On Difficulties of Attention Factorization through Shared Memory

Uladzislau Yorsh[1], **Ondřej Bojar**[1], **Martin Holeňa**[2], **David Herel**[2]
[1]Charles University in Prague, [2]Czech Technical University in Prague
vlad.yorsh@mff.cuni.cz, hereldav@fel.cvut.cz

Abstract

Transformers have revolutionized deep learning in numerous fields, including natural language processing, computer vision, and audio processing. Their strength lies in their attention mechanism, which allows for the discovering of complex input relationships. However, this mechanism's quadratic time and memory complexity poses challenges for larger inputs. Researchers are now investigating models like Linear Unified Nested Attention (Luna) or Memory Augmented Transformer, which leverage external learnable memory to either reduce the attention computation complexity down to linear, or to propagate information between chunks in chunk-wise processing. Our findings challenge the conventional thinking on these models, revealing that interfacing with the memory directly through an attention operation is suboptimal, and that the performance may be considerably improved by filtering the input signal before communicating with memory.

1 Introduction & Related Work

In the era of big data and natural language processing, handling long-form text is crucial. Transformers (Vaswani et al., 2017) have shown promise in some tasks, but they do not scale well with longer inputs due to their quadratic time and memory complexity inherent in their attention framework. This challenge has given rise to multiple approaches designed to handle sequences exceeding typical input lengths, including attention reformulation for efficient computing (Rabe & Staats, 2021; Dao et al., 2022), exploration of weight sharing techniques (Dehghani et al., 2018; Raffel et al., 2019), heavy use of quantization (Shen et al., 2019) or replacing the attention operation itself with a faster alternative.

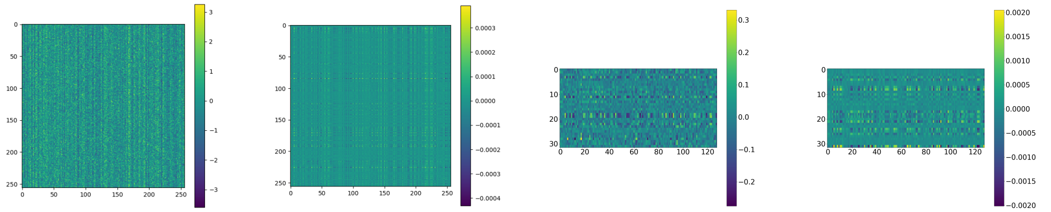
In the present work, we focus on designs that alter the Transformer architecture to lower the computational demands by leveraging an external memory in the form of a set of learnable vectors. Models like Linear Unified Nested Attention (Luna; Ma et al., 2021) or Perceiver (Jaegle et al., 2021) use it to factorize an attention operation into a sequence of attentions with a linear complexity, while the Memory Augmented Transformer (Wu et al., 2022) processes long inputs chunk-by-chunk using the memory as a hidden state to carry information between chunks. While these models adopt different perspectives on long input processing, they all leverage the attention mechanism as an interface for communication between the input and memory. The latter can be used as a convenient fixed-length dense representation of sparse inputs such as texts.

Given the properties of the attention operation, we discover the phenomenon which does not allow to utilize multiple memory cells properly, which we call *memory degradation*. Overcoming it may significantly improve the performance of the named models, and we propose several tweaks which lead to noticeable performance gains on the considered benchmarks.

2 Methodology and Experimental Setup

After inspecting the Luna and Set Transformer (Lee et al., 2019) memory states during training, we have found out that they tend to converge to a single or a small number of points. That means that the memory is not being used completely, and this conclusion is being supported by the experimental results from the Luna paper (Ma et al., 2021), where the differences in performance between memory

Code: https://github.com/vladyorsh/lra_efficient_transformers



(a) Luna memory (b) Luna memory grad (c) Set Transformer memory (d) Set Transformer memory grad

Figure 1: Memory degradation illustrated. The horizontal axis is the feature dimension, the vertical one is the memory vector index, and the color indicates element values. Despite that memory matrices (learned parameters fed to the first model block as a memory input P on the Figure 3) were randomly initialized, during training they converge to a small number of unique vectors, see the vertical stripes on a) and b) and similar blocks on c) and d).

of sizes 16 and 256 are marginal. Unlike in vanilla Transformers, the attention logits of an input-memory attention matrix remain at relatively high entropy during training—the attention scores are distributed almost uniformly, and the resulting value vectors are similar for all input tokens.

