ABC: Attention with Bounded-Memory Control

Hao Peng[♠] **Jungo Kasai**[♠] **Nikolaos Pappas**[★*] **Dani Yogatama**[♣] **Zhaofeng Wu**[♢*]
Lingpeng Kong[♦] **Roy Schwartz**[♡] **Noah A. Smith**[♠♢]

[♠]Paul G. Allen School of Computer Science & Engineering, University of Washington
[★]Amazon Web Services [♣]DeepMind [♢]Allen Institute for Artificial Intelligence
[♡]School of Computer Science & Engineering, Hebrew University of Jerusalem
[♦]Department of Computer Science, The University of Hong Kong
{hapeng,jkasai,npappas,zfw7,nasmith}@cs.washington.edu
dyogatama@deepmind.com, lpk@cs.hku.hk
roy.schwartz1@mail.huji.ac.il

Abstract

Transformer architectures have achieved state-of-the-art results on a variety of natural language processing (NLP) tasks. However, their attention mechanism comes with a quadratic complexity in sequence lengths, making the computational overhead prohibitive, especially for long sequences. Attention context can be seen as a random-access memory with each token taking a slot. Under this perspective, the memory size grows linearly with the sequence length, and so does the overhead of reading from it. One way to improve the efficiency is to bound the memory size. We show that disparate approaches can be subsumed into one abstraction, **a**ttention with **b**ounded-memory **c**ontrol (ABC), and they vary in their *organization* of the memory. ABC reveals new, unexplored possibilities. First, it connects several efficient attention variants that would otherwise seem distinct. Second, this abstraction gives new insights—an established approach (Wang et al., 2020b) previously thought to *not* be applicable in causal attention, *actually is*. Last, we present a new instance of ABC, which draws inspiration from existing ABC approaches, but replaces their heuristic memory-organizing functions with a *learned*, contextualized one. Our experiments on language modeling, machine translation, and masked language model fine-tuning show that our approach outperforms previous efficient attention models; compared to strong transformer baselines, it significantly improves the inference time and space efficiency with no or negligible accuracy loss.

1 Introduction

Transformer architectures are now central in natural language processing (Vaswani et al., 2017). They rely on the attention mechanism (Bahdanau et al., 2015) to contextualize the input. The context can be seen as a random access **memory** whose size linearly grows with the sequence length; each query reads from it using a softmax-normalized linear combination, with overhead linear in the memory size. This amounts to a quadratic complexity overall, making transformers' computational overhead prohibitive, especially for long sequences.

One way to improve attention's efficiency is to bound its memory size. Imposing a constant-sized constraint over the memory ensures that reading from it has constant time and space overhead, yielding a linear overall complexity in sequence lengths. This is in fact a common strategy adopted by several recent works. In this work, we show that some of these works are closely connected in ways that, to date, have gone unremarked. We propose **a**ttention with **b**ounded-memory **c**ontrol (ABC), a unified abstraction over them. In ABC, constant-sized memories are organized with various control strategies, e.g., induced from heuristic patterns (Beltagy et al., 2020; Zaheer et al., 2020; Ainslie et al., 2020; Rae et al., 2020, *inter alia*), locality assumptions (Parmar et al., 2018; Liu et al., 2018), or positions (Wang et al., 2020b).

These strategies, by and large, are "context-agnostic." In response to this, we propose ABC_{MLP}, a particular instance of ABC that learns a contextualized control strategy from data. Specifically, ABC_{MLP} uses a neural network to determine how to store each token into the memory (if at all). Compared to previous bounded-memory models, it strikes a better trade-off between accuracy and efficiency: controlling for the accuracy, ABC_{MLP} can get away with much smaller memory sizes.

ABC models (including ABC_{MLP}) come with a linear complexity in sequence lengths, and admit recurrent computation graphs in causal attention (self-attention over the prefix). Therefore they are appealing choices in a variety of applications, including text encoding, language modeling and text generation. This leads to a surprising finding. Linformer (Wang et al., 2020b), an established efficient attention method, was previously thought *not*

*This work was done while Zhaofeng Wu and Nikolaos Pappas were at the University of Washington.

to be applicable in causal attention or autoregressive decoding (Tay et al., 2020). Through the ABC view, we show that it actually *is*, and achieves competitive performance in our machine translation experiments.

ABC connects existing models that would otherwise seem distinct, reveals new insights into established methods, and inspires new efficient attention architectures. We explore its applications in transformers, as a drop-in substitute for the canonical softmax attention. ABC offers a novel lens that can help future research in the analysis of transformers, where the theoretical insights are still catching up with empirical success. Experiments on language modeling, machine translation, and masked language model finetuning show that our ABC_{MLP} model outperforms previous ABC approaches in accuracy with a much smaller memory size. Compared to the strong transformer baseline, ABC_{MLP} achieves a significant speedup and memory savings at inference time, with no or negligible accuracy loss. The efficiency improvements are more prominent for long sequences, suggesting that the asymptotic savings are even more appealing in applications involving long sequences. We release our code at https://github.com/Noahs-ARK/ABC.

2 An Outer-Product View of Attention

This section presents our outer-product memory perspective of attention, which allows for a smooth transition to later discussion.

In attention, a sequence of **queries** $\{\mathbf{q}_i\}_{i=1}^N$ attend to a **memory** with N slots, each storing a **key** and **value** pair: $\mathbf{K} = [\mathbf{k}_1, \dots, \mathbf{k}_N]^\top, \mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_N]^\top \in \mathbb{R}^{N\times d}$.[1] Query $\mathbf{q}$ reads from the memory using a softmax-normalized linear combination, producing a d-dimensional vector:

$$\text{attn}(\mathbf{q}, \{\mathbf{k}_i\}, \{\mathbf{v}_i\}) = \mathbf{V}^\top \text{softmax}\left(\mathbf{K}\mathbf{q}\right). \quad (1)$$

This takes $\mathcal{O}(N)$ time and space. When the attention with N queries can be parallelized (e.g., in text encoding), it takes linear time and quadratic space; when it *cannot* be (e.g., in decoding), it takes quadratic time and linear space.

The memory can be equivalently represented as sums of vector outer products: $\mathbf{K} = \mathbf{I}\mathbf{K} = \sum_{i=1}^N \mathbf{e}_i \otimes \mathbf{k}_i$, $\mathbf{V} = \sum_{i=1}^N \mathbf{e}_i \otimes \mathbf{v}_i$. $\mathbf{I}$ is the identity matrix, and $\otimes$ denotes the outer product: $[\mathbf{x} \otimes \mathbf{y}]_{i,j} = x_i y_j$. N-dimensional vectors $\{\mathbf{e}_i\}$ form the standard basis: $\mathbf{e}_i$ has the ith element being one and others zeros. We can view $\mathbf{e}_i$ as **control vectors** that determine where to store $\mathbf{k}_i$ and $\mathbf{v}_i$:

$$\begin{aligned}\mathbf{e}_i \otimes \mathbf{k}_i &= \big[\underbrace{0, \dots 0}_{i-1}, 1, \underbrace{0, \dots, 0}_{N-i}\big]^\top \otimes \mathbf{k}_i \\ &= \big[\underbrace{\mathbf{0}}_{d\times(i-1)}; \mathbf{k}_i; \underbrace{\mathbf{0}}_{d\times(N-i)}\big]^\top.\end{aligned} \quad (2)$$

The N-by-d matrix on the last line has its ith row being $\mathbf{k}_i^\top$ and all others zeros; in this sense, $\mathbf{k}_i$ is stored in the ith slot by $\mathbf{e}_i$, not affecting others.

