Convolutions and Self-Attention: Re-interpreting Relative Positions in Pre-trained Language Models

Tyler A. Chang[1,2],   **Yifan Xu**[1],   **Weijian Xu**[1],   **Zhuowen Tu**[1]
[1]University of California San Diego
[2]Halıcıoğlu Data Science Institute
{tachang, yix081, wex041, ztu}@ucsd.edu

Abstract

In this paper, we detail the relationship between convolutions and self-attention in natural language tasks. We show that relative position embeddings in self-attention layers are equivalent to recently-proposed dynamic lightweight convolutions, and we consider multiple new ways of integrating convolutions into Transformer self-attention. Specifically, we propose composite attention, which unites previous relative position embedding methods under a convolutional framework. We conduct experiments by training BERT with composite attention, finding that convolutions consistently improve performance on multiple downstream tasks, replacing absolute position embeddings. To inform future work, we present results comparing lightweight convolutions, dynamic convolutions, and depthwise-separable convolutions in language model pre-training, considering multiple injection points for convolutions in self-attention layers.

1 Introduction

In recent years, Transformer-based language models have brought dramatic improvements on a wide range of natural language tasks (Brown et al., 2020; Devlin et al., 2019). The central innovation of Transformer architectures is the self-attention mechanism (Vaswani et al., 2017), which has grown beyond NLP, extending into domains ranging from computer vision (Dosovitskiy et al., 2021) and speech recognition (Dong et al., 2018) to reinforcement learning (Parisotto et al., 2020; Touvron et al., 2020).

In computer vision, self-attention and convolutions have been combined to achieve competitive results for image classification (Bello et al., 2019). Similarly, researchers in NLP have begun integrating convolutions into self-attention for natural language tasks. Recent work has shown initial success

adding convolutional modules to self-attention in pre-trained language models (Jiang et al., 2020), or even replacing self-attention entirely with dynamic convolutions (Wu et al., 2019). These successes defy theoretical proofs showing that multi-headed self-attention with relative position embeddings is strictly more expressive than convolution (Cordonnier et al., 2020). To identify why convolutions have been successful in NLP, we seek to isolate the differences between self-attention and convolution in the context of natural language.

In this work, we formalize the relationship between self-attention and convolution in Transformer encoders by generalizing relative position embeddings, and we identify the benefits of each approach for language model pre-training. We show that self-attention is a type of dynamic lightweight convolution, a data-dependent convolution that ties weights across input channels (Wu et al., 2019). Notably, previous methods of encoding relative positions (Shaw et al., 2018; Raffel et al., 2020) are direct implementations of lightweight convolutions. Under our framework, the benefits of convolution come from an ability to capture local position information in sentences.

Then, we propose composite attention, which applies a lightweight convolution that combines previous relative position embedding methods. We find that composite attention sufficiently captures the information provided by many other convolutions. To validate our framework, we train BERT models that integrate self-attention with multiple convolution types, evaluating our models on the GLUE benchmark (Wang et al., 2018). All of our convolutional variants outperform the default model, demonstrating the effectiveness of convolutions in enhancing self-attention for natural language tasks. Our empirical results provide evidence for future research integrating convolutions and self-attention for NLP.
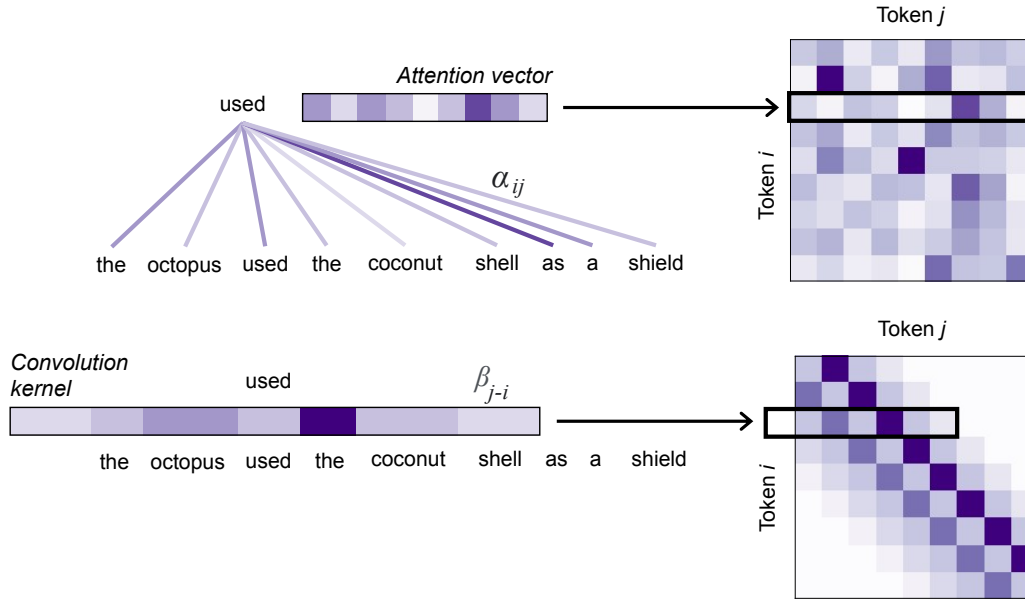
4322

Figure 1: Generating attention maps using standard self-attention (top) and fixed lightweight convolution (bottom). Attention weights α_{ij} are analogous to convolution kernel weights β_{j-i}.

2 Self-attention and lightweight convolutions

First, we outline the relationship between self-attention and convolutions. Specifically, we show that a self-attention operation can be viewed as a dynamic lightweight convolution, a depthwise convolution that ties weights along channels (Wu et al., 2019). We then isolate the differences between self-attention and lightweight convolutions, highlighting the benefits of each approach in language models.