As an attempt to counter the issue of the memory degradation, we experiment with two techniques: filtering input before attending to the memory and lowering the softmax temperature (see Appendices B and C for the detailed description). The first technique consists in applying a convolution or a pooling on the keys and values in the "packing" attention, leaving only the relevant signal which can form a better new memory representation after the attention. The second technique consists in replacing the $d^{-\frac{1}{2}}$ normalizing term in the attention equation with its learnable logarithm, initialized at zero. We apply both techniques only on "packing" attention, and refer to our model as ConvLuna.

We test our findings on the subset of the Long Range Arena (LRA; Tay et al., 2020) benchmark. We do not report the score for the Pathfinding task, because we have not managed to replicate the results of Ma et al. (2021). We provide a more detailed description of the experimental setup in Appendix D and an ablation study results in Appendix E.

3 Results and Conclusion

Our work demonstrates notable performance improvements on several kinds of classification tasks through our proposed methods. We also find out that models with even a single memory cell outperform the standard Transformer model. These results highlight the inefficiency of the traditional direct input-memory interfacing through attention, and suggest that pre-filtering inputs before interacting with memory can lead to significant gains. However, throughout our experiments we were not able to achieve statistically significant impact of the memory size on the performance, which may indicate a substantial scope for future enhancements in efficiency and effectiveness of the architectures leveraging the external memory in a form of learnable vectors.

Model	Classification ↑	Matching ↑	ListOps ↑	CIFAR-10 ↑	Average ↑
Transformer	64.27	57.46	36.37	42.44	50.14
Luna-1	65.67 ± 0.18	75.46 ± 1.36	37.02 ± 0.12	49.06 ± 0.64	56.81
Luna-16	65.53 ± 0.08	75.93 ± 0.89	36.98 ± 0.30	50.36 ± 0.43	57.20
Luna-256	65.65 ± 0.35	79.44 ± 0.76	37.21 ± 0.22	50.90 ± 0.51	58.30
ConvLuna-1	82.10 ± 0.45	**81.76** ± 0.73	43.95 ± 1.75	56.66 ± 0.44	66.12
ConvLuna-16	**84.25** ± 0.16	80.47 ± 1.03	**44.14** ± 0.69	**56.93** ± 0.59	**66.45**
ConvLuna-256	83.29 ± 0.15	80.74 ± 1.08	43.56 ± 2.55	56.69 ± 0.64	65.90

Table 1: Vanilla Transformer and Luna compared with ConvLuna. We report accuracy mean and standard deviation across five training runs for each setup. Values for the vanilla Transformer are taken from (Tay et al., 2020), other results are by us. Numbers in names indicate memory size.

URM Statement

The authors acknowledge that at least one key author of this work meets the URM criteria of ICLR 2024 Tiny Papers Track.

Acknowledgments

Uladzislau Yorsh was supported by the grant №290223 of the Charles University Grant Agency and partially by SVV project number 260 698. Ondřej Bojar was supported by the grant 19-26934X (NEUREM3) of the Czech Science Foundation.

References

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. *CoRR*, abs/1807.03819, 2018. URL http://arxiv.org/abs/1807.03819.

Salvador García and Francisco Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research*, 9(89): 2677–2694, 2008. URL http://jmlr.org/papers/v9/garcia08a.html.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver: General perception with iterative attention. *CoRR*, abs/2103.03206, 2021. URL https://arxiv.org/abs/2103.03206.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, Toronto, Ontario, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Juho Lee, Yoonho Lee, et al. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/lee19d.html.

Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. Luna: Linear unified nested attention, 2021.

Markus N. Rabe and Charles Staats. Self-attention does not need o(n^2) memory. *CoRR*, abs/2112.05682, 2021. URL https://arxiv.org/abs/2112.05682.

Dragomir R. Radev, Pradeep Muthukrishnan, and Vahed Qazvinian. The ACL Anthology network corpus. In *Proceedings of the 2009 Workshop on Text and Citation Analysis for Scholarly Digital Libraries (NLPIR4DL)*, pp. 54–61, Suntec City, Singapore, August 2009. Association for Computational Linguistics. URL https://aclanthology.org/W09-3607.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019. URL http://arxiv.org/abs/1910.10683.

Sheng Shen, Zhen Dong, et al. Q-BERT: hessian based ultra low precision quantization of BERT. *CoRR*, abs/1909.05840, 2019. URL http://arxiv.org/abs/1909.05840.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *CoRR*, abs/2011.04006, 2020. URL https://arxiv.org/abs/2011.04006.

Ashish Vaswani, Noam Shazeer, et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Qingyang Wu, Zhenzhong Lan, et al. Memformer: A memory-augmented transformer for sequence modeling. In *Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022*, pp. 308–318. Association for Computational Linguistics, 2022.

A Statistical Comparison of Different Memory Sizes

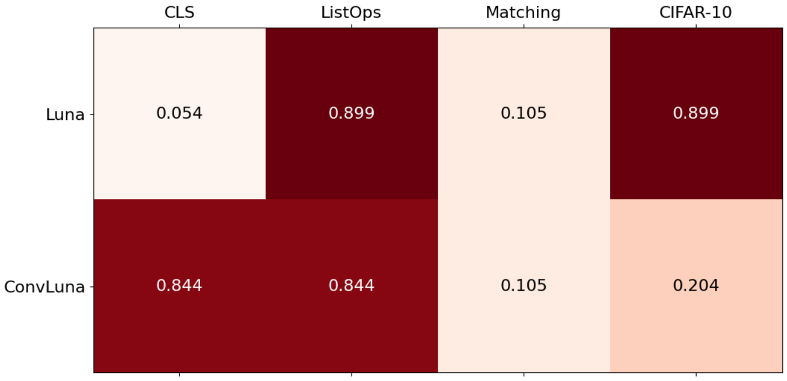


Figure 2: Achieved significances (p-values) of the Friedman test across all considered memory sizes with the H0 = "expected accuracies are equal". Color codes and numbers correspond to the p-values, corrected using the Holm metod (García & Herrera, 2008). We could not reject the null hypothesis on the 5% level of significance for any setup.

To find out whether the increasing memory actually provides statistically significant performance gains, we conduct the Friedman test (Figure 2) over accuracy score samples for memory sizes {1, 16, 256} for each combination of the model and the task. As mentioned before, each sample thus contains five experiments.

B Attention and Luna Definition

The attention (Vaswani et al., 2017) performs the following calculation over the three matrix inputs Q, K and V:

$$\begin{aligned} \text{MultiHeadAttention}(Q, K, V) &= [\text{head}_1, \text{head}_2, \ldots, \text{head}_h]W^O \\ \text{head}_h &= \text{Att}(QW_h^Q, KW_h^K, VW_h^V) \end{aligned}$$

