced the number of global tokens and removed global-local attention, They were replaced with pack and unpack attention, as explained in the following subsection.

3.3 Pack & Unpack Attention

To effectively replace random and global attention, we employed pack and unpack attention (Ma et al., 2021). However, in the original pack and unpack attention, information loss is unavoidable because all sequence information is packed into a small capacity. We propose adding the sliding window attention to the unpacking step to improve this. Figure 1 depicts the entire architecture of the LittleBird layer.

$$\mathbf{C}_P = \text{Attn}\left(\mathbf{P}, \mathbf{X}\right)$$
$$\mathbf{P}' = \text{LayerNorm}\left(\mathbf{C}_P + \mathbf{P}\right)$$
$$\mathbf{C}_X = \text{USWAttn}\left(\mathbf{X}, \mathbf{C}_P\right)$$
$$\mathbf{A} = \text{LayerNorm}\left(\mathbf{C}_X + \mathbf{X}\right)$$
$$\mathbf{X}' = \text{LayerNorm}\left(\text{FFN}(\mathbf{A}) + \mathbf{A}\right)$$

$$\text{USWAttn}\left(\mathbf{X}, \mathbf{C}_P\right) = \sigma\left(\frac{Q(\mathbf{X})\left[K(\mathbf{C}_P); K(\mathbf{X})\right]^\intercal}{\sqrt{d}} - \left[\mathbf{D}_P; \mathbf{D}\right]^\intercal\right) \cdot \left[V(\mathbf{C}_P); V(\mathbf{X})\right]$$

$$\mathbf{D}_P = \left(\frac{\beta + \gamma}{2} b\right) \mathbf{J}_{s,l}$$

where $\mathbf{X} \in \mathbb{R}^{l \times d}$ is the input sequence with length l, $\mathbf{P} \in \mathbb{R}^{s \times d}$ is an extra sequence for packing contextual information with length s, $[\mathbf{A}; \mathbf{B}]$ denotes concatenation of matrix $\mathbf{A}$ and $\mathbf{B}$ in row axis, $\mathbf{D} \in \mathbb{R}^{l \times l}$ is a distance matrix from BiALiBi, $\mathbf{D}_P \in \mathbb{R}^{s \times l}$ is a distance matrix for packing tokens and $\mathbf{J}_{s,l}$ is an all-ones matrix with shape (s, l).

The overall structure is the same as pack and unpack attention (Ma et al., 2021), but only one part, USWAttn (Unpack & Sliding Window Attention), is different. In this step, we split $\mathbf{X}$ into blocks with size b and perform block-level attention like Zaheer et al. (2020), which is demonstrated at Figure 2. We set only the first block as a global token, and allow local-to-global attention. This is because in most QA tasks, [CLS] tokens and questions are placed in the front part of the input sequence, and we believe it is important to allow the rest of the input sequence to access information of these tokens directly.

Also, we apply different distance matrices depending on the type of attention. BiALiBi's distance matrix $\mathbf{D}$ is applied to $\mathbf{X}$-to-$\mathbf{X}$ attention, but

uniform distance matrix $\mathbf{D}_P$ is applied to $\mathbf{X}$-to-$\mathbf{C}_P$ attention. Since $\mathbf{D}_P$ is defined as a value obtained by multiplying the average of β and γ by block size b, for input tokens, each packing token is considered to be separated by a distance b.

Finally, each block can attend itself (b tokens), the blocks next to it ($2b$ tokens), the global tokens(b tokens) and the packed context (s tokens). This can be effectively converted to batch matrix multiplication with $O\left(l(4b+s)\right)$ time complexity by rolling and concatenating the key blocks as BigBird. Also, pack attention can be done in $O\left(ls\right)$, so LittleBird attention is done in $O\left(l(4b+2s)\right)$. If s and b are sufficiently smaller than l, it can be considered to have linear time complexity to input length l. We choose $s = b = 64$.

3.4 Efficient Training and Padding Insertion

Even with better model structure, pre-training on long inputs is costly. This section describes how to train LittleBird efficiently by reusing an existing PLM. We trained our model by the following steps.

1. Initialize all parameters of new LittleBird layers as the corresponding parameters of PLM's layer. Both QKV of pack and of unpack attention are initialized from the attention layer.

2. Distil knowledge from PLM to the new model using not only soft target probability but also self-attention distribution of each layer as distillation loss following Sun et al. (2020), but not hidden states of layers. Since long inputs cannot be fed into the teacher model, short inputs are used in this step.

3. Further train the new model on longer inputs without distillation.

In the step 2, it is important to transfer the self-attention distribution. The parameters of BiALiBi largely dominates the overall attention pattern. We discovered that when we train these parameters without distillation, it takes several epochs to converge, but it converges considerably more stably and fast when we train them with distillation.

When we feed data into the LittleBird model in steps 2 and 3, we can use Padding Insertion (PI), a simple trick to fool the model into thinking it is receiving longer data even though receiving shorter data actually. Consider the following scenario: a virtual padding token is randomly inserted in the middle of input data. Padding tokens are masked because they should not be attended by other tokens. As a result, inserting padding has no effect on other tokens, but only on the computing distance matrix of BiALiBi. So, instead of inserting padding tokens actually, we can achieve the same result by manipulating the position ids of input sequence for the distance matrix. This allows us to train the model to prepare for long inputs while actually taking short inputs. It makes the extrapolation capability of the model robust, and the experimental results for this are given in Section 4.4.

Also, it may be important to balance the epochs of steps 2 and 3 for efficient training. We obtained good results in the following experiments by simply allocating the same epochs to steps 2 and 3, but better balancing is worth further study.

4 Experiment

The goal of this section is to demonstrate the benefits of LittleBird compared to other long transformer models. First we check if the proposed training method works well with the architecture of LittleBird. Following that, we consider QA tasks of English and Korean, which requires longer sequence modeling. Lastly we prove that Padding Insertion significantly improves accuracy and that the efficiency of our model by performing training and inference for various input lengths.