2.1 Self-attention

In a Transformer self-attention layer, inputs $\mathbf{x}_1, ..., \mathbf{x}_n \in \mathbb{R}^d$ are projected to corresponding queries, keys, and values by linear transformations $W^Q, W^K, W^V \in \mathbb{R}^{d \times d_h}$ for each attention head, projecting into the head dimension size d_h. Output vectors $\mathbf{y}_1, ..., \mathbf{y}_n \in \mathbb{R}^d$ are linear combinations of values, concatenating all attention heads. Value weights (before softmaxing) are determined by:

$$\alpha_{ij} = \frac{(\mathbf{x}_i W^Q)(\mathbf{x}_j W^K)^T}{\sqrt{d_h}}. \qquad (1)$$

Intuitively, α_{ij} represents the attention that token i pays to token j, incorporating the value $\mathbf{x}_j W^V$ into the resulting vector $\mathbf{y}_i$. From the attention scores between various tokens i and j, an attention map of α_{ij} is produced (see Figure 1).

2.2 Lightweight convolutions

In contrast, a standard one-dimensional convolution slides a kernel of weights along the input sequence; each feature in each output representation $\mathbf{y}_i$ is a weighted sum of all features (called "channels") in the surrounding $\mathbf{x}_i$. To save parameters, it is common to consider depthwise convolutions where each channel c in $\mathbf{y}_i$ is a weighted sum only of the features in channel c for the surrounding $\mathbf{x}_i$. Formally, each entry of $\mathbf{y}_i$ can be written as:

$$y_{i,c} = \sum_{-k \leq j-i \leq k} \beta_{j-i,c}\, x_{j,c} \qquad (2)$$

where k is the kernel size in each direction. Each scalar $\beta_{j-i,c}$ represents the attention paid to relative position $j - i$ for channel c. To further simplify depthwise convolutions for use in language models, Wu et al. (2019) propose lightweight convolutions, which tie weights $\beta_{j-i,c}$ along all channels c. As a result, the lightweight convolution contains only $2k + 1$ weights, one scalar β_{j-i} for each relative position considered. Then, each $\mathbf{y}_i$ is a linear combination of surrounding $\mathbf{x}_i$:

$$\mathbf{y}_i = \sum_{-k \leq j-i \leq k} \beta_{j-i}\, \mathbf{x}_j \qquad (3)$$

Importantly, we can then consider each β_{j-i} as an attention weight analogous to self-attention, representing the attention that token i pays to token j.

The lightweight convolution produces an attention map of β_{j-i} as visualized in Figure 1.

Finally, furthering the similarity between lightweight convolutions and self-attention, Wu et al. (2019) propose dynamic lightweight convolutions, which dynamically compute relative weights β_{j-i} based on individual input tokens. In other words, each row in Figure 1 has relative weights determined dynamically based on the input token $\mathbf{x}_i$ for that row. Because attentions for relative positions are no longer fixed across rows, the attention map in Figure 1 achieves similar flexibility to standard self-attention.

2.3 Self-attention vs. convolution

We have shown that both self-attention and lightweight convolution compute linear combinations of token representations, but we now isolate the differences between the two approaches. Perhaps most importantly, the two methods assign attention scores α_{ij} and β_{j-i} in fundamentally different ways.

Self-attention computes α_{ij} based on the dot product between query i and key j, ignoring the relative position between i and j. In this way, self-attention layers model interactions exclusively between token representations. If the tokens are arbitrarily shuffled in a standard self-attention layer, the output for each token is unchanged. All position information is injected before the first self-attention layer in the form of absolute position embeddings.

In contrast, dynamic lightweight convolutions assign attention scores directly to relative positions. This allows convolutions to directly integrate relative position information without relying on absolute positions. Thus, convolutions could be better at capturing local information in sentences. However, convolutions alone are limited in their ability to model interactions between tokens because they lack the query-key mechanism central to standard self-attention. In future sections, we consider methods of integrating the two approaches.

3 Integrating lightweight convolutions

Previous work has sought to integrate local information into global self-attention. This can be achieved by restricting the range of self-attention to nearby tokens, or by incorporating relative position information into attention maps (Hofstätter et al., 2020; Raganato et al., 2020; Wei et al., 2021). Notably, Shaw et al. (2018) introduced relative position em-

beddings, which inspired similar embeddings in models such as Transformer-XL and XLNet (Dai et al., 2019; Yang et al., 2019). In this section, we show that several previous methods of encoding relative positions are direct implementations of lightweight convolutions.

3.1 Relative embeddings as lightweight convolutions

First, the simplest way to combine self-attention with lightweight convolution is to generate a standard attention map, then add the attention map generated by a lightweight convolution. Given a fixed lightweight convolution, this results in attention scores as follows:

$$\alpha_{ij} = \frac{(\mathbf{x}_i W^Q)(\mathbf{x}_j W^K)^T}{\sqrt{d_h}} + \beta_{j-i} \qquad (4)$$

This is exactly the relative position term used in T5 (Raffel et al., 2020) and TUPE (Ke et al., 2021).

We further consider a dynamic lightweight convolution, where the β_{j-i} weights are computed by passing the query through a linear feedforward layer $W^C \in \mathbb{R}^{d_h \times (2k+1)}$ (Wu et al., 2019).[1] Because W^C is linear, each weight β_{j-i} is equal to the dot product between the query and the $(j-i)$ column of W^C. We then obtain attention scores:

$$\alpha_{ij} = \frac{(\mathbf{x}_i W^Q)(\mathbf{x}_j W^K)^T}{\sqrt{d_h}} + (\mathbf{x}_i W^Q)(W^C_{j-i})^T$$

If we scale the dynamic lightweight convolution term according to the head dimension size, we obtain precisely the relative embeddings proposed in Shaw et al. (2018):

$$\alpha_{ij} = \frac{(\mathbf{x}_i W^Q)(\mathbf{x}_j W^K + W^C_{j-i})^T}{\sqrt{d_h}} \qquad (5)$$

Under this interpretation, Shaw's relative embeddings are essentially identical to the dynamic lightweight convolutions used in Wu et al. (2019). In both formulations, relative position weights are computed as dot products between the query and a learned relative position embedding. Previous work has considered relative positions in language models independently from convolutions, but our derivations suggest that the underlying mechanisms may be the same.