where

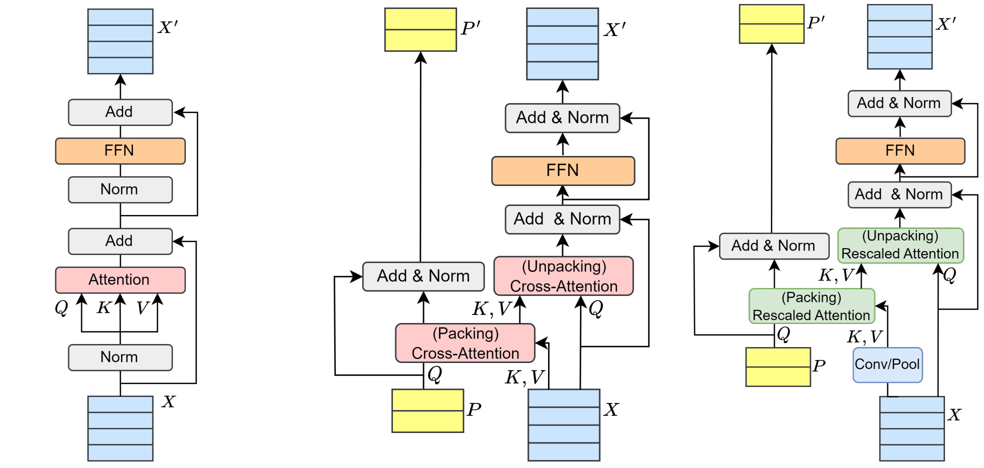
$$\text{Att}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_h}}\right)V$$

and where W_h is the corresponding weight matrix vertical slice for a particular attention head. For simplicity, most of Transformer implementations keep input sizes $Q \in \mathbf{R}^{L\times d}$, $K, V \in \mathbf{R}^{H\times d}$ and weight matrices $W_h^Q, W_h^K, W_h^V, W_h^O \in \mathbf{R}^{d\times d_h}$, $d_h = d/h$. The matrix $\text{softmax}(QK^T d_h^{-1/2}) \in \mathbf{R}^{L\times H}$ is often being referred to as the attention matrix, and may be interpreted as a matrix of relevance scores between Q and K vectors. The Transformer encoder module equation is thus (see Figure 3a):

$$\begin{aligned} X_{\text{norm}} &= \text{LayerNorm}(X) \\ I &= X + \text{MultiHeadAttention}(X_{\text{norm}}, X_{\text{norm}}, X_{\text{norm}}) \\ X' &= I + \text{FFN}(\text{LayerNorm}(I)) \end{aligned}$$

Since the computation of the $L \times H$ attention matrix may be expensive (e.g. quadratic when $Q = K = V$), Luna (Ma et al., 2021) factorizes the computation into two attentions (Figure 3b): one

To avoid the overloaded term "layer", by "module" we denote the whole encoder/decoder block, consisting of attentions, FFNs and skip connections, see Figure 3.



(a) Vanilla Transformer (b) Luna (c) ConvLuna

Figure 3: Comparison of the ordinary Transformer, Luna and the proposed ConvLuna encoder blocks (we refer to them as modules). “Packing” and “unpacking” attention layers in Luna are analogous to the attentions in the vanilla Transformer, while the Rescaled Attention additionally multiplies attention logits with a learnable parameter. The “unpacking” attention of ConvLuna is also denoted as rescalable; however, in our experiments we keep the normalization fixed and identical to the vanilla Transformer.

(“packing”) with Q given as a memory of a fixed length (we denote it as P in the figure) and K, V being module inputs (we denote them as X); and the second one (“unpacking”) with the Q given as the module input and K, V as packing attention outputs (Figure 3b):

$$\begin{aligned}
P_{\text{packed}} &= \text{MultiHeadAttention}(P, X, X) \\
X_{\text{unpacked}} &= \text{MultiHeadAttention}(X, P_{\text{packed}}, P_{\text{packed}}) \\
I &= \text{LayerNorm}(X + X_{\text{unpacked}}) \\
P', X' &= \text{LayerNorm}(P + P_{\text{packed}}), \text{LayerNorm}(\text{FFN}(I) + I)
\end{aligned}$$

The packing attention output serves as the memory input into the next module after applying a skip connection, while the unpacking attention output proceeds to summation with the module input and the residual FFN as in the original Transformer model. Since the P length is constant and typically significantly lower than the length of X, the computation of both the packing and unpacking attentions has linear complexity in H, which considerably accelerates the computation for longer inputs compared to vanilla Transformer.