3 Attention with Bounded Memory

A straightforward way to improve attention's efficiency is to bound its memory size. Our outer-product view of attention provides a straightforward way to devise this, by replacing $\{\mathbf{e}_i\}$ with control vectors that select $n \ll N$ vectors to attend to. We dub this approach **a**ttention with **b**ounded-memory **c**ontrol (ABC). Concretely, let $\widetilde{\mathbf{K}}, \widetilde{\mathbf{V}} \in \mathbb{R}^{n\times d}$ denote a constant-size memory with n slots, with n set *a priori*.

$$\widetilde{\mathbf{K}} = \sum_{i=1}^N \boldsymbol{\phi}_i \otimes \mathbf{k}_i, \quad \widetilde{\mathbf{V}} = \sum_{i=1}^N \boldsymbol{\phi}_i \otimes \mathbf{v}_i. \quad (3)$$

$\{\boldsymbol{\phi}_i \in \mathbb{R}^n\}_{i=1}^N$ denotes a sequence of **control** vectors. The output is calculated by attending to $\widetilde{\mathbf{K}}$ and $\widetilde{\mathbf{V}}$: $\text{ABC}\left(\mathbf{q}, \{\mathbf{k}_i\}, \{\mathbf{v}_i\}, \{\boldsymbol{\phi}_i\}\right) =$

$$\widetilde{\mathbf{V}}^\top \text{softmax}\left(\widetilde{\mathbf{K}}\mathbf{q}\right). \quad (4)$$

We will discuss various ways to construct $\{\boldsymbol{\phi}_i\}$ in the subsequent sections. Reading from the memory takes a constant $\mathcal{O}(n)$ time and space; therefore ABC's overall complexity is $\mathcal{O}(Nn)$, linear in the sequence length.[2]

Eq. 3 offers an equivalent recurrent computation, which is particularly useful in causal attention where only the prefix is looked at,

$$\widetilde{\mathbf{K}}_{t+1} = \widetilde{\mathbf{K}}_t + \boldsymbol{\phi}_{t+1} \otimes \mathbf{k}_{t+1}, \quad (5)$$

likewise for $\widetilde{\mathbf{V}}_t$. $\widetilde{\mathbf{K}}_t$ and $\widetilde{\mathbf{V}}_t$ can be seen as the **recurrent hidden state** that encodes the prefix.

In what follows, we study several existing efficient attention approaches and show that they are in fact instances of the ABC abstraction.

[1] The number of queries and key-value pairs may differ, e.g., in the cross attention of a sequence-to-sequence model.

[2] Using bounded memory distinguishes ABC from softmax attention. If growing-size memory *were* allowed ($n = N$), an ABC with $\boldsymbol{\phi}_i = \mathbf{e}_i$ would fall back to softmax attention.

3.1 Linformer

Linformer (Wang et al., 2020b) is an established efficient transformer variant that has proven successful in masked language modeling and text encoding. It assumes fixed-length inputs and learns a low-rank approximation of the attention weights. A learned n-by-N matrix $\mathbf{W}^{\text{LF}}$ down projects the N-by-d dimensional keys and values along the *timestep* dimension, to an n-by-d memory: $\widetilde{\mathbf{K}}^{\text{LF}} = \mathbf{W}^{\text{LF}}\mathbf{K}$, $\widetilde{\mathbf{V}}^{\text{LF}} = \mathbf{W}^{\text{LF}}\mathbf{V}$; they are then used for attention computation with Eq. 4. This yields a linear complexity in the input length. Linformer is an ABC instance with $\boldsymbol{\phi}_i^{\text{LF}} = \mathbf{W}^{\text{LF}}_{:,i}$ (ith column), and in this sense, it learns a control vector for each position.

Previous works have noted that Linformer *cannot* be efficiently applied in causal attention (Table 1 of Tay et al., 2020). Indeed, it is less straightforward to avoid mixing future with the past when projecting along the timestep dimension. ABC reveals that, in fact, Linformer *is* applicable in causal attention. Like all ABC models, it admits a linear-complexity recurrent computation (Eq. 5): $\widetilde{\mathbf{K}}^{\text{LF}}_{t+1} = \widetilde{\mathbf{K}}_t + \boldsymbol{\phi}^{\text{LF}}_{t+1} \otimes \mathbf{k}_{t+1}$. This confirms ABC's benefits: it reveals new insights about existing models and reassesses their applications and impact. Our experiments show that Linformer achieves competitive performance in machine translation.

3.2 Clustering-Based Attention

Improving attention's efficiency with clustering has received an increasing amount of interest (Kitaev et al., 2020; Roy et al., 2020; Wang et al., 2020a, *inter alia*). ABC bears interesting connections to clustering-based methods. Here we discuss an approach that closely follows Vyas et al. (2020), except that it clusters keys and values instead of queries, and only attends to the centroids to reduce the effective context size. Formally, keys and values are grouped into $n < N$ clusters $\{\widetilde{\mathbf{k}}^{\text{CL}}_j\}_{j=1}^n$, $\{\widetilde{\mathbf{v}}^{\text{CL}}_j\}_{j=1}^n$.[3] Let an N-by-n binary matrix $\mathbf{M}$ denote the cluster membership shared between keys and values. $M_{i,j} = 1$ iff. $\mathbf{k}_i$ is assigned to cluster $\widetilde{\mathbf{k}}^{\text{CL}}_j$ and $\mathbf{v}_i$ to $\widetilde{\mathbf{v}}^{\text{CL}}_j$. The jth centroid for the keys is

$$\widetilde{\mathbf{k}}^{\text{CL}}_j = \sum_{i=1}^{N} \frac{M_{i,j}}{\sum_{\ell=1}^{N} M_{\ell,j}} \mathbf{k}_i; \tag{6}$$

[3] We use $\widetilde{\mathbf{k}}^{\text{CL}}_j$ to denote both the jth cluster and its centroid.