4.1 Pre-training using RSS

We pre-trained the model with the RSS objective (Ram et al., 2021) to create a English and a Korean model optimized for QA. For pre-training the English model, RoBERTa checkpoint[1] was used for initializing and it was also used as the teacher model in the procedure described in Section 3.4. Three public datasets, OpenWebText(Gokaslan and Cohen, 2019), CC-News(Guu et al., 2020) and English Wikipedia were used. Similarly, KoELECTRA checkpoint [2] was used for the Korean model, and various Korean corpora including Wikipedia were used. Details for pre-training are attached in the Appendix A.

Table 1 displays the accuracy of the four English pre-trained models on dev datasets of 512, 2k and 4k lengths. An accuracy of RSS tasks is defined as the proportion in which the span selected by a model exactly matches the golden answer. The four models **A**, **B**, **C** and **D** were pre-trained using different training settings. The LittleBird model **A**

[1] https://huggingface.co/roberta-base
[2] https://github.com/monologg/KoELECTRA

Model	512	2k	4k
A: LittleBird, distilled with 512 length	84.46	42.02	32.08
B: LittleBird, trained with 2k length from **A**	73.68	72.67	47.92
C: LittleBird, trained with 4k length from **A**	65.34	64.08	63.59
D: BigBird, trained with 4k length	66.04	48.97	58.30

Table 1: RSS pre-training accuracy of English dev datasets. Each column represents the sequence length of the dev datasets.

was trained using only steps 1 and 2 in Section 3.4, and the models **B** and **C** were further trained from **A** using step 3. The model **D** was trained using warm-starting from the same RoBERTa checkpoint without any distillations in BigBird architecture for the same amount of time as **B** and **C**. All results of the table are without Padding Insertion. It is noteworthy that the LittleBird model **C** achieved higher pre-training accuracy than the BigBird model **D** at 2k and 4k dev datasets when trained at the same time, and **B** and **C** loses their accuracy for short inputs as trained for longer inputs.

4.2 QA Benchmark for English

In this section, we compare the performance of LittleBird with other models for long inputs, such as Longformer (Beltagy et al., 2020), BigBird and ETC. We used the LittleBird models pre-trained on 2048-length with Padding Insertion (PI) in this experiments. The English BigBird model was also trained in the same setting as LittleBird using RSS for comparison. Experiments were performed on four QA datasets, HotpotQA (Yang et al., 2018), NaturalQ(Kwiatkowski et al., 2019), TriviaQA(Joshi et al., 2017) and WikiHop(Welbl et al., 2018), used by Zaheer et al. (2020). Because the most of datasets have long input data of more than 4K tokens, they are appropriate for evaluating long text understanding. For detailed statistics on these datasets, see Table 9 in the appendix. The top three rows of the Table 2 are from previous papers' results, and the row **BigBird + RSS** denotes the model pre-trained using RSS objective with the same training time as LittleBird in Section 4.1. LittleBird shows more effectiveness when computational resources and time are limited and achieved higher accuracy than other models in the most of cases.

The same experiment was repeated while changing the pack size s to examine the effect of pack and unpack attention, and the results are shown in Table 3. When $s = 0$, LittleBird model is equivalent to a model with only sliding window attention and local-global attention. It can be confirmed that pack and unpack attention is significantly effective in all cases.

4.3 QA Benchmark for Korean

First, we compared the base size LittleBird model with other Korean PLM using dev set of KorQuAD2.0, and the results are shown in Table 4. For KLUE-RoBERTa and KoBigBird, we described the reported performance, however for KoELECTRA, we conducted fine-tuning and evaluation using the published model. In the table, **KoELECTRA** denotes the results of using the published model as is, and **KoELECTRA + RSS** denotes the results after further training with RSS. Also, on the same LittleBird model, we repeated the experiment by varying the maximum sequence length of fine-tuning. The LittleBird model fine-tuned on 8K showed the best performance and this clearly demonstrates the significance of broad context when performing QA on long documents.

A large model was trained using the best setting of the base-size model, we submitted both the base and the large size model to the evaluation system, and the results are shown in Table 5[3]. The models listed in the table with LittleBird are the previous top three models. Although these models' exact structure or size is not disclosed, it can be confirmed that LittleBird is faster and more accurate than them.

4.4 Effect of Padding Insertion

In addition, experiments were performed to determine whether the Padding Insertion (PI) introduced in Section 3.4 is effective. First, we verified that the model fine-tuned with PI performs well with inputs that are longer than those used for fine-tuning. We chose a Korean fake news dataset(Lee et al., 2019) for fine-tuning, which requires binary clas-

[3]The whole list is available at https://korquad.github.io/

Model	HotpotQA			NaturalQ		TriviaQA	Wikihop
	Ans	Sup	Joint	LA	SA	Full	MCQ
Longformer	74.3	84.4	64.4	-	-	75.2	75.0
BigBird	75.7	86.8	67.7	70.8	53.3	**79.5**	75.9
ETC	75.5	**87.1**	67.8	73.9	54.9	78.7	75.9
BigBird + RSS	73.7	83.2	63.2	71.1	51.2	75.5	75.1
LittleBird	**77.7**	86.3	**68.5**	**76.7**	**57.5**	76.9	**82.0**

Table 2: QA Dev results using base-size models. We report accuracy for WikiHop and F1 for the others.

Pack Size	HotpotQA			NaturalQ		TriviaQA	Wikihop
	Ans	Sup	Joint	LA	SA	Full	MCQ
0	69.0	77.0	54.6	73.3	54.1	71.8	73.9
32	72.2	83.0	61.7	73.4	55.1	72.5	77.1
64	**77.7**	**86.3**	**68.5**	**76.7**	**57.5**	**76.9**	**82.0**

Table 3: QA Dev results of LittleBird with varying pack size. We report accuracy for WikiHop and F1 for the others.