C ConvLuna Description

The proposed model differs from the Luna in the packing attention implementation. We change the following:

- **Convolution/MaxPooling layers**, which we apply on keys and values to filter input signal.
- **Learnable softmax temperature**, which we use to rescale attention logits.

The main change we introduce into the model is applying either a convolution or pooling on the input in the “packing” attention. In particular, we apply it on the keys and values (which represent an input text), and not on the queries (which come from the memory); both operations only affect the length dimension, and there is no depthwise interaction in convolutions. The computation flow is similar to the Luna; however, we employ a slightly changed attention module for “packing” attention:

$$\begin{aligned}
\text{RescaledAttention}(Q, K, V) &= [\text{head}_1, \text{head}_2, \ldots, \text{head}_h]W^O \\
\text{head}_h &= \text{RescaledAtt}(QW_h^Q, KW_h^K, VW_h^V) \\
\text{RescaledAtt}(Q, K, V) &= \text{softmax}\left(\frac{Q\tilde{K}^T}{\exp(\tau)}\right)\tilde{V} \\
\tilde{K}, \tilde{V} &= \text{FilterOp}(K), \text{FilterOp}(V)
\end{aligned}$$

where FilterOp is either a convolution or max pooling. The hyperparameters of the FilterOp are kernel size K and stride S; given the input sequence of size $L \times d$ (where L is input length and d is the hidden dimension), we apply a kernel of size (K, 1) on it with a stride (S, 1). We leverage the computer vision interpretation of the operations as filtering the input signal, and leaving only a relevant fraction of it.

Another change we made in the model is the learnable softmax temperature. The original attention formulation (Appendix B) contains the $d^{-\frac{1}{2}}$ multiplier of the query-key product to avoid the softmax saturation; however, during preliminary experiments we have found memory-input product demonstrating less variance than input-input product in the vanilla Transformer. To alleviate this, we have replaced the $d^{-\frac{1}{2}}$ normalizer with $\exp(\tau)$, where τ is a learnable scalar which we initialize with zero. This does not increase the model flexibility per se, since the term may be absorbed either into W^Q or W^K matrix, but it allows to accelerate the training by increasing the variance of value vectors at the beginning, and imposes only a negligible computational overhead.

D Experimental Setup and Hyperparameters

We examine our models on the subset of the Long Range Arena (LRA; Tay et al., 2020) benchmark, which consists of four classification tasks:

- **Byte pair-encoded (BPE) text classification.** This task consists in binary sentiment classification of the IMDB dataset texts encoded as byte pairs. This creates input sequences up to $4k$ tokens long with relatively short subword units.
- **BPE text matching**. The dataset is the ACL Anthology Network (Radev et al., 2009), encoded in a way similar to the previous task with sequences up to $4k$ tokens long. The model needs to process two inputs and to use the concatenated hidden representations as an input to the final layer to classify, whether there is a citation link between the documents.
- **ListOps**. This task consists in processing nested arrays of digits, coupled with aggregation operations such as max, min, median and sum modulo, up to $2k$ tokens long. The model has to predict the correct answer out of ten, and this task tests the ability of the model to process hierarchical inputs. An example input looks like:
  `[MED 9 [MAX 4 [MIN 6 3 7 8 9 X 1 2 ...`
- **CIFAR-10 image classification**. This task is an image classification, but the input is represented as a sequence of pixel values. The data are CIFAR-10 dataset images (Krizhevsky, 2009), converted to 8-bit grayscale and flattened inputs 1024 tokens long.

The benchmark suggests training from scratch and limiting the additional parametrization, which allows for a fairer comparison with other models.

Table 2 lists the values of hyperparameters for our experiments. All the ConvLuna models share projection matrices between packing and unpacking attention, and share the W^K and $W^V = \mathbf{I}_d$ weights within a module to increase throughput. We have found the filtering (convolution or pooling) operations hyperparameters relatively shareable across tasks, with only ListOps benefiting from another configuration. We apply both operations length-wise, with the stride being the main hyperparameter to trade the throughput for performance. To better see how memory is being utilized by the model, we use averaged memory cells instead of the [CLS] token as an input embedding before the final layer.