likewise for the values. It then attends over the centroids using Eq. 4, with $\widetilde{\mathbf{K}}^{\text{CL}} = [\widetilde{\mathbf{k}}^{\text{CL}}_1, \dots, \widetilde{\mathbf{k}}^{\text{CL}}_n]^\top =$

$$\sum_{j=1}^{n} \mathbf{e}_j \otimes \widetilde{\mathbf{k}}^{\text{CL}}_j = \sum_{j=1}^{n} \mathbf{e}_j \otimes \sum_{i=1}^{N} \frac{M_{i,j}}{\sum_{\ell=1}^{N} M_{\ell,j}} \mathbf{k}_i$$
$$= \sum_{i=1}^{N} \left(\sum_{j=1}^{n} \mathbf{e}_j \frac{M_{i,j}}{\sum_{\ell=1}^{N} M_{\ell,j}} \right) \otimes \mathbf{k}_i.$$

The last line indicates that this model is an instance of ABC: $\boldsymbol{\phi}_i = \sum_{j=1}^{n} (M_{i,j} / \sum_{\ell=1}^{N} M_{\ell,j}) \mathbf{e}_j$. The stack of centroids can be seen as the constant-size memory. Putting aside the clustering overhead (i.e., constructing $\mathbf{M}$ and computing centroids), it has a linear complexity in the sequence length.

3.3 Sliding-Window Attention

In some applications, being able to remove entries from the memory can be beneficial: clearing up older context frees slots for more recent ones, promoting a locality inductive bias. ABC offers the capability to do so, if augmented with an additional matrix multiplication. We use the sliding-window attention as an example.

Attending to the most recent n input tokens (Beltagy et al., 2020; Zaheer et al., 2020; Sukhbaatar et al., 2021, *inter alia*) can be seen as a first-in-first-out queue that "pops" out the oldest token while "pushing" in the most recent one: $\widetilde{\mathbf{K}}^{\text{WD}}_t = [\mathbf{k}_{t-n+1}, ..., \mathbf{k}_t]^\top$. The pop operation can be achieved by multiplying an n-by-n **upper shift matrix**: $U_{i,j} = \delta_{i+1,j}$, with δ being the Kronecker delta (i.e., $\mathbf{U}$ has ones only on the superdiagonal and zeros elsewhere). Left-multiplying $\mathbf{U}$ against $\widetilde{\mathbf{K}}^{\text{WD}}_t$ shifts its rows one position up, with zeros appearing in the last:

$$\mathbf{U}\widetilde{\mathbf{K}}^{\text{WD}}_t = \mathbf{U}\big[\underbrace{\mathbf{k}_{t-n+1}, \dots, \mathbf{k}_t}_{n}\big]^\top$$
$$= \big[\underbrace{\mathbf{k}_{t-n+2}, \dots, \mathbf{k}_{t-1}, \mathbf{k}_t}_{n-1}, \mathbf{0}\big]^\top \in \mathbb{R}^{n \times d}.$$

Then the most recent token can be put into the slot freed up: $\widetilde{\mathbf{K}}^{\text{WD}}_{t+1} = \mathbf{U}\widetilde{\mathbf{K}}^{\text{WD}}_t + \mathbf{e}_n \otimes \mathbf{k}_{t+1}$. $\mathbf{U}$ and $\boldsymbol{\phi}_t = \mathbf{e}_n$ ensure a first-in-first-out queue. Dilated and stride convolution patterns (Beltagy et al., 2020) can be similarly recovered (§A.4).

Recurrently multiplying $\mathbf{U}$ simulates the discrete pop operation (Grefenstette et al., 2015; Joulin and Mikolov, 2015; Yogatama et al., 2018) in a differentiable way. This is reminiscent of recurrent neural networks, while in this case $\mathbf{U}$ is *never* updated as

parameters. It is exciting to explore learning $\mathbf{U}$, but is beyond the scope of this work.

Discussion. Besides the models discussed above, certain variants of Rae et al. (2020) and sparse attention patterns (local-to-global attention; Beltagy et al., 2020; Zaheer et al., 2020; Ainslie et al., 2020) can also be seen as instances of ABC (§A). ABC provides a unified perspective of them, and at the same time points out their limitations: their control strategies are context-agnostic. In response to this, in §4 we propose to learn a contextualized strategy from data. Table 1 analyzes various ABC models, and Table 2 details their complexity.

4 Learned Memory Control

The ABC abstraction connects several existing approaches that would otherwise seem distinct. This inspires the design of new architectures. We hypothesize that learning a contextualized strategy can achieve better performance. This section introduces ABC_{MLP}. It parameterizes $\boldsymbol{\phi}$ with a single-layer multi-layer perceptron (MLP) that takes as input the token's representation $\mathbf{x}_i$, and determines which slots to write it into and how much.

$$\boldsymbol{\alpha}_i = \mathbf{exp}\left(\mathbf{W}_{\boldsymbol{\phi}}\mathbf{x}_i\right), \quad \boldsymbol{\phi}_i = \boldsymbol{\alpha}_i \Big/ \sum_{j=1}^{N} \boldsymbol{\alpha}_j. \quad (7)$$

Matrix $\mathbf{W}_{\boldsymbol{\phi}}$ is learned. $\mathbf{exp}$ is an elementwise activation function. The motivation is to allow for storing a "fractional" (but *never* negative) amount of input into the memory.[4] Using a non-negative activation, however, has a drawback: the scales of $\sum_i \boldsymbol{\phi}_i \otimes \mathbf{k}_i$ and $\sum_i \boldsymbol{\phi}_i \otimes \mathbf{v}_i$ would grow with the sequence lengths, making training less stable. To overcome this, we divide $\boldsymbol{\alpha}_i$ vectors by their sum. This functions as normalization and aims to offset the impact of varying sequence lengths.[5] It admits the recurrent computation graph as in Eq. 5, and has a linear complexity in the sequence length.

A key design choice of ABC_{MLP} is that its $\boldsymbol{\phi}_i$ depends *only* on current input $\mathbf{x}_i$. This helps (1) keep the recurrent computation efficient in practice (Lei et al., 2018), and (2) make it applicable in not only encoder self-attention and cross attention, but also causal attention. Concurrently to this work, Goyal et al. (2021) and Ma et al. (2021) also proposed methods to learn contextualized control. They compute $\boldsymbol{\phi}_i$ from *previous* layer's memory, revealing the full sequence to the control vectors. As a result, these two approaches are *unsuitable* for causal attention.[6]

ABC_{MLP}, as other ABC models, can be used as a drop-in replacement for the canonical softmax attention, and we apply its multihead variant in transformers. With proper parameter sharing, the number of additional parameters ABC_{MLP} incurs is small: inspired by Wang et al. (2020b), we tie $\boldsymbol{\phi}$-MLP's parameters across different layers, which adds less than 1% parameters to the models.

ABC_{MLP}: context-agnostic then context-dependent attention. We now dissect ABC_{MLP} and show that it can be seen as a cascade of two attention mechanisms: one with a learned context-agnostic "pseudo query" followed by one with a context-dependent query. Our analysis starts with a one-dimensional example; the conclusion generalizes to higher-dimensional cases.