Model / Seq. Len.	EM	F1
KLUE-RoBERTa / 512 (Park et al., 2021)	55.44	73.02
KoBigBird / 4K (Park and Kim, 2021)	67.77	82.03
KoELECTRA / 512 (Park, 2020)	66.95	77.75
KoELECTRA + RSS / 512	68.56	79.30
LittleBird / 512	68.11	80.38
/ 1K	72.82	83.09
/ 2K	75.50	84.78
/ 4K	76.01	85.20
/ 8K	**77.79**	**89.39**

Table 4: KorQuAD2.0 Dev results using Base size models

Model	EM	F1	Latency
SDS-Net v1.3	77.86	89.92	10.43
Ko-LongBERT	77.88	89.62	10.05
SkERT-Large 1.1	77.44	88.81	10.05
LittleBird-Base	76.66	88.57	2.29
LittleBird-Large	**78.70**	**90.22**	6.16

Table 5: KorQuAD2.0 Test results with previous SOTAs. Latency is a measure of the average response time per question in seconds.

Seq. Len.	512	1024	1536	2048
512	99.53	48.17	47.31	74.23
1K	99.49	**99.15**	98.21	88.66
512 + PI	99.36	97.67	96.06	95.88
1K + PI	**99.57**	99.11	**98.92**	**96.91**
True labels	44.7%	52.5%	56.3%	27.6%

Table 6: Classification accuracy with varying max sequence length of fine-tuning Korean fake news data. Each column represents the sequence length of the dev dataset. The last row represents the distribution of true labels for documents in each interval.

sification of whether a given text is fake or not. Since the dataset's average token length of 774 and maximum token length of 17,488 and it contains many documents longer than 512 tokens, it was appropriate for evaluating accuracy on long inputs. For this experiment, the Korean LittleBird in base size, pre-trained without PI, was used. Table 6 displays the result. Because this dataset's label distribution is slightly skewed, we presented the ratio of true labels for each interval in the table so that the accuracy at each interval may be compared more equitably.

The row in the table with **+ PI** indicates that PI was used in the fine-tuning phase. Paddings with a length between 0 and 256 were inserted randomly with a 20% probability at the end of each sentence (after the '.', '?' and '!' token). When the model was fine-tuned on 512-length sequences, it did not work well for inputs longer than 512. This result is consistent to the prior study (Press et al., 2021) that found that the extrapolation performance is poor

when the head-specific slope parameters of ALiBi are learnable. However, when fine-tuned with PI on 512-length sequences, the accuracy was comparable to one fine-tuned on sequences of 1024-length and the model fine-tuned with PI on 1024-length sequences show the best accuracy. The results show that PI can improve the model's extrapolation performance for longer inputs.

We also performed experiments about the effect of PI in the pre-training stage. We looked at the four QA datasets used in Section 4.2. Table 7 shows the effect of sequence length and PI in the pre-training stage for the English LittleBird model. Pre-trained models on longer inputs did not always perform well. Models with PI, on the other hand, consistently outperformed those without. Particularly, the model pre-trained on 2048-length with PI outperforms the other models. Pre-training on long inputs does not guarantee accuracy on shorter inputs, as shown in Table 1. When PI is used, all inputs will have varying lengths, causing the model to encounter inputs of varying lengths, which makes the model more robust.

4.5 Ablation Study

We conducted an ablation study to assess how much each component contributed to the LittleBird model's performance, and the results are shown in Table 8. As can be seen from the table, the factors that have the biggest impact on performance are Pack Attention and Sliding Window Attention. And in the case of BiALiBi, its contribution to performance improvements is small but it helps to replace Absolute Positional Embedding, allowing it to accept variable-length inputs. It can be seen that applying Padding Insertion without changing the structure of the model makes an additional improvement by itself.

4.6 Speed & Memory efficiency

To confirm the efficiency of LittleBird empirically, we investigate the speed and memory footprint of base-size BigBird, ETC and LittleBird with varying input lengths (1K to 4K for BigBird, 1K to 8K for others). All models are evaluated on simple masked-language model task with the same single batch. The result is shown in Figure 3.

In all three models, memory and computation time increase in proportion to input length, and in the case of the ETC model, the number of global tokens has a significant impact on their overall ef-

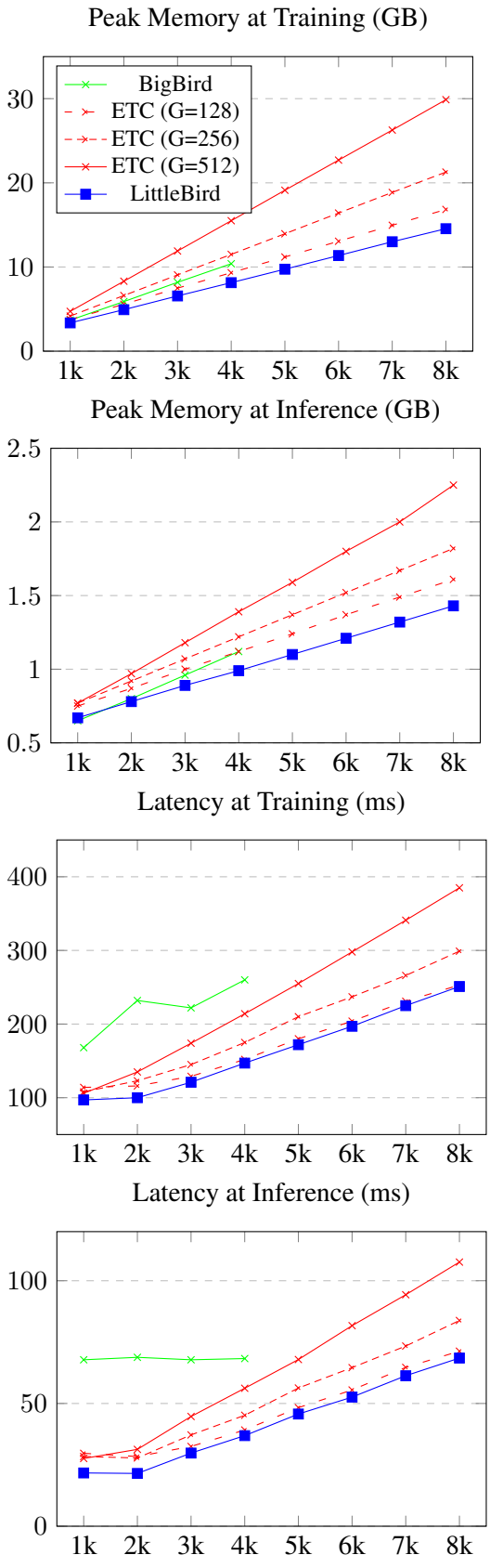


Figure 3: Peak memory usage and latency at training and inference in a single batch with varying input sequence lengths. Measured at PyTorch 1.8.1 & CUDA 11.1 environment on a single NVIDIA A100. (G=n) indicates the number of global tokens in ETC model.