---

[1] Wu et al. (2019) generate dynamic lightweight convolutions based on the entire query layer (dimension size d). In our work, we generate convolutions based on queries for individual attention heads (dimension size d_h), to be consistent with the relative embeddings in Shaw et al. (2018).

Lightweight convolution type, BERT-small	Params	CoLA	MNLI-m	MNLI-mm	MRPC	QNLI	QQP	RTE	SST	STS	GLUE
No convolution	13.41M	13.9	73.2	71.8	77.9	80.7	74.5	62.0	81.9	79.3	68.4
No convolution + abs position*	13.43M	30.8	76.1	75.9	80.4	78.5	74.4	62.2	85.1	76.8	71.1
Fixed (Raffel et al. 2020)	13.42M	**42.1**	77.2	76.3	83.8	82.7	75.9	64.4	87.1	81.4	74.5
Dynamic (Shaw et al. 2018)	13.43M	39.1	**78.4**	**77.4**	83.8	**83.4**	77.5	64.4	87.3	81.4	74.7
Composite (Equation 6; ours)	13.43M	40.4	78.2	**77.4**	**85.0**	83.3	**77.7**	**64.7**	87.8	82.1	**75.2**

Lightweight convolution type, BERT-base	Params	CoLA	MNLI-m	MNLI-mm	MRPC	QNLI	QQP	RTE	SST	STS	GLUE
No convolution + abs position*	108.82M	50.3	**82.0**	**81.2**	85.0	84.6	78.6	68.9	91.4	84.9	78.5
Fixed (Raffel et al. 2020)	108.73M	50.0	81.5	80.5	**85.6**	**86.0**	78.5	68.9	91.4	84.9	78.6
Dynamic (Shaw et al. 2018)	108.74M	**50.9**	81.6	80.5	84.6	85.3	78.5	69.5	**91.6**	84.8	78.6
Composite (Equation 6; ours)	108.74M	50.4	81.6	80.8	85.4	85.1	**78.7**	**69.7**	91.2	**85.7**	**78.7**

Table 1: GLUE test set performance for models with lightweight convolutions added to self-attention. Columns indicate scores on individual GLUE tasks; the final GLUE score is the average of individual task scores. * denotes the default BERT model.

3.2 Composite attention and lightweight convolution experiments

To validate lightweight convolutions in combination with self-attention, we pre-trained and evaluated BERT-small models (Devlin et al., 2019; Clark et al., 2020) that incorporated lightweight convolutions.

Pre-training   To maximize similarity with Devlin et al. (2019), we pre-trained models on the BookCorpus (Zhu et al., 2015) and WikiText-103 datasets (Merity et al., 2017) using masked language modeling. Small models were pre-trained for 125,000 steps, with batch size 128 and learning rate 0.0003. Full pre-training and fine-tuning details are outlined in Appendix A.1.[2]

Evaluation   Models were evaluated on the GLUE benchmark, a suite of sentence classification tasks including natural language inference (NLI), grammaticality judgments, sentiment classification, and textual similarity (Wang et al., 2018). For each task, we ran ten fine-tuning runs and used the model with the best score on the development set. We report scores on the GLUE test set. Development scores and statistics for all experiments are reported in Appendix A.2.

Models   We trained two baseline models, a default BERT-small with standard absolute position embeddings, and a BERT-small with no position information whatsoever. Then, we trained models with fixed lightweight convolutions (Equation 4;

Raffel et al. 2020), and dynamic lightweight convolutions that generated convolution weights based on each query (i.e. using relative embeddings, Equation 5; Shaw et al. 2018).

Finally, we propose composite attention, which simply adds dynamic lightweight convolutions to fixed lightweight convolutions, resulting in attention scores α_{ij} as follows:

$$\underbrace{\frac{(\mathbf{x}_i W^Q)(\mathbf{x}_j W^K)^T}{\sqrt{d_h}}}_{\text{Self-attention}} + \underbrace{\frac{(\mathbf{x}_i W^Q)(W_{j-i}^C)^T}{\sqrt{d_h}}}_{\substack{\text{Dynamic convolution} \\ \text{(relative embeddings)}}} + \underbrace{\beta_{j-i}}_{\substack{\text{Fixed} \\ \text{convolution}}}$$

(6)

Intuitively, composite attention has the flexibility of dynamic lightweight convolutions, while still allowing models to incorporate relative positions directly through fixed lightweight convolutions. Alternatively, composite attention can be interpreted as adding a fixed bias term to relative position embeddings.

All of our experiments used a convolution kernel size of 17, or eight positions in each direction, a mid-range value that has been found to work well for both relative positions and convolution in language models (Huang et al., 2020; Jiang et al., 2020; Shaw et al., 2018). As in Shaw et al. (2018), relative embeddings W_{j-i}^C shared weights across heads. Unless stated otherwise, models used no absolute position embeddings.

For completeness, we also considered dynamic lightweight convolutions based on the key (as opposed to the query). In contrast to query-based

4325

lightweight convolutions, key-based convolutions allow each token to dictate which relative positions should pay attention to it, rather than dictating which relative positions it should pay attention to. Referring to the visualization in Figure 1, key-based dynamic convolutions correspond to columns instead of rows. These key-based dynamic lightweight convolutions are the same as the relative embeddings proposed in Huang et al. (2020), but they are now formulated as dynamic lightweight convolutions.