Example 1. Consider ABC_{MLP} with a *single* memory slot ($n = 1$). It is parameterized with a learned vector $\mathbf{w}_{\boldsymbol{\phi}}$, and $\phi_i = \exp(\mathbf{w}_{\boldsymbol{\phi}} \cdot \mathbf{x}_i) / \sum_{j=1}^{N} \exp(\mathbf{w}_{\boldsymbol{\phi}} \cdot \mathbf{x}_j)$. Since ϕ_i is a scalar here, $\phi_i \otimes \mathbf{k}_i = \phi_i \mathbf{k}_i^\top$.

$$\begin{aligned}\widetilde{\mathbf{K}}^\top &= \sum_{i=1}^{N} \left(\phi_i \otimes \mathbf{k}_i\right)^\top \\ &= \sum_{i=1}^{N} \frac{\exp(\mathbf{w}_{\boldsymbol{\phi}} \cdot \mathbf{x}_i)}{\sum_{j=1}^{N} \exp(\mathbf{w}_{\boldsymbol{\phi}} \cdot \mathbf{x}_j)} \mathbf{k}_i \\ &= \mathrm{attn}\left(\mathbf{w}_{\boldsymbol{\phi}}, \{\mathbf{x}_i\}_{i=1}^N, \{\mathbf{k}_i\}_{i=1}^N\right).\end{aligned}$$

In other words, $\widetilde{\mathbf{K}}$ uses $\mathbf{w}_{\boldsymbol{\phi}}$ as a "pseudo-query" to attend to $\{\mathbf{x}_i\}$ and $\{\mathbf{k}_i\}$. Likewise, $\widetilde{\mathbf{V}}^\top = \mathrm{attn}(\mathbf{w}_{\boldsymbol{\phi}}, \{\mathbf{x}_i\}_{i=1}^N, \{\mathbf{v}_i\}_{i=1}^N)$. Despite its similarity to the standard softmax attention, Example 1 has a more efficient linear complexity in sequence lengths. $\mathbf{w}_{\boldsymbol{\phi}}$'s being context-independent is the key to the savings. Table 2 details its complexity.

Example 1's conclusion generalizes to higher-dimensional cases: the jth dimension of $\{\boldsymbol{\phi}_i\}$ attends to $\{\mathbf{x}_i\}$ and $\{\mathbf{k}_i\}$ using the jth row of $\mathbf{W}_{\boldsymbol{\phi}}$ as the context-independent pseudo-query; n such attention mechanisms run in parallel, stacking the

[4] We experiment with other activations in §C.2.

[5] Here encoder self-attention or cross attention is assumed, and the normalization sums over the entire sequence. Causal attention is slightly different, normalizing by the sum over the prefix instead: $\boldsymbol{\phi}_i = \boldsymbol{\alpha}_i / \sum_{j=1}^{i} \boldsymbol{\alpha}_j$. This does *not* require access to future tokens. §B.1 details a linear complexity computation graph of causal $\boldsymbol{\phi}_i$.

[6] Both are instances of ABC (§A.5). Ma et al. (2021) resorts to a variant of Katharopoulos et al. (2020) for causal attention.

Model	Section	$\boldsymbol{\phi}_t$	Mem. Control
Sliding-window	§3.3	$\mathbf{e}_n$	$\widetilde{\mathbf{K}}_{t+1} = \mathbf{U}\widetilde{\mathbf{K}}_t + \boldsymbol{\phi}_{t+1} \otimes \mathbf{k}_{t+1}$
Linformer	§3.1	$\mathbf{W}^{\text{LF}}_{:,t}$	$\widetilde{\mathbf{K}}_{t+1} = \widetilde{\mathbf{K}}_t + \boldsymbol{\phi}_{t+1} \otimes \mathbf{k}_{t+1}$
L2G Pattern	§A.1	$\mathbf{e}_i$ if $\mathbf{x}_t$ is the ith global token	
ABC$_{\text{RD}}$	§A.2	$\mathbf{e}_{i_t}$, where $i_t \sim \text{unif}\{1, n\}$	
Comp. Trans.	§A.3	$\mathbf{e}_{\lfloor nt/N \rfloor}$	
Clustering	§3.2	$\sum_{j=1}^{n} \left(M_{t,j} / \sum_{\ell=1}^{N} M_{\ell,j} \right) \mathbf{e}_j$	
ABC$_{\text{MLP}}$	§4	$\mathbf{exp}\left(\mathbf{W}_{\boldsymbol{\phi}}\mathbf{x}_t\right) / \sum_{i=1}^{t} \mathbf{exp}\left(\mathbf{W}_{\boldsymbol{\phi}}\mathbf{x}_t\right)$	

Table 1: A comparison of different ABC models. N denotes the sequence length, and n the memory size. $\boldsymbol{\phi}_t$ denotes the memory control vector for $\mathbf{k}_t$ and $\mathbf{v}_t$, and unif is the discrete uniform distribution.

	Time Complexity			Space Complexity		
Model	**Mem.**	**Per Query**	**Overall**	**Mem.**	**Per Query**	**Overall**
Softmax Attention	-	$\mathcal{O}(N)$	$\mathcal{O}(N^2)$	-	$\mathcal{O}(N)$	$\mathcal{O}(N^2)$
ABC	$\mathcal{O}(N)$	$\mathcal{O}(n)$	$\mathcal{O}(nN)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(nN)$

Table 2: ABC's time and space complexity in sequence length against the softmax attention's. "Mem." indicates the time and space needed for calculating and storing memory $\widetilde{\mathbf{K}}, \widetilde{\mathbf{V}}$. N denotes the sequence length, and n the memory size. The time complexity analysis assumes that the softmax attention *cannot* be parallelized across the queries. In practice, this is common in autoregressive decoding or for long sequences where the accelerators (e.g., GPUs) do not have enough threads to fully parallelize softmax attention's computation across different queries.

results into n-by-d memory $\widetilde{\mathbf{K}}$ and $\widetilde{\mathbf{V}}$. Intuitively, it is the "real queries" $\{\mathbf{q}_i\}$ that encode "what information is useful for the prediction task." Without access to them, ABC$_{\text{MLP}}$ summarizes the input for n times using different pseudo-queries, aiming to preserve enough information in the memory for onward computation. The attention output is calculated with the context-dependent real queries using Eq. 4. §B.2 presents a detailed derivation.

Connections to other prior works. Although starting from distinct motivations, ABC$_{\text{MLP}}$ closely relates to hierarchical attention (HA; Yang et al., 2016). HA summarizes the context into higher-level representations with a cascade of attention mechanisms, e.g., words to sentences, and then to documents. ABC$_{\text{MLP}}$ applies two types of attention. The first learns context-agnostic pseudo-queries and attends to the same sequence for n times in parallel, while the second retrieves from the memory with real queries. HA, in contrast, summarizes non-overlapping segments at each level.