Seq. Len.	HotpotQA			NaturalQ		TriviaQA	Wikihop
	Ans	Sup	Joint	LA	SA	Full	MCQ
512	75.3	83.5	64.2	74.3	55.8	70.1	77.8
2K	75.5	85.3	65.9	74.8	55.2	74.6	79.7
4K	74.3	84.8	64.7	74.8	55.9	74.3	79.1
512 + PI	76.1	85.6	66.6	74.9	57.3	75.8	79.9
2K + PI	**77.7**	**86.3**	**68.5**	**76.7**	**57.5**	**76.9**	**82.0**

Table 7: QA Dev results with varying max sequence length of pre-training data. We report accuracy for WikiHop and F1 for the others.

Model	HotpotQA			NaturalQ		TriviaQA	Wikihop
	Ans	Sup	Joint	LA	SA	Full	MCQ
LittleBird	**77.7**	**86.3**	**68.5**	**76.7**	**57.5**	**76.9**	**82.0**
- BiALiBi	75.2	85.2	65.6	74.3	55.4	76.5	78.3
- Sliding Window Attn	16.1	27.9	6.0	11.8	1.4	33.9	20.3
- Pack Attn	69.0	77.0	54.6	73.3	54.1	71.8	73.9
- Padding Insertion	75.5	85.3	65.9	74.8	55.2	74.6	79.7

Table 8: Ablation study of LittleBird. We report accuracy for WikiHop and F1 for the others.

ficiency. [4] It is noteworthy that the authors set the number of global tokens in the range of 230 to 430 when testing on QA datasets, it is clear that the result between $G = 256$ and $G = 512$ is the performance of a practically usable ETC model. In all cases, the efficiency of LittleBird far outperforms that of the other two models.

5 Conclusion

We propose LittleBird, which is more efficient in terms of memory and computational time than existing Transformer models for long sequences, and its effective way to train. It combines a novel position encoding method, BiALiBi, and pack & unpack with sliding window attention to achieve high speed and accuracy, particularly in question answering tasks for long documents. The distillation and training method with Padding Insertion allows the model to be trained by reusing the existing pre-trained language model for short inputs and work well for long inputs even if trained on short inputs. We demonstrated through experiments that the accuracy of question answering improves as the model is fed a longer input, and we achieved state-of-the-art performance in KorQuAD2.0 using LittleBird.

[4] In the case of model BigBird, as an exception, the inference latency did not increase in proportion to input length. Since the same results were obtained in several repeated experiments, it is assumed that there is an overhead in the BigBird implementation of transformers 4.6.1, which is used in the experiment.

Limitations

First, despite the efficiency of LittleBird and its excellent performance in question answering, it is still unknown whether LittleBird works well for other NLP tasks. Further research is needed on other tasks. Second, in the encoder-decoder architecture model that requires cross-attention, since position information is not injected into each token at all, it may be difficult for the decoder layer to find appropriate tokens of the encoder to attend. Lastly, causal masking cannot be applied to the pack and unpack attention due to its characteristics, which means that LittleBird cannot be used to the decoder-only autoregressive language model.

Ethics Statement

No private data is used at all, and all the datasets used in this paper for pre-training and fine-tuning are accessible to the public. Thus, our work is free of any privacy issues.

References

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284, Online. Association for Computational Linguistics.

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

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 845–855, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Youngmin Kim, Seungyoung Lim, Hyunjeong Lee, Soyoon Park, and Myungji Kim. 2019. Korquad 2.0: Korean qa dataset for web document machine comprehension. In *Annual Conference on Human and Language Technology*, pages 97–102. Human and Language Technology.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2019. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Dong-Ho Lee, Yu-Ri Kim, Hyeong-Jun Kim, Seung-Myun Park, and Yu-Jun Yang. 2019. Fake news detection using deep learning. *Journal of Information Processing Systems*, 15(5):1119–1130.

Seungyoung Lim, Myungji Kim, and Jooyoul Lee. 2019. Korquad1. 0: Korean qa dataset for machine reading comprehension. *arXiv preprint arXiv:1909.07005*.

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

Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. 2021. Luna: Linear unified nested attention. In *Advances in Neural Information Processing Systems*, volume 34, pages 2441–2453. Curran Associates, Inc.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel Bowman. 2022. QuALITY: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5336–5358, Seattle, United States. Association for Computational Linguistics.

Jangwon Park. 2020. Koelectra: Pretrained electra model for korean. `https://github.com/monologg/KoELECTRA`.

Jangwon Park and Donggyu Kim. 2021. Kobigbird: Pretrained bigbird model for korean.

Sungjoon Park, Jihyung Moon, Sungdong Kim, Won Ik Cho, Jiyoon Han, Jangwon Park, Chisung Song, Junseong Kim, Yongsook Song, Taehwan Oh, Joohong Lee, Juhyun Oh, Sungwon Lyu, Younghoon Jeong, Inkwon Lee, Sangwoo Seo, Dongjun Lee, Hyunwoo Kim, Myeonghwa Lee, Seongbo Jang, Seungwon Do, Sunkyoung Kim, Kyungtae Lim, Jongwon Lee, Kyumin Park, Jamin Shin, Seonghyun Kim, Lucy Park, Alice Oh, Jungwoo Ha, and Kyunghyun Cho. 2021. Klue: Korean language understanding evaluation.