Parameter	**Classif.**	**Matching**	**ListOps**	**CIFAR**
Seq. Length	4000	4000	2000	1024
Batch Size	32	32	32	64
Training Steps	25 000	30 000	50 000	30 000
Optimizer	AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$)			
Base LR	0.005	0.015	0.005	0.01
Weight Decay	0.01	0.04	0.01	0.01
Warmup Steps	8000	8000	1000	700
Schedule	Base LR * Warmup * Sqrt Decay			
Warmup Mul.	$min(1, \text{Current Step}/\text{Warmup Steps})$			
Sqrt Decay Mul.	$1/\sqrt{max(\text{CurrentStep, WarmupSteps})}$			
Loss	CCE			
Blocks	4	4	6	8
Heads	4			
Hidden dim.	128			
QKV dim.	128			
MLP dim.	512			
Filter Op.	MaxPool	MaxPool	Conv	MaxPool
Kernel Size	4	4	32	4
Stride	1			
Dropout	0.1	0.1	0.1	0.0
Activation	GELU			
Pooling	Memory Cells Average			
Pos. encoding	Learnable Absolute			

Table 2: Hyperparameters used for the LRA experiments.

Model	Classification ↑	Matching ↑	ListOps ↑	CIFAR-10 ↑	Average ↑
OnlyScaling-1	65.45 ± 0.30	72.46 ± 0.68	37.12 ± 0.47	49.23 ± 1.35	56.07
OnlyScaling-16	66.81 ± 0.21	74.99 ± 1.67	37.08 ± 0.18	48.67 ± 0.65	56.89
OnlyScaling-256	66.95 ± 0.20	75.55 ± 0.61	37.52 ± 0.33	48.90 ± 1.01	57.23
OnlyFiltering-1	**82.39** ± 0.62	82.38 ± 0.73	40.44 ± 2.31	**58.69** ± 1.13	**65.98**
OnlyFiltering-16	80.89 ± 0.44	**82.66** ± 1.46	**41.48** ± 1.74	58.06 ± 0.59	65.77
OnlyFiltering-256	80.27 ± 0.97	82.10 ± 1.68	41.24 ± 1.42	58.12 ± 0.66	65.43

Table 3: Results of the ablation experiments. OnlyScaling denotes the setup where we apply no filtering operations in packing attention, while OnlyFiltering means that we do not use the learnable softmax temperature there. Following the Table 1, we report accuracy score mean and standard deviation across five runs for each setup, and highlight the best result for each task with bold.

E Ablation Study

During the ablation experiments, we run the training of our models again, but now applying only one change at a time. The Table 3 reveals that the applying convolution or pooling indeed leads to significant performance improvements on the considered tasks. However, a straighforward improvement is not the case when we apply the learnable logit scaling; in some setups it can improve the accuracy on its own or in combination with the convolution/pooling, but in some setups (CIFAR-10) it degrades the performance.

Although the effect of the learnable logit scaling is not the same for all the considered tasks, the combination of both methods yields the best average accuracy across the all setups (Table 1). We also refer to the fact that the initial value of the scalar is lower than the fixed $d^{-\frac{1}{2}}$ of the vanilla attention, which in some cases may be the more important factor than the trainable parameter itself. Dividing the attention logits with a lower value leads to more saturated and varied attention scores at the training start compared to the nearly uniform distribution of the vanilla attention, which results in a better result and a faster convergence in some tasks.

We provide the following interpretation: for the setups such as ListOps, where the models demonstrate very slow convergence, or Classification, where we hypothesize that the model operates similarly to the Bag-of-Words and thus needs to focus more on particular tokens, such behavior that forces more attention to a smaller subset of tokens may be beneficial. At the same time, within the CIFAR-10 and Matching tasks the answer is unlikely to be dependent only on a small subset of input tokens, so a more focused attention at the beginning stops being a benefit and the additional parameter may contribute to the overfitting.