The learned pseudo-queries closely relate to the inducing point method in set attention (ISA; Lee et al., 2019). ISA applies a non-linear feedforward network between a cascade of two attention modules. This precludes the outer-product memory computation and efficient recurrences in ABC.

Another line of work "linearizes" attention through kernel tricks and also applies bounded memory: their feature map dimensions are analogous to memory sizes. They substitute the softmax with approximations (Peng et al., 2021; Choromanski et al., 2021), heuristically designed (Katharopoulos et al., 2020; Schlag et al., 2021), or learned (Kasai et al., 2021b) functions. ABC$_{\text{MLP}}$ keeps the softmax, but over a smaller constant-sized context. This can be useful in practice: (1) ABC provides a unified perspective of several efficient attention methods, allowing for borrowing from existing wisdom to design new architectures; (2) it draws a close analogy to the canonical softmax attention, and is better-suited as its drop-in substitute in various application settings, as we will show in the experiments; (3) empirically, we find that ABC$_{\text{MLP}}$ can get away with a much smaller memory size to retain the accuracy. Peng et al. (2021) and Schlag et al. (2021) use gating to promote recency bias. The same technique is equally applicable in ABC models.

The learned contextualized memory control is reminiscent of the content-based addressing in neu-

ral Turing machines (NTM; Graves et al., 2014). ABC$_{\text{MLP}}$ computes the control vectors $\{\phi_i\}$ as a function of the input, but *not* of the memory as in NTM. This ensures that the control vectors at different timesteps can be computed in parallel, improving the time efficiency in practice (Lei et al., 2018; Peng et al., 2018). Analogies between memory and neural architectures are also made by other previous works (Hochreiter and Schmidhuber, 1997; Weston et al., 2015; Le et al., 2020, *inter alia*).

5 Experiments

We evaluate ABC models on language modeling (§5.1), sentence-level and document-level machine translation (§5.2), and masked language model fine-tuning (§5.3). Dataset statistics and implementation details are summarized in §C.

5.1 Language Modeling

Setting. We experiment with WikiText-103, sampled text from English Wikipedia (Merity et al., 2017). The BASE model with standard softmax attention is the strong transformer-based language model by Baevski and Auli (2019). We compare the following ABC variants, which build on BASE, but replace the softmax attention with linear-complexity bounded-memory attention alternatives while keeping other components the same.

- ABC$_{\text{MLP}}$, as described in §4, learns a contextualized exp-MLP as the ϕ function.
- Linformer (§3.1; Wang et al., 2020b).
- ABC$_{\text{RD}}$ stores each token in a randomly-selected memory slot with $\phi_t = \mathbf{e}_{i_t}$. i_t is uniformly drawn from $\{1, \ldots, n\}$ at each time step. This helps us quantify the differences between random and learned bounded-memory controls.

We consider two model size settings:

- 16 layers (Baevski and Auli, 2019). All models have around ∼242M parameters. They train with 512-token segments, and evaluate with 0 or 480 context sizes: a 0- or 480- length prefix precedes each evaluation segment.
- 32 layers (Kasai et al., 2021b). All models have ∼484M parameters. This setting applies layer dropout (Fan et al., 2020), and evaluates with a 256 context size. It aims to compare ABC$_{\text{MLP}}$ to several kernel-based efficient attention variants: ELU (Katharopoulos et al., 2020), RFA (Peng et al., 2021), and T2R (Kasai et al., 2021b).

Results. Table 3a compares ABC variants using Baevski and Auli (2019)'s 16-layer setting. Among ABC models, ABC$_{\text{MLP}}$ achieves the best performance for both context sizes. With a memory size $n = 64$, ABC$_{\text{MLP}}$ outperforms both Linformer and ABC$_{\text{RD}}$ by more than 2.9 test perplexity; and the gap is larger with the longer 480-length context: more than 3.6 test perplexity. ABC$_{\text{MLP}}$-32 outperforms its larger-memory ABC counterparts by more than 2.1 test perplexity. These results confirm ABC$_{\text{MLP}}$'s advantages of using a contextualized strategy. Surprisingly, Linformer *underperforms* ABC$_{\text{RD}}$, and its performance drops with the larger 480-length context window. This suggests that, while successful in text encoding, Linformer's position-based strategy is a suboptimal design choice for causal attention, at least for long context. All ABC models *underperform* the BASE, with ABC$_{\text{MLP}}$-64 having the smallest gap of 0.5 perplexity. ABC$_{\text{MLP}}$-32 outperforms kernel-based methods by more than 0.9 test perplexity, using Kasai et al. (2021b)'s 32-layer setting (Table 3b).

Model	n	Dev. 0	Dev. 480	Test 0	Test 480
BASE	-	19.8	18.4	20.5	19.0
Linformer	64	26.5	27.1	27.2	30.7
ABC$_{\text{RD}}$	64	23.2	22.3	24.0	23.1
ABC$_{\text{MLP}}$	32	21.2	19.7	21.9	20.5
ABC$_{\text{MLP}}$	64	**20.4**	**18.9**	**21.1**	**19.5**

(a) 16-layer setting. 0/480 indicate evaluation context sizes.

Model	n	Dev.	Test
†BASE	-	17.9	18.5
†ELU	128	22.0	22.8
†RFA	32	20.4	21.3
†T2R	32	20.1	20.8
ABC$_{\text{MLP}}$	32	**19.2**	**19.9**

(b) 32-layer setting. A 256-length context is used at evaluation time. † numbers are due to Kasai et al. (2021b).

Table 3: WikiText-103 language modeling perplexity (**lower is better**). n denotes the memory size. Bold numbers perform the best among linear-complexity models.

5.2 Machine Translation

Datasets. To assess their performance over various output lengths, we compare ABC models on sentence- and document- level machine translation.

- Sentence-level translation with WMT14 EN-DE

Model	Cross n	Causal n	BLEU
BASE	-	-	27.2
ABC$_{\text{RD}}$	32	32	25.7
ABC$_{\text{RD}}$	64	64	26.2
Linformer	32	32	26.6
Linformer	64	64	26.7
ABC$_{\text{MLP}}$	32	8	27.1
ABC$_{\text{MLP}}$	32	32	**27.3**

(a) Bolded number outperforms BASE.

Model	Cross n	Causal n	BLEU
BASE	-	-	39.9
Linformer	128	64	-
ABC$_{\text{RD}}$	128	64	38.6
ABC$_{\text{MLP}}$	128	64	**39.7**

(b) Linformer fails to converge even with multiple random seeds. Bold number performs the best among ABC models.

Table 4: Machine translation test SacreBLEU. Left: sentence-level translation with WMT14 EN-DE; right: document-level translation with IWSLT14 ES-EN.

(Bojar et al., 2014). The preprocessing and data splits follow Vaswani et al. (2017).