Ofir Press, Noah A Smith, and Mike Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits

of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Ori Ram, Yuval Kirstain, Jonathan Berant, Amir Globerson, and Omer Levy. 2021. Few-shot question answering by pretraining span selection. 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 3066–3079, Online. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2020. Long range arena: A benchmark for efficient transformers. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc.

A Experiments details

A.1 Pre-training

We used three publicly available datasets OpenWebText(Gokaslan and Cohen, 2019), CC-News(Guu et al., 2020) and Wikipedia to pre-train English LittleBird model. The RoBERTa model's vocabulary was borrowed, and the weights were also initialized from the same RoBERTa checkpoint[5]. For Korean LittleBird model, Newspaper, Written, Web corpus from 모두의 말뭉치(Moduui Malmungchi)[6] and Wikipedia were used, we borrowed the vocabulary of KoELECTRA(Park, 2020) and used its weights to initialize.

Following Ram et al. (2021), we preprocessed the corpora to find recurring spans. In the cluster selection step, up to 30 spans were dynamically selected for each 512-token document, 120 spans for each 2K-token document and 240 spans for each 4K-token document. QASS was used as a pre-training architecture, but instead of their proposal of predicting the start and the end positions of the answer independently, we predict the start position first and then jointly predict the end position conditionally on the start position.

A.2 Fine-tuning on Question Answering tasks

We basically applied QASS on the model architectures for question answering used by Zaheer et al. (2020). The task-specific detailed structure is as follows.

HotpotQA: We place the [CLS] token, the [QUESTION] token and the question in sequence, and then each paragraph separated by [SEP] is followed. Also, 32 virtual paddings are inserted between paragraphs for LittleBird model. A linear layer is used to predict the start position of the answer, and a concatenation of the representations of the start position and of the end position is fed to double layer consisting of a gelu-activated layer and a linear layer to predict the ending position. For evidence classification, a concatenation of the representations of the start and end positions of each evidence is fed into a double layer with a gelu-activation. Finally, for answer type classification, the representation of the [CLS] token is fed into a double layer with a gelu-activation.

NaturalQ: We place the [CLS] token, the [QUESTION] token and the question like HotpotQA, and the whole paragraph is followed. A sliding window with 4K-length stride was used for paragraphs exceeding 8K in length. For short answers, a model predicts the start position first, and then predicts the end position by concatenating the representation of start and the end position, similarly to HotpotQA. To predict long answers, a concatenation of the representation of the start and the end position is fed to double layer with a gelu-activation. Also, the representation of the [CLS] token is used for answer type classification like HotpotQA.

TriviaQA: We place the [CLS] token, the [QUESTION] token and the question, and then each paragraph separated by [SEP] is followed. Also, 32 virtual paddings are inserted between paragraphs for LittleBird model. A sliding window with 4K-length stride was used for paragraphs exceeding 8K in length. For training noisy spans we follow Clark and Gardner (2018). To predict the start and the end positions of answers, we use the same predictor as HotpotQA's one.

WikiHop: We place the [CLS] token, the [QUESTION] token, and the question, and then each answer separated by [SEP] is followed next, and each paragraph separated by [SEP] is followed lastly. Also, 32 virtual paddings are inserted between paragraphs for LittleBird model. To predict the start and the end positions of answers, we use the same predictor as HotpotQA's one.

KorQuAD2.0: We place the [CLS] token, the [QUESTION] token and the question like HotpotQA, and the whole paragraph is followed. A sliding window with 4K-length stride was used for paragraphs exceeding 8K in length. Also we add tokens (<H1>, <H2>, <P>, <Table>, <Tr>, <Td>, <Th>, <Ul>, <Ol>, <Li>) for key HTML elements to the pre-trained model's vocabulary. Also, HTML documents of KorQuAD2 was preprocessd to remove unnecessary headers, footers, script, and style tags. To predict the start and the end positions of answers, we use the same predictor as HotpotQA's one.

Table 9 shows the statistics of QA dataset used in this paper. Sequence lengths were measured in sub-word tokens and tabulated with averages and 0 to 100 percentiles. Table 10 displays the hyperparameters for English LittleBird that were used to create Tables 2 and 3. Hyperparameters for Korean Littlebird, used for creating Table 4 and 5 are shown in Table 11.

[5] https://huggingface.co/roberta-base
[6] https://corpus.korean.go.kr/

Lang.	Dataset	# of Examples		Sequence Length in Tokens	
		Train	Dev	0 / 25 / 50 / 75 / 100th Perc.	Avg.
English	HotpotQA-distractor (Yang et al., 2018)	90447	7405	59 / 1063 / 1259 / 1476 / 3717	1284
	Natural Questions (Kwiatkowski et al., 2019)	307373	7830	269 / 3852 / 7371 / 14120 / 147467	10485
	TriviaQA (Joshi et al., 2017)	61888	7993	32 / 3687 / 8828 / 16469 / 177156	11789
	WikiHop (Welbl et al., 2018)	43738	5129	78 / 759 / 1308 / 2076 / 19802	1563
Korean	KorQuAD2.0 (Kim et al., 2019)	83486	10165	194 / 1661 / 2813 / 4985 / 20557	4411

Table 9: Statistics of Question Answering datasets

Parameter	HotpotQA	NaturalQ	TriviaQA	WikiHop
Pack Size	64			
Block Size	64			
Max Seq. Length	4096	8192	8192	8192
# of heads	12			
# of hidden layers	12			
Hidden layer size	768			
Total Parameters	145M			
Batch size	64	128	128	32
Optimizer	AdamW			
Learning rate	5e-5			
Compute resources	4 $\times$ NVIDIA A100			

Table 10: Hyperparameters of base LittleBird model used for English Question Answering