3.3 Lightweight convolution results

GLUE test set results are presented in Table 1.

Lightweight convolutions consistently improved performance. Notably, even the fixed lightweight convolution was sufficient to replace absolute position embeddings, outperforming the default BERT-small model. This indicates that even naïve sampling from nearby tokens can be beneficial to language model performance.

Dynamic convolutions provided further improvements. When the lightweight convolutions were generated dynamically based on token queries, the models outperformed the default model by even larger margins. This improvement over fixed lightweight convolutions suggests that different tokens find it useful to generate different lightweight convolutions, paying attention to different relative positions in a sentence.

Composite attention performed the best. Combining fixed lightweight convolutions with dynamic lightweight convolutions proved an effective strategy for encoding relative positions. Although composite attention is simply a combination of Shaw et al. (2018) and Raffel et al. (2020)'s relative position embeddings, it validates convolution as a viable method of encoding relative positions in self-attention.

Key-based dynamic convolutions provided no additional benefit. When we generated an additional lightweight convolution based on keys, the model performed worse than composite attention alone (GLUE 74.0 compared to 75.2). This result clarifies the findings of Huang et al. (2020), who reported only small improvements from query and key-based relative position embeddings for a subset of the GLUE tasks.
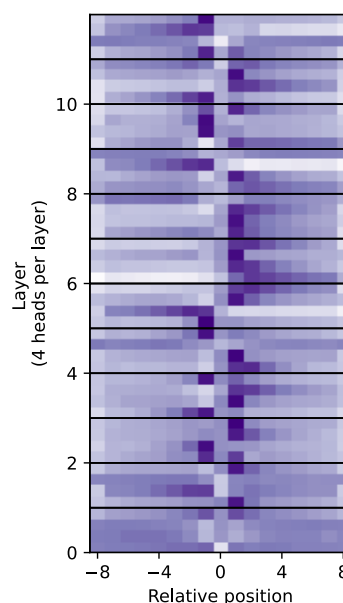

Figure 2: Learned convolution kernel weights β_{j-i} for the fixed lightweight convolution (Equation 4).

Grammaticality judgments were particularly sensitive to position information. On the CoLA task (the corpus of linguistic acceptability; Warstadt et al. 2019), there was a dramatic performance drop when absolute position embeddings were removed. However, when any type of lightweight convolution was added, performance improved even over the baseline established by absolute positions. The pronounced effects of local position information on the CoLA task support the intuitive hypothesis that local dependencies are particularly important for grammaticality judgments. This result also suggests that convolutions could be beneficial to more local tasks (e.g. token-level tasks) along with sentence classification tasks.

3.4 Interpreting lightweight convolutions

To better understand how lightweight convolutions improve language models, we visualized the learned lightweight convolution kernel weights in Figure 2. Qualitatively, the kernels exhibited specific types of patterns:

- Paying particular attention to the previous or next token.

- Paying graded attention either to past or future tokens, dictated by how far the target token is from the present token.

These observations support the assumption that nearby tokens are relevant to the interpretation of the current token. They also align with the findings

of Voita et al. (2019), who identified "positional" attention heads that focus primarily on the next or previous token. From this perspective, lightweight convolutions allow language models to explicitly represent nearby tokens' positions.

Interestingly, we also found that some kernels paid fairly uniform attention to all tokens, even decreasing attention to nearby and adjacent tokens. It is likely that these attention heads focused on more global information, relying on the query-key attention mechanism rather than the convolution.

3.5 BERT-base models

To thoroughly assess the impact of composite attention on pre-trained language models, we trained full-sized BERT models for 1M steps each, replicating our BERT-small experiments. Pre-training details are outlined in Appendix A.1.

Results are presented in Table 1. Differences between models decreased substantially for full sized models, and the relative performances of different approaches varied across tasks. Our results suggest that relative position information is more useful for smaller or more data-limited models; extending the benefits of convolutions robustly from small models to larger models is an important direction for future research. That said, even in the larger models, composite attention slightly outperformed the other position embedding methods in overall GLUE score. Our results demonstrate that convolutions can perform at least on par with absolute position embeddings even in larger models.

4 Non-lightweight convolutions

The previous section found that lightweight convolutions consistently improved pre-trained language model performance. Next, we investigated whether the additional flexibility of non-lightweight convolutions could provide additional benefits. Specifically, we considered convolutions that were fixed but non-lightweight. In other words, convolution weights were fixed regardless of the input query, but weights were not tied across channels, equivalent to a standard depthwise convolution. We only considered fixed depthwise convolutions because under existing frameworks, dynamic depthwise convolutions would introduce large numbers of parameters.

To implement depthwise convolutions, we added a convolution term identical to the fixed lightweight convolution in Equation 4, except that β_{j-i} was
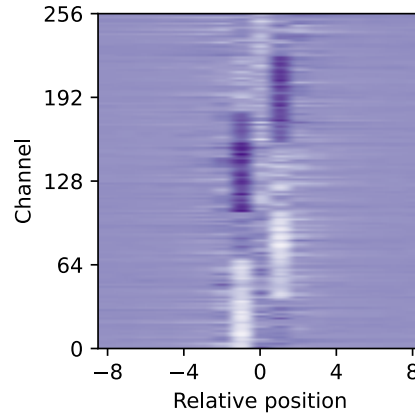

Figure 3: Learned convolution kernel weights $\beta_{j-i,c}$ (Equation 7) for the depthwise convolution in the deepest attention layer. Channels correspond to the 256 features in each token representation. Channels are sorted such that kernels differentiating the previous and next token are grouped together.

learned separately for each feature channel:[3]

$$\alpha_{ij,c} = \frac{(\mathbf{x}_i W^Q)(\mathbf{x}_j W^K)^T}{\sqrt{d_h}} + \beta_{j-i,c} \qquad (7)$$

This is equivalent to adding a depthwise convolution of the token values to the standard self-attention output.