- Document-level translation with IWSLT14 ES-EN (Cettolo et al., 2014). We use Miculicich et al. (2018)'s data splits and preprocessing. Following standard practice (Voita et al., 2019), a 4-sentence sliding window is used to create the dataset, i.e., each instance has 4 sentences.

Setting. We compare ABC variants as in §5.1. §C.2 further compares to the clustering-based (§3.2) and sliding-window (§3.3) ABC variants.

The BASE model they build on is our implementation of transformer-base (Vaswani et al., 2017). ABC variants replace decoder cross attention and causal attention with bounded-memory attention, while keeping softmax attention for the encoder, since its overhead is much less significant (Kasai et al., 2021a); other components are kept the same. §C.2 studies a model that replaces *all* softmax attention with ABC$_{\text{MLP}}$. It performs on par with BASE, confirming ABC$_{\text{MLP}}$'s broad applicability in various application scenarios. We evaluate with SacreBLEU (Post, 2018).

Results. Table 4a summarizes sentence-level machine translation results on the WMT14 EN-DE test set. Overall ABC$_{\text{MLP}}$ performs on par with BASE, with either 32-32 cross-causal memory sizes or 32-8. Even with smaller memory sizes, it outperforms other ABC variants by more than 1.1 BLEU. Differently from the trend in the language modeling experiment (§5.1), Linformer outperforms ABC$_{\text{RD}}$ by more than 0.5 BLEU. We attribute this to the smaller sequence lengths of this dataset. ABC$_{\text{MLP}}$ outperforms other ABC models by more than 0.4 BLEU, even with smaller memory sizes.

The trend is similar on document-level translation with IWSLT14 ES-EN (Table 4b), except that ABC$_{\text{MLP}}$ slightly *underperforms* BASE by 0.2 BLEU. This suggests that even with longer sequences, ABC$_{\text{MLP}}$ is effective despite its bounded memory size. Linformer fails to converge even with multiple random seeds, suggesting the limitations of its purely position-based strategy in tasks involving decoding varying-length text.

5.3 Masked Language Model Finetuning

Setting. We compare the ABC variants as in §5.1. It is interesting to pretrain ABC from scratch, but we lack the resources to do so. Instead, we warm-start from a pretrained RoBERTa-base (Liu et al., 2019) trained with the softmax transformer, swap its attention with ABC variants, and continue pretraining with the masked language modeling (MLM) objective on a concatenation of BookCorpus (Zhu et al., 2015), English Wikipedia, OpenWebText (Gokaslan and Cohen, 2019), and RealNews (Zellers et al., 2019).[7] Then the models are finetuned and evaluated on downstream classification datasets from the the GLUE benchbark (Wang et al., 2019). This is an appealing setting, since it avoids reinvesting the huge amounts of resources already put into pretraining.[8]

Results. Table 5 compares downstream text classification performance. BASE indicates a baseline that continues pretraining RoBERTa-base on our data.[9] Following standard practice, we report development accuracy. Linformer achieves competitive

[7] Our data differs from RoBERTa's, which we do *not* have access to. We replace CC-News (Nagel, 2016) with RealNews, and drop Stories (Trinh and Le, 2018), whose public access is broken at the time of this work.

[8] In preliminary experiments, we explored swapping in ABC, and then directly finetuning on downstream tasks *without* continued MLM pretraining; all models fail.

[9] BASE slightly *underperforms* RoBERTa-base. This could be due to overfitting, or the pretraining data discrepancy.

Model	n	MNLI	QNLI	QQP	SST	Avg.
BASE	-	87.2	92.4	91.7	94.3	91.4
Linformer	64	85.3	91.8	90.8	92.4	90.1
Linformer	128	86.1	91.9	91.4	93.7	90.8
ABC$_{\text{MLP}}$	64	85.6	91.8	91.7	93.8	90.7
ABC$_{\text{MLP}}$	128	**87.1**	**92.6**	**91.8**	**94.4**	**91.5**

Table 5: Text classification development set accuracy. All models continue pretraining RoBERTa-base on our data with the MLM objective. Bold numbers perform the best among ABC models, and underlined ones perform on par with or better than BASE.

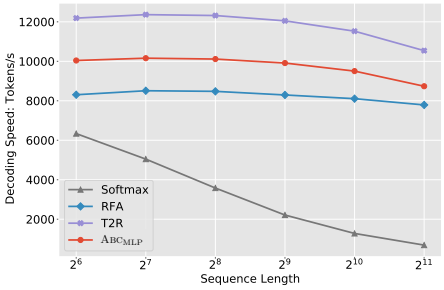
performance, aligned with Wang et al. (2020b)'s results. ABC$_{\text{MLP}}$ outperforms Linformer, and performs on par with or better than BASE, affirming the benefits of using contextualized memory organization in MLM. ABC$_{\text{RD}}$ fails to converge in continued pretraining even with multiple seeds.

Based on the above results, we think ABC$_{\text{MLP}}$ can achieve competitive performance when pretrained from scratch, just as Linformer does (Wang et al., 2020b). Further empirical exploration is beyond our budget and left for future work.

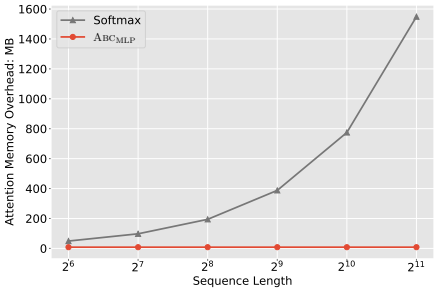
6 Analysis

Decoding efficiency over varying sequence lengths. ABC's efficiency gains can be more prominent for long sequences. We study ABC$_{\text{MLP}}$'s decoding overhead with varying sequence lengths. Following Kasai et al. (2021b), we consider a sequence-to-sequence generation experiment. Three linear-complexity models are compared: RFA (with 256/128 cross/causal memory sizes; Peng et al., 2021), T2R (32/4; Kasai et al., 2021b), and ABC$_{\text{MLP}}$ (32/8). The sizes are chosen to maximize efficiency *without* accuracy drop. T2R needs to be finetuned from a pretrained transformer to match its performance, while others don't.

All linear-time models achieve consistent decoding speed for different lengths (Figure 1a), substantially outpacing the softmax attention baseline, especially for long sequences. In particular, ABC$_{\text{MLP}}$ decodes ∼1.25 times faster than RFA, another competitive model that can match transformer's accuracy *without* a warm start from a pretrained model. This can be attributed to the fact that ABC$_{\text{MLP}}$ achieves similar accuracy with a much smaller memory. T2R's memory sizes are similar to ABC$_{\text{MLP}}$'s, but it decodes about 20% faster.



(a) Decoding Speed.



(b) Decoding memory overhead.

Figure 1: Sequence-to-sequence decoding speed (top) and memory consumption (bottom) varying sequence lengths. Greedy decoding is used, with batch size 16.