Parameter	Base	Large
Pack Size	64	
Block Size	64	
Max Seq. Length	6144	8192
# of heads	12	16
# of hidden layers	12	24
Hidden layer size	768	1024
Total Parameters	133M	414M
Batch size	80	144
Optimizer	AdamW	
Learning rate	1e-4	5e-5
Compute resources	4 × NVIDIA A100	

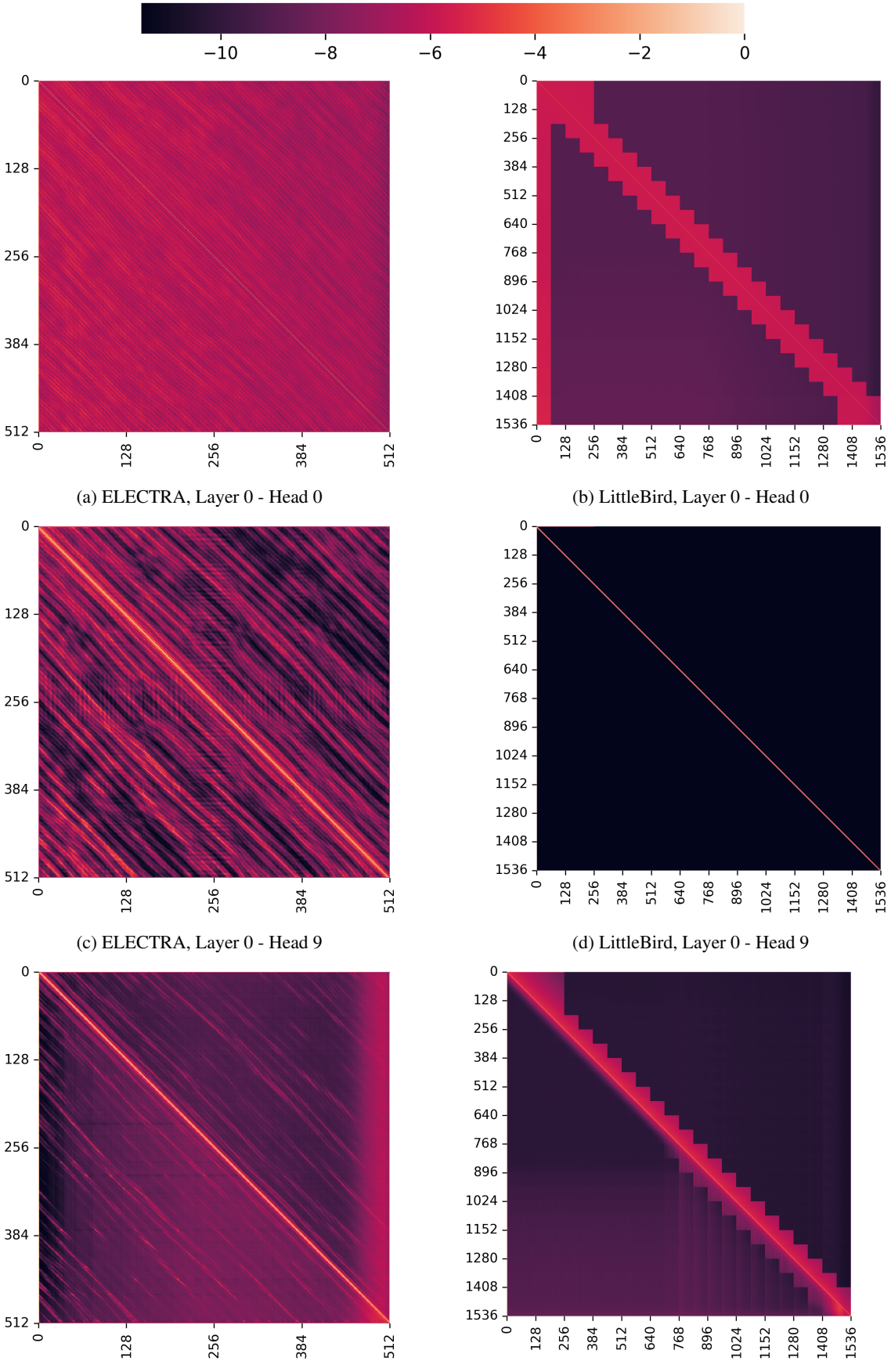
Table 11: Hyperparameters of LittleBird models submitted to KorQuAD2.0

B How expressive is LittleBird Attention?

One might think that the linear bias and pack and unpack attention of BiALiBi cannot compete with the flexibility of trainable positional embeddings. We conducted experiments on this as well, but we determined that it was not necessary for the core content of this paper, so the results were simply inserted into the appendix.

We collected attention distributions for each head of layers when feeding held out data to base-size KoELECTRA and Korean LittleBird to investigate the effect of trainable positional embedding and USWAttention with BiALiBi. For KoELECTRA and Korean LittleBird, sequences of length 512 and 1536 were fed, respectively. The collected distributions were converted into probabilities based on query positions and key positions, and the average was then converted to log scale to be plotted. Figure 4 shows 9 selected heads of various layers from the results. The x-axis in each subfigure represents the query position, and the y-axis represents the key position. The brighter the color, the more intense the attention.

Attention patterns between layer heads are comparable due to the distillation of the attention distribution. In trainable positional embedding, repeating diagonal stripes are a common occurrence. Given the nature of text data, translation-invariance, this is an unusual result that could even be considered noise. In the case of USWAttention with BiALiBi, on the other hand, attention intensity decreases consistently with distance due to a strong inductive bias from linear bias, and it can be seen that the insufficient part is compensated by pack and unpack attention. LittleBird's Attention module, in particular, seems to have learned key patterns (attention to far-away, attention to near-left or near-right, attention to far-left or far-right, etc.) by distilling from trainable positional embeddings. Considering the above results, it is clear that LittleBird's attention provides adequate capacity for natural language modeling.