4.1 Non-lightweight convolution experiments

We ran experiments using the same setup as the lightweight convolution experiments in Section 3.2. To compare the effects of dynamic lightweight convolutions (e.g. composite attention) and non-lightweight (depthwise) convolutions, we trained models using each possible combination of the two convolutions. Results are presented in Table 2.

Depthwise convolutions were less effective than lightweight convolutions. As with lightweight convolutions, the depthwise convolutions effectively replaced absolute position embeddings, outperforming the default model. However, fixed depthwise convolutions performed worse than fixed lightweight convolutions on the majority of tasks. This indicates that flexibility across channels is not critical to the success of convolutions in language models.

---

[3]For computational efficiency, we applied the softmax to the attention scores prior to adding the convolution term $\beta_{j-i,c}$, to avoid computing softmax scores separately for each individual channel. Softmax is not commonly applied in depthwise convolutions.

Convolutions	Params	CoLA	MNLI-m	MNLI-mm	MRPC	QNLI	QQP	RTE	SST	STS	GLUE
No convolution + abs position*	13.43M	30.8	76.1	75.9	80.4	78.5	74.4	62.2	85.1	76.8	71.1
Composite (Equation 6)	13.43M	**40.4**	**78.2**	**77.4**	**85.0**	83.3	**77.7**	64.7	**87.8**	82.1	**75.2**
Fixed depthwise	13.47M	36.9	77.6	76.1	80.6	81.9	76.4	64.5	87.5	79.7	73.5
Fixed depthwise + composite	13.48M	38.0	77.4	76.3	82.8	**83.7**	77.7	**65.3**	87.3	**82.3**	74.5

Table 2: GLUE test set performance for BERT-small models with added depthwise convolutions and composite attention. * denotes the default BERT-small model.

No composite attention			
Query/Key	**Value**	**Params**	**GLUE**
Linear	Linear	13.43M	*71.1
Convolution	Linear	13.53M	71.9
Linear	Convolution	13.47M	73.4
Convolution	Convolution	13.58M	72.0

+Composite attention			
Query/Key	**Value**	**Params**	**GLUE**
Linear	Linear	13.43M	**75.2**
Convolution	Linear	13.54M	74.5
Linear	Convolution	13.48M	73.9
Convolution	Convolution	13.59M	74.0

Table 3: BERT-small performance on the GLUE test set when replacing queries, keys, and values with depthwise-separable convolutions for half of the attention heads. * denotes the use of absolute position embeddings in the default BERT-small model.

Composite attention already provided the necessary flexibility. Composite attention outperformed the fixed depthwise convolutions; even when composite attention was combined with depthwise convolutions, there was no overall improvement over composite attention alone. This suggests that in the context of language, dynamic lightweight convolutions efficiently encode any local position information provided by depthwise convolutions.

Depthwise convolutions differentiated previous and next tokens. In previous sections, we found that lightweight convolution kernels often pay attention specifically to adjacent tokens. As can be seen in Figure 3, this result was even more pronounced in depthwise convolutions, with individual channels focusing on the previous or next token. Interestingly, other channels specifically directed attention away from adjacent tokens. This indicates that the relevant information about next and previous tokens can be compressed into a subset of the feature channels, freeing other channels to consider more distant or position-independent information.

5 Convolutional queries, keys, and values

Improvements over the non-convolutional baselines indicate that convolutions are beneficial to language model pre-training, serving as replacements for absolute position embeddings. Our previous experiments applied different types of convolutions to self-attention values. To take this result one step

further, we replaced the linear query, key, and value projections themselves with convolutional layers.

Intuitively, applying convolutions before self-attention induces even more mixing of token representations. If convolutions are built into every query, key, and value, then it becomes impossible for a token i to pay attention to a single token j without also incorporating information about tokens surrounding token j.

5.1 Convolutional Q, K, V experiments

As in Sections 3.2 and 4.1, we ran experiments on BERT-small. We replaced the query, key and value projections with depthwise-separable convolutions in half of the self-attention heads.[4] This aligns with previous work in which only half of the output dimensions for each token were generated using convolutions (Jiang et al., 2020). Indeed, our initial explorations found that it was more effective to replace the linear projections in only half, not all, the attention heads.

Then, we considered whether convolutions from previous experiments provided additional benefits over convolutional queries, keys, and values. To test this, we trained BERT-small models with composite attention (Equation 6), adding convolutional queries, keys, and values.

---

[4]Depthwise-separable convolutions are a common way to save convolution parameters. A depthwise convolution is applied first, applying an independent convolution for each channel. Then, a pointwise convolution (i.e. a feedforward layer) mixes the channels to produce the final output.

5.2 Convolutional Q, K, V results

Results are presented in Table 3. Similar to our previous convolution experiments, all convolutional replacements successfully outperformed the default model. These results strongly support the conclusion that convolutions are a viable method of encoding positional information for language tasks.

However, all convolutional replacements for queries, keys, and values slightly decreased the performance of models using composite attention. Convolutional values in particular were effective in models without composite attention, but they slightly decreased performance in models that already incorporated such lightweight convolutions. We conclude that although convolutions can benefit models by adding local position information, there is a limit to how much local mixing should be done. It is sufficient to apply convolutions to token values on top of self-attention; additional convolutional layers applied before the self-attention map enforce unnecessary mixing of token representations.

6 Discussion

Our results demonstrate that convolutions provide consistent benefits to pre-trained language models. Our proposed composite attention mechanism combines previous relative position embedding methods, showing that convolutions can effectively compensate for the lack of local position information in Transformer models.