This is because it does *not* compute the softmax when calculating attention output, while ABC$_{\text{MLP}}$ does (Eq. 4). These results show that ABC$_{\text{MLP}}$ is an appealing modeling choice for decoding tasks, especially when training from scratch is desired.

ABC$_{\text{MLP}}$ also achieves significant savings in terms of memory overhead (Figure 1b). ABC$_{\text{MLP}}$, RFA, and T2R's curves are similar.

Text encoding efficiency. We compare the efficiency of ABC$_{\text{MLP}}$ against softmax attention and Linformer when used as text encoders. The models' sizes mirror those in the MLM experiment (§5.3). Table 6 summarizes inference time and memory overhead with 512-length inputs, batch size 16. Both ABC$_{\text{MLP}}$ and Linformer achieve inference speed gains and memory savings over BASE. Linformer is faster, since its linear projection is cheaper to compute than ABC$_{\text{MLP}}$'s MLP. Inference speed is measured on the same V100 GPU. The trend in memory overhead is similar.

Although ABC$_{\text{MLP}}$ slightly underperforms Linformer in terms of inference speed, it can be a more appealing architectural choice in practice: in all of our 5 experiments, ABC$_{\text{MLP}}$ outperforms other ABC models in accuracy. Linformer, in contrast, fails to converge or yields sub-optimal performance on some tasks. This confirms its flexibility and ap-

	BASE	Linformer		ABC$_{\text{MLP}}$	
n	-	64	128	64	128
Speed	1.0×	1.7×	1.5×	1.5×	1.3×
Memory	1.0×	0.5×	0.6×	0.5×	0.6×

Table 6: Text encoding inference speed (higher is better) and memory (lower is better). Inputs are text segments with 512 tokens and batch size 16.

		Cross n			
		8	16	32	64
Causal n	8	24.7	25.2	25.6	25.5
	16	-	25.4	25.7	25.6
	32	-	-	25.7	25.8
	64	-	-	-	25.8

Table 7: ABC$_{\text{MLP}}$'s SacreBLEU on WMT14 EN-DE development data varying memory sizes.

plicability in various settings.

Memory size's impact on accuracy. Practically, one may want to minimize the memory size to improve efficiency. We use the WMT14 EN-DE experiment to investigate how memory size affects accuracy. Using the §5.2's setup, we vary ABC$_{\text{MLP}}$'s cross and causal attention memory sizes and compare their translation quality on the development data. They are selected from $\{8, 16, 32, 64\}$, with cross attention's equal to or larger than causal's: cross attention is more important than causal attention in machine translation (Michel et al., 2019). Our results (Table 7) align with this observation: when cross attention memory is large enough, reducing causal attention memory size from 64 to 8 has a minor 0.3 BLEU drop. Surprisingly, ABC$_{\text{MLP}}$ with 8-8 sized cross-causal memory is only 1.1 BLEU behind the best-performing configuration.

7 Conclusion

We presented attention with bounded-memory control (ABC). It provides a unified perspective of several recently-proposed models, and shows that they vary in the organization of the bounded memory. ABC reveals new insights into established methods and inspires new architectures. We proposed ABC$_{\text{MLP}}$, a particular instance of ABC that learns a contextualized memory control. On language modeling, machine translation, and masked language model finetuning, ABC$_{\text{MLP}}$ outperforms previous ABC models. Compared to the strong transformer baseline, ABC$_{\text{MLP}}$ achieves substantial efficiency improvements with no or negligible accuracy loss.

Acknowledgments

We would like to thank the ARK group at the University of Washington for their helpful feedback, and the anonymous reviewers for their thoughtful comments. This work was supported in part by NSF grant 2113530 and a Google Fellowship. Nikolaos Pappas was supported by the Swiss National Science Foundation grant P400P2_183911.

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 *Proc. of EMNLP*.

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *Proc. of ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

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

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proc. of WMT*.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT evaluation campaign. In *Proc. of IWSLT*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Łukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2021. Rethinking attention with performers. In *Proc. of ICLR*.

Kornél Csernai. 2017, accessed September 1, 2020. *First Quora Dataset Release: Question Pairs*.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *Proc. of ICLR*.

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

Anirudh Goyal, Aniket Didolkar, Alex Lamb, Kartikeya Badola, Nan Rosemary Ke, Nasim Rahaman, Jonathan Binas, Charles Blundell, Michael Mozer, and Yoshua Bengio. 2021. Coordination among neural modules through a shared global workspace.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines.

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. 2015. Learning to transduce with unbounded memory. In *Proc. of NeurIPS*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Armand Joulin and Tomás Mikolov. 2015. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Proc. of NeurIPS*.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A. Smith. 2021a. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *Proc. of ICLR*.

Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A. Smith. 2021b. Finetuning pretrained transformers into RNNs. In *Proc. of EMNLP*.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and Francois Fleuret. 2020. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proc. of ICML*.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *Proc. of ICLR*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL*.

Hung Le, Truyen Tran, and Svetha Venkatesh. 2020. Self-attentive associative memory. In *Proc. of ICML*.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proc. of ICML*.

Tao Lei, Yu Zhang, Sida I. Wang, Hui Dai, and Yoav Artzi. 2018. Simple recurrent units for highly parallelizable recurrence. In *Proc. of EMNLP*.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Łukasz Kaiser, and Noam Shazeer. 2018. Generating Wikipedia by summarizing long sequences. In *Proc. of ICLR*.

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.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proc. of ICLR*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Proc. of NeurIPS*.

Lesly Miculicich, Dhananjay Ram, Nikolaos Pappas, and James Henderson. 2018. Document-level neural machine translation with hierarchical attention networks. In *Proc. of EMNLP*.

Sebastian Nagel. 2016. News dataset available. `https://commoncrawl.org/2016/10/news-dataset-available/`.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. In *Proc. of ICML*.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. 2021. Random feature attention. In *Proc. of ICLR*.

Hao Peng, Roy Schwartz, Sam Thomson, and Noah A. Smith. 2018. Rational recurrences. In *Proc. of EMNLP*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proc. of WMT*.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. 2020. Compressive transformers for long-range sequence modelling. In *Proc. of ICLR*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proc. of EMNLP*.

Aurko Roy, Mohammad Taghi Saffar, David Grangier, and Ashish Vaswani. 2020. Efficient content-based sparse attention with routing transformers. *TACL*.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. 2021. Linear transformers are secretly fast weight programmers. In *Proc. of ICML*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proc. of ACL*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*.

Sainbayar Sukhbaatar, Da Ju, Spencer Poff, Stephen Roller, Arthur Szlam, Jason Weston, and Angela Fan. 2021. Not all memories are created equal: Learning to forget by expiring. In *Proc. of ICML*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey.