(a) ELECTRA, Layer 0 - Head 0

(b) LittleBird, Layer 0 - Head 0

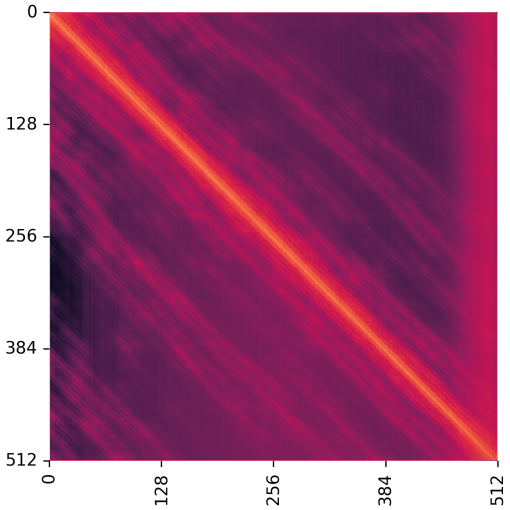
(c) ELECTRA, Layer 0 - Head 9

(d) LittleBird, Layer 0 - Head 9

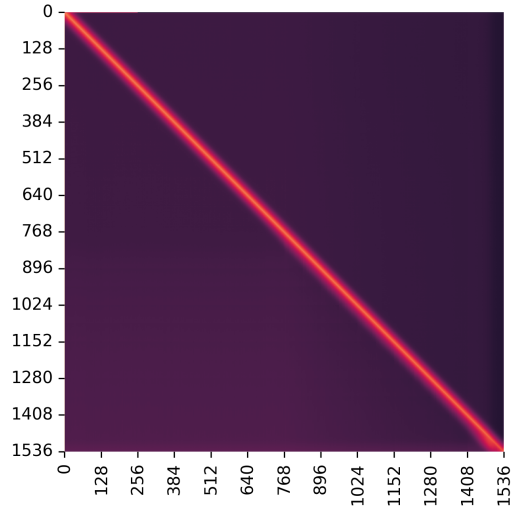
(e) ELECTRA, Layer 1 - Head 3

(f) LittleBird, Layer 1 - Head 3

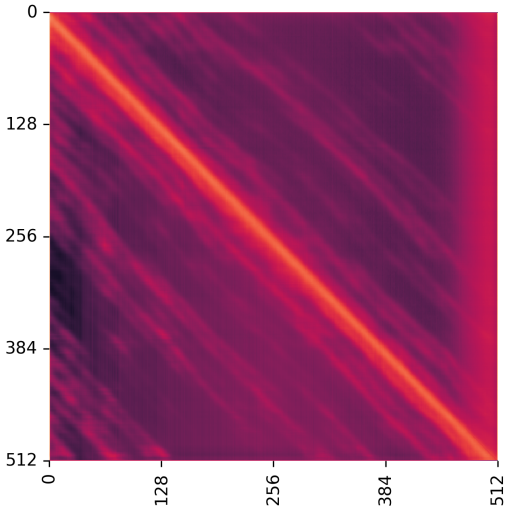
Figure 4: Average attention heatmap of ELECTRA and LittleBird