6.1 Related work

Our work unites and builds upon previous work using convolutions and relative positions in Transformers. We adopted the relative embeddings from Shaw et al. (2018) and Huang et al. (2020), showing that these embeddings are equivalent to the dynamic lightweight convolutions in Wu et al. (2019). Combining these dynamic lightweight convolutions with fixed lightweight convolutions (equivalent to the relative position terms in Raffel et al. 2020), we studied relative embeddings under the framework of convolution integrated with self-attention. As far as we are aware, our work is the first to holistically compare relative positions, convolutions, and self-attention in language models.

Building upon dynamic lightweight convolutions, recent work has incorporated both depthwise-separable and dynamic lightweight convolutions in pre-trained language models. Jiang et al. (2020) proposed ConvBERT, which adds a convolutional module alongside the standard self-attention mechanism in BERT. ConvBERT's convolutional module consists of a depthwise-separable convolution combining with a query to generate a dynamic lightweight convolution. Under our integrated framework, this is analogous to the model which uses depthwise-separable convolutions for queries and keys, using composite attention as a query-based dynamic lightweight convolution (see Table 3). To make this comparison concrete, we trained a ConvBERT-small model using the same setup as our experiments. Indeed, the analogous model under our framework outperformed ConvBERT-small (GLUE score 74.5 compared to 70.3). Details for the ConvBERT comparison can be found in Appendix A.3.

Finally, recent work has proved theoretical relationships between self-attention and convolution. Cordonnier et al. (2020) showed that given enough self-attention heads, self-attention weights can express any convolution; in fact, they showed that self-attention layers often learn such convolutional structures when trained on vision tasks. However, this theoretical equivalence does not explain convolution-based improvements for Transformers in language tasks. To clarify the relationship between self-attention and convolution in language, our work characterizes self-attention as a type of dynamic lightweight convolution. By establishing a per-parameter equivalence between relative position embeddings and Wu's dynamic lightweight convolutions, we provide a concrete foundation where self-attention and convolution are used together in practice.

7 Conclusion

In this work, we formalized the relationship between self-attention and convolution. We proposed composite attention, which combines self-attention with lightweight convolution, uniting previous approaches to relative positions. Our formulation and empirical results demonstrate that convolutions can improve self-attention by providing local position information in sentences, capable of replacing absolute position embeddings entirely.

Our findings provide a solid foundation from which to study convolutions and self-attention in language tasks. The spatially-oriented nature of convolutional neural networks translates directly into positional information in language. As vision and language researchers strive towards common

deep learning architectures, it is important to recognize how architectures for vision tasks can be adapted to linguistic domains.

Acknowledgments

References

Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc Le. 2019. Attention augmented convolutional networks. In *International Conference on Computer Vision*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th Conference on Neural Information Processing Systems*.

Kevin Clark, Minh-Thang Luong, Quoc Le, and Christopher Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *Proceedings of the International Conference on Learning Representations*.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. 2020. On the relationship between self-attention and convolutional layers. In *Proceedings of the International Conference on Learning Representations*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. 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.

Linhao Dong, Shuang Xu, and Bo Xu. 2018. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5884–5888.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proceedings of the International Conference on Learning Representations*.

Sebastian Hofstätter, Hamed Zamani, Bhaskar Mitra, Nick Craswell, and Allan Hanbury. 2020. Local self-attention over long text for efficient document retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA. Association for Computing Machinery.

Zhiheng Huang, Davis Liang, Peng Xu, and Bing Xiang. 2020. Improve transformer models with better relative position embeddings. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3327–3335, Online. Association for Computational Linguistics.

Zihang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. 2020. ConvBERT: Improving BERT with span-based dynamic convolution. In *Proceedings of the 34th Conference on Neural Information Processing Systems*.

Guolin Ke, Di He, and Tie-Yan Liu. 2021. Rethinking positional encoding in language pre-training. In *Proceedings of the International Conference on Learning Representations*.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proceedings of the Fifth International Conference on Learning Representations*.

Emilio Parisotto, Francis Song, Jack Rae, Razvan Pascanu, Caglar Gulcehre, Siddhant Jayakumar, Max Jaderberg, Raphaël Lopez Kaufman, Aidan Clark, Seb Noury, Matthew Botvinick, Nicolas Heess, and Raia Hadsell. 2020. Stabilizing transformers for reinforcement learning. In *Proceedings of the International Conference on Machine Learning*.

Jason Phang, Thibault Févry, and Samuel Bowman. 2018. Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. 2020. Fixed encoder self-attention patterns in transformer-based machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 556–568, 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.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st Conference on Neural Information Processing Systems*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. 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*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Wei Wei, Zanbo Wang, Xianling Mao, Guangyou Zhou, Pan Zhou, and Sheng Jiang. 2021. Position-aware self-attention based neural sequence labeling. *Pattern Recognition*, 110.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *Proceedings of the Seventh International Conference on Learning Representations*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ Salakhutdinov, and Quoc Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision*, pages 19–27.

Hyperparameter	Small	Base
Layers	12	12
Hidden size	256	768
Intermediate hidden size	1024	3072
Attention heads	4	12
Attention head size	64	64
Embedding size	128	768
Vocab size	30004	30004
Max sequence length	128	128
Mask proportion	0.15	0.15
Learning rate decay	Linear	Linear
Warmup steps	10000	10000
Learning rate	3e-4	1e-4
Adam ϵ	1e-6	1e-6
Adam β_1	0.9	0.9
Adam β_2	0.999	0.999
Attention dropout	0.1	0.1
Dropout	0.1	0.1
Weight decay	0.01	0.01
Batch size	128	256
Train steps	125K	1M

Table 4: Pre-training hyperparameters.