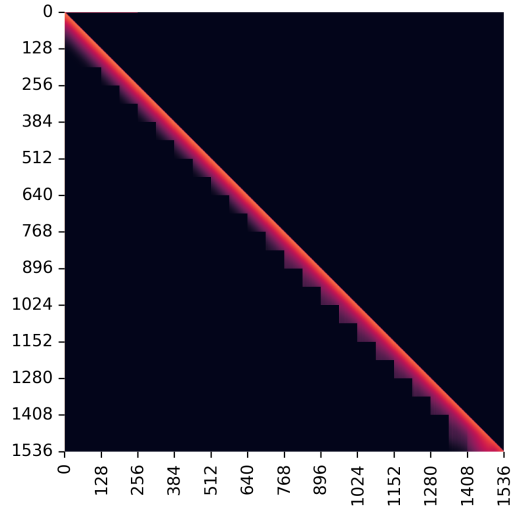
(g) ELECTRA, Layer 1 - Head 4



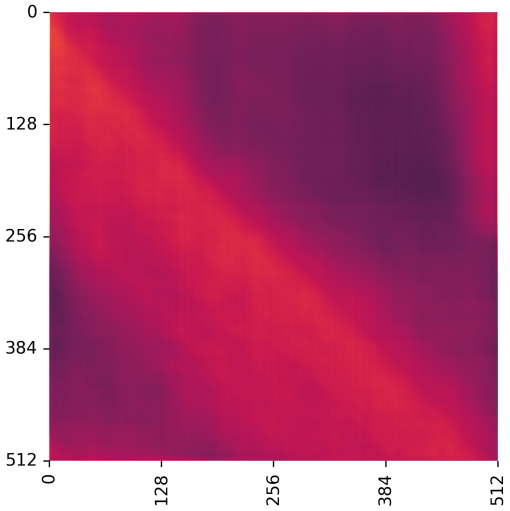
(h) LittleBird, Layer 1 - Head 4



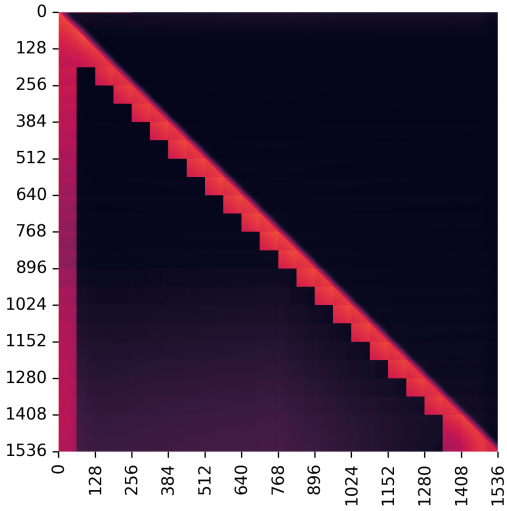
(i) ELECTRA, Layer 2 - Head 4



(j) LittleBird, Layer 2 - Head 4

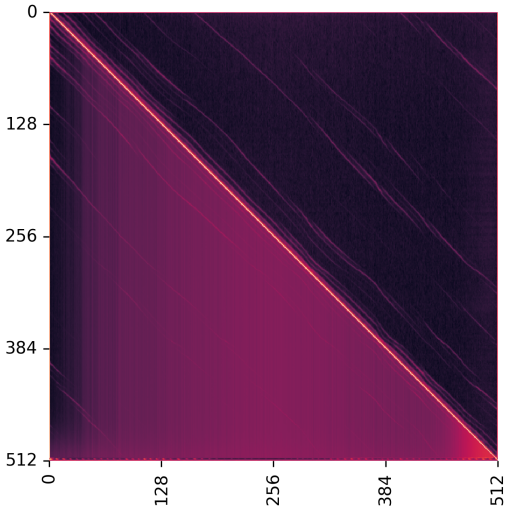


(k) ELECTRA, Layer 4 - Head 4

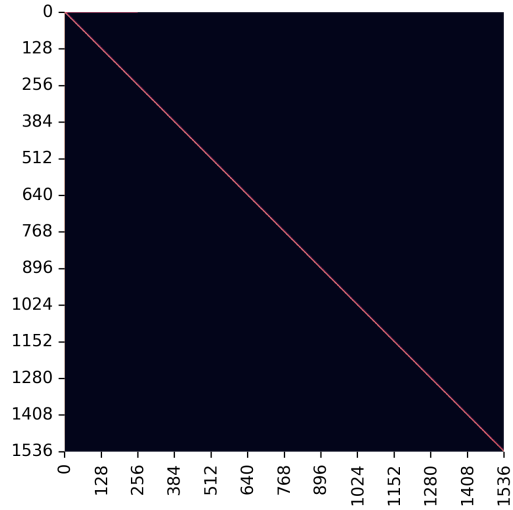


(l) LittleBird, Layer 4 - Head 4

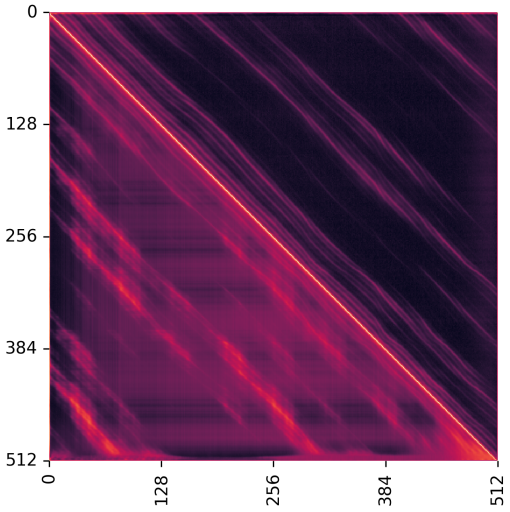
Figure 4: Average attention heatmap of ELECTRA and LittleBird (cont.)



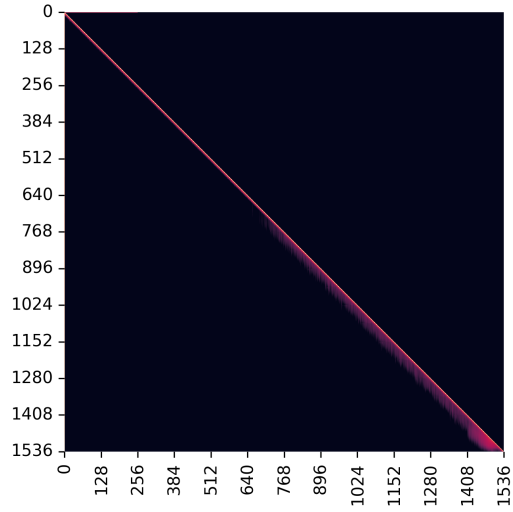
(m) ELECTRA, Layer 7 - Head 2



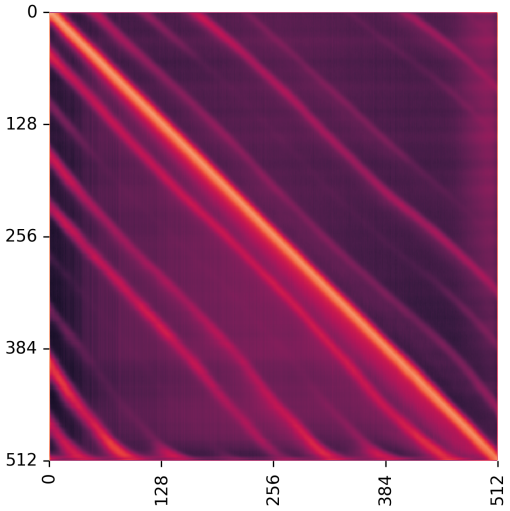
(n) LittleBird, Layer 7 - Head 2



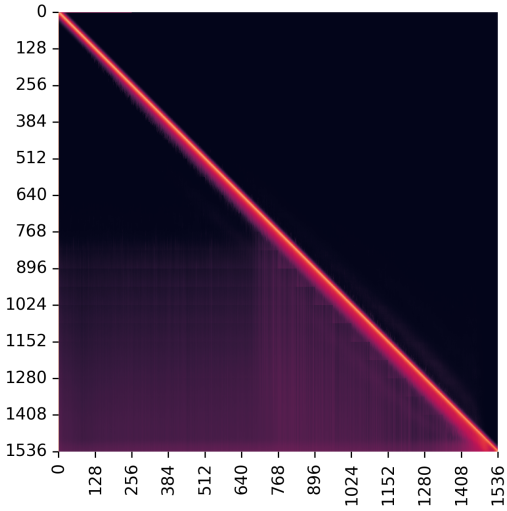
(o) ELECTRA, Layer 8 - Head 0



(p) LittleBird, Layer 8 - Head 0



(q) ELECTRA, Layer 11 - Head 8



(r) LittleBird, Layer 11 - Head 8

Figure 4: Average attention heatmap of ELECTRA and LittleBird (cont.)