A   Appendix

A.1   Pre-training and fine-tuning details

BERT models (Devlin et al. 2019; Clark et al. 2020) were pre-trained on the BookCorpus (Zhu et al., 2015) and WikiText-103 datasets (Merity

Hyperparameter	Value
Learning rate decay	Linear
Warmup steps	10% of total
Learning rate	1e-4 for QNLI or base-size
	3e-4 otherwise
Adam ϵ	1e-6
Adam β_1	0.9
Adam β_2	0.999
Attention dropout	0.1
Dropout	0.1
Weight decay	0
Batch size	128 for MNLI/QQP
	32 otherwise
Train steps	10 epochs for RTE/STS
	4 epochs for MNLI/QQP
	3 epochs otherwise

Table 5: Fine-tuning hyperparameters. We used intermediate task training for RTE, STS, and MRPC, initializing from a checkpoint fine-tuned on the MNLI task (Clark et al. 2020; Phang et al. 2018).

et al., 2017) using masked language modeling. Pre-training examples were formatted as sentence pairs without the next sentence prediction objective. In total, our dataset consisted of 31M unique sentence pairs.[5] Sentences were tokenized by training an uncased SentencePiece tokenizer (Kudo and Richardson, 2018), and input and output token embeddings were tied during pre-training. Models were evaluated on the GLUE benchmark (Wang et al., 2018). Including ten fine-tuning runs for each GLUE task, each BERT-small model took about 24 hours to train on two Titan Xp GPUs. Each BERT-base model took about 16 days to train on 8 GPUs. Pre-training hyperparameters are listed in Table 4, and fine-tuning hyperparameters are listed in Table 5. Hyperparameters are based on those used in Clark et al. (2020) and Devlin et al. (2019).

A.2   GLUE development results

Results for each model on the GLUE development set are reported in Table 6. We report averages over ten fine-tuning runs for each task, including standard errors of the mean. Each overall GLUE score was computed as the average of individual task scores; we computed GLUE score averages and standard errors over ten GLUE scores, corresponding to the ten fine-tuning runs. We note that development scores were generally higher than test scores due to differences between the test and

training distributions (Wang et al., 2018).

A.3   Detailed ConvBERT comparison

ConvBERT adds a convolutional module alongside the standard self-attention mechanism in BERT (Jiang et al., 2020). ConvBERT uses half the number of standard self-attention heads, using convolutional modules for the other half. In each convolutional module, a depthwise-separable convolution is multiplied pointwise with the query in the corresponding self-attention head. This convolutional query is fed into a linear layer to generate a dynamic lightweight convolution.

Under our framework, the analogous model replaces half of the queries and keys with depthwise-separable convolutions and uses composite attention (a query-based dynamic lightweight convolution; see Table 3 in the full paper). In both models (ConvBERT and our own), half of the attention heads use a convolutional query. Additionally, in both models, the convolutional query is used to generate a dynamic lightweight convolution.

However, in our model, the dynamic lightweight convolution (in this case, composite attention) is used for all attention heads, not just the convolutional heads. Furthermore, our convolutional heads still use a self-attention mechanism along with the dynamic lightweight convolutions, by generating convolutional keys. In this way, our model adds convolutions to ConvBERT's self-attention heads, and adds self-attention to ConvBERT's convolutional heads.

Then, we investigated whether the separate self-attention and convolutional modules in ConvBERT provide any benefit over our integrated convolution and self-attention. We trained a ConvBERT-small model using the same pre-training setup as our BERT-small experiments, comparing performance to the analogous model under our framework. Results are shown in Table 7. Indeed, integrated convolutions and self-attention outperformed ConvBERT-small, using only 3% more parameters.

---

[5]Because BERT-small models were only trained for 125,000 steps with batch size 128, small models were trained on 16M sentence pairs.

Convolution type, BERT-small	Params	CoLA	MNLI-m	MNLI-mm	MRPC	QNLI
No convolution	13.41M	7.0 ± 2.4	73.0 ± 0.1	73.0 ± 0.1	80.9 ± 0.4	80.1 ± 0.2
No convolution + abs position[*]	13.43M	33.5 ± 0.4	75.8 ± 0.1	76.1 ± 0.1	83.3 ± 0.4	78.2 ± 0.3
Fixed lightweight (Raffel et al. 2020)	13.42M	38.3 ± 0.8	77.2 ± 0.1	77.2 ± 0.1	84.0 ± 0.5	82.1 ± 0.1
Dynamic lightweight (Shaw et al. 2018)	13.43M	38.4 ± 0.7	$\mathbf{77.9} \pm 0.1$	77.6 ± 0.1	85.6 ± 0.5	82.8 ± 0.1
Composite (Equation 6)	13.43M	$\mathbf{40.9} \pm 0.7$	$\mathbf{77.9} \pm 0.1$	78.0 ± 0.1	86.2 ± 0.3	83.0 ± 0.1
Composite + key-based dynamic	13.44M	40.0 ± 0.6	$\mathbf{77.9} \pm 0.1$	77.7 ± 0.1	$\mathbf{86.3} \pm 0.3$	83.3 ± 0.1
Fixed depthwise	13.47M	38.0 ± 0.6	76.9 ± 0.0	76.8 ± 0.1	82.8 ± 0.5	81.9 ± 0.1
Composite + fixed depthwise	13.48M	40.4 ± 0.7	77.2 ± 0.1	77.4 ± 0.1	85.0 ± 0.3	83.3 ± 0.1
Convolutional QK	13.53M	33.4 ± 0.4	76.3 ± 0.1	76.4 ± 0.1	83.3 ± 0.2	81.3 ± 0.2
Convolutional value	13.47M	34.7 ± 0.9	76.2 ± 0.0	76.6 ± 0.1	83.4 ± 0.4	82.4 ± 0.1
Convolutional QKV	13.58M	31.9 ± 0.7	76.3 ± 0.1	76.3 ± 0.1	83.7 ± 0.4	80.4 ± 0.2
Composite + convolutional QK	13.54M	39.3 ± 0.8	77.4 ± 0.1	77.2 ± 0.1	85.4 ± 0.3	81.9 ± 0.1
Composite + convolutional value	13.48M	37.9 ± 0.7	77.8 ± 0.1	$\mathbf{78.1} \pm 0.1$	85.6 ± 0.4	$\mathbf{83.6} \pm 0.1$
Composite + convolutional QKV	13.59M	38.2 ± 1.0	77.4 ± 0.1	77.3 ± 0.1	85.3 ± 0.4	82.8 ± 0.1
ConvBERT	13.09M	33.3 ± 1.5	76.7 ± 0.1	76.8 ± 0.1	83.9 ± 0.5	77.1 ± 0.8
Convolution type, BERT-base						
No convolution + abs position[*]	108.82M	57.6 ± 0.6	$\mathbf{82.0} \pm 0.1$	$\mathbf{81.9} \pm 0.1$	$\mathbf{88.4} \pm 0.2$	84.7 ± 0.3
Fixed lightweight (Raffel et al. 2020)	108.73M	$\mathbf{58.9} \pm 0.5$	81.9 ± 0.1	81.6 ± 0.1	87.7 ± 0.3	$\mathbf{86.2} \pm 0.1$
Dynamic lightweight (Shaw et al. 2018)	108.74M	58.4 ± 0.5	81.8 ± 0.1	81.8 ± 0.1	86.7 ± 0.4	85.6 ± 0.2
Composite (Equation 6)	108.74M	58.5 ± 0.5	81.9 ± 0.1	81.6 ± 0.1	86.0 ± 1.2	85.0 ± 0.3

Convolution type, BERT-small	QQP	RTE	SST	STS	**GLUE**
No convolution	84.4 ± 0.1	61.0 ± 0.5	80.9 ± 0.9	83.7 ± 0.1	69.3 ± 0.3
No convolution + abs position[*]	84.9 ± 0.0	64.4 ± 0.5	85.0 ± 0.2	82.4 ± 0.1	73.7 ± 0.1
Fixed lightweight (Raffel et al. 2020)	86.2 ± 0.0	64.7 ± 0.9	86.9 ± 0.2	85.2 ± 0.1	75.7 ± 0.2
Dynamic lightweight (Shaw et al. 2018)	87.2 ± 0.0	65.1 ± 0.9	86.8 ± 0.2	85.6 ± 0.1	76.3 ± 0.1
Composite (Equation 6)	87.3 ± 0.0	66.1 ± 0.7	86.9 ± 0.1	85.9 ± 0.1	$\mathbf{76.9} \pm 0.1$
Composite + key-based dynamic	87.4 ± 0.0	$\mathbf{66.3} \pm 0.4$	86.5 ± 0.3	86.1 ± 0.2	76.8 ± 0.1
Fixed depthwise	86.1 ± 0.1	64.2 ± 0.7	87.2 ± 0.2	84.4 ± 0.1	75.4 ± 0.1
Composite + fixed depthwise	87.3 ± 0.0	63.5 ± 0.8	87.1 ± 0.2	86.1 ± 0.1	76.4 ± 0.1
Convolutional QK	85.1 ± 0.1	63.0 ± 1.0	86.1 ± 0.2	84.5 ± 0.1	74.4 ± 0.1
Convolutional value	86.6 ± 0.0	65.2 ± 0.7	87.2 ± 0.3	85.0 ± 0.1	75.2 ± 0.1
Convolutional QKV	84.6 ± 0.2	66.1 ± 0.9	86.4 ± 0.1	84.4 ± 0.1	74.4 ± 0.1
Composite + convolutional QK	86.7 ± 0.0	64.0 ± 0.9	$\mathbf{87.5} \pm 0.2$	85.7 ± 0.1	76.1 ± 0.1
Composite + convolutional value	$\mathbf{87.5} \pm 0.0$	65.1 ± 0.5	$\mathbf{87.5} \pm 0.1$	$\mathbf{86.4} \pm 0.1$	76.6 ± 0.1
Composite + convolutional QKV	87.0 ± 0.0	64.9 ± 0.8	86.9 ± 0.1	85.9 ± 0.1	76.2 ± 0.2
ConvBERT	85.1 ± 0.1	64.6 ± 0.5	86.3 ± 0.3	84.0 ± 0.2	74.2 ± 0.3
Convolution type, BERT-base					
No convolution + abs position[*]	88.7 ± 0.0	69.9 ± 0.5	90.4 ± 0.1	$\mathbf{88.4} \pm 0.1$	81.0 ± 0.2
Fixed lightweight (Raffel et al. 2020)	$\mathbf{88.8} \pm 0.0$	70.9 ± 0.7	90.8 ± 0.1	88.1 ± 0.1	$\mathbf{81.3} \pm 0.2$
Dynamic lightweight (Shaw et al. 2018)	88.7 ± 0.0	70.6 ± 0.6	$\mathbf{91.1} \pm 0.1$	87.7 ± 0.3	81.1 ± 0.2
Composite (Equation 6)	88.7 ± 0.0	$\mathbf{71.0} \pm 0.7$	90.5 ± 0.1	$\mathbf{88.4} \pm 0.1$	81.2 ± 0.2

Table 6: GLUE development set scores for each model described in the main paper, reporting averages and standard errors of the mean over ten fine-tuning runs for each task. [*] denotes the default BERT model.

Model, BERT-small	Params	CoLA	MNLI-m	MNLI-mm	MRPC	QNLI	QQP	RTE	SST	STS	**GLUE**
ConvBERT	13.1M	25.5	75.4	73.9	79.7	76.0	74.7	64.3	85.6	77.9	70.3
Integrated convolutions and self-attention (ours)	13.5M	**37.9**	**77.5**	**76.6**	**83.7**	**83.1**	**76.6**	**65.3**	**88.7**	**81.1**	**74.5**

Table 7: Comparison between ConvBERT-small and the analogous model under our framework, reporting GLUE test set results.