Trieu H. Trinh and Quoc V. Le. 2018. A simple method for commonsense reasoning.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NeurIPS*.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. When a good translation is wrong in context: Context-aware machine translation improves on deixis, ellipsis, and lexical cohesion. In *Proc. of ACL*.

Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. 2020. Fast transformers with clustered attention. In *Proc. of NeurIPS*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proc. of ICLR*.

Shuohang Wang, Luowei Zhou, Zhe Gan, Yen-Chun Chen, Yuwei Fang, Siqi Sun, Yu Cheng, and Jingjing Liu. 2020a. Cluster-Former: Clustering-based sparse transformer for long-range dependency encoding. *Findings of ACL*.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020b. Linformer: Self-attention with linear complexity.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proc. of ICLR*.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proc. of NAACL*.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proc. of NAACL*.

Dani Yogatama, Yishu Miao, Gabor Melis, Wang Ling, Adhiguna Kuncoro, Chris Dyer, and Phil Blunsom. 2018. Memory architectures in recurrent neural network language models. In *Proc. of ICLR*.

Manzil Zaheer, Guru Guruganesh, 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 *Proc. of NeurIPS*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Proc. of NeurIPS*.

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 *Proc. of ICCV*.

Appendices

A Other ABC Models

A.1 Sparse Local-to-global Attention

It sparsifies attention pattern to reduce the number of tokens that are attended to (Beltagy et al., 2020; Zaheer et al., 2020, *inter alia*). All queries attend to a subset of $n < N$ "global tokens," while ignoring others. Therefore the effective context size is reduced to n. The global tokens are usually preselected by positions according to some heuristics. Local-to-global attention is an instance of ABC: it can be recovered by letting $\boldsymbol{\phi}_t = \mathbf{e}_i$ if x_t is the ith global token ($i = 1, \ldots, n$), and the zero vectors for others.

A.2 Random Memory Control

As a baseline, ABC_{RD} stores each token in a randomly-selected memory slot. This is achieved by letting $\boldsymbol{\phi}_t = \mathbf{e}_{i_t}$, where i_t is uniformly drawn from $\{1, \ldots, n\}$ for each t. It is designed as a baseline to ABC_{MLP} and Linformer to quantify the differences between random and learned bounded-memory control.

Random sparse attention patterns are explored by Zaheer et al. (2020), where a subset of $n < N$ tokens are randomly selected to be attended to by all tokens. ABC_{RD} is different, and it attends to *all* tokens, but randomly "squash" them into an n-slot memory.

A.3 Compressive Transformer with Mean Pooling

The compressive transformer (Rae et al., 2020) explores various ways to "squash" long context into smaller and more compact representations. It achieves state-of-the-art performance on several language modeling benchmarks. We show that at least the mean-pooling variant of the compressive transformer can be seen as an ABC instance.

The mean-pooling variant of the compressive transformer compresses the context by

$$\begin{aligned}\mathbf{K} &= \left[\mathbf{k}_1, \ldots, \mathbf{k}_N\right]^\top \in \mathbb{R}^{N \times d} \\ \rightarrow \widetilde{\mathbf{K}} &= \Big[\underbrace{\left(\mathbf{k}_1 + \cdots + \mathbf{k}_c\right)}_{c} / c, \\ &\qquad \underbrace{\left(\mathbf{k}_{c+1} + \cdots + \mathbf{k}_{2c}\right)}_{c} / c \ldots, \\ &\qquad \underbrace{\left(\mathbf{k}_{N-c+1} + \cdots + \mathbf{k}_N\right)}_{c} / c\Big]^\top \in \mathbb{R}^{n \times d}.\end{aligned}$$

where $c = N/n$ is the compression ratio. Here $N \bmod n = 0$ is assumed, since otherwise the sequence can be padded to.

The above model is an ABC instance by letting

$$\boldsymbol{\phi}_i = \mathbf{e}_{\lfloor (i-1)/c \rfloor + 1} / c. \tag{8}$$

A.4 Dilated Convolution Attention Patterns

The dilated attention pattern is similar to the sliding window attention and only considers the context within a predefined window. It differs in that it attends to every other token:

$$\widetilde{\mathbf{K}}_t = [\mathbf{k}_{t-2n+2}, \mathbf{k}_{t-2n+4}, ..., \mathbf{k}_{t-2}, \mathbf{k}_t]^\top. \tag{9}$$

It can be simulated with two separate queues $\widetilde{\mathbf{K}}^{\text{odd}}$ and $\widetilde{\mathbf{K}}^{\text{even}}$:

$$\widetilde{\mathbf{K}}^{\text{odd}}_t = \begin{cases} \mathbf{U}\widetilde{\mathbf{K}}^{\text{odd}}_{t-1} + \mathbf{e}_n \otimes \mathbf{k}_t, & \text{if } t \text{ is odd} \\ \widetilde{\mathbf{K}}^{\text{odd}}_{t-1}, & \text{otherwise} \end{cases}$$

$$\widetilde{\mathbf{K}}^{\text{even}}_t = \begin{cases} \mathbf{U}\widetilde{\mathbf{K}}^{\text{even}}_{t-1} + \mathbf{e}_n \otimes \mathbf{k}_t, & \text{if } t \text{ is even} \\ \widetilde{\mathbf{K}}^{\text{even}}_{t-1}, & \text{otherwise} \end{cases}$$

Likewise for the values. Depending on t, the query attends to one of the two queues: $\text{output} =$

$$\begin{cases} \left(\widetilde{\mathbf{V}}^{\text{odd}}\right)^\top \text{softmax}(\widetilde{\mathbf{K}}^{\text{odd}}\mathbf{q}_t), & \text{if } t \text{ is odd} \\ \left(\widetilde{\mathbf{V}}^{\text{even}}\right)^\top \text{softmax}(\widetilde{\mathbf{K}}^{\text{even}}\mathbf{q}_t), & \text{otherwise.} \end{cases}$$

The above implementation could incur considerable amount of overhead and may be actually more expensive than the the original dilated window formulation. Therefore it has more conceptual value than practical value.

A.5 Shared Workspace and Linear Unified Nested Attention

Concurrently to this work, shared workspace (SW; Goyal et al., 2021) and linear unified nested attention (LUNA; Ma et al., 2021) also propposed methods to learn contextualized memory control strategies. Both can be seen as instances of ABC. At layer ℓ, their $\boldsymbol{\phi}^\ell_i$ is a function of previous layer's memory $\widetilde{\mathbf{X}}^{\ell-1} \in \mathbb{R}^{n \times d}$ and current layer's input $\mathbf{X}^\ell \in \mathbb{R}^{N \times d}$:

$$\boldsymbol{\phi}_i = \left[\text{softmax}\left(\widetilde{\mathbf{X}}^{\ell-1}\mathbf{X}^{\ell\top}\right)\right]_{:,i}, \tag{10}$$

where $[\cdot]_{:,i}$ denotes the ith column of a matrix. Query, key, and value projections are suppressed for notation clarity.

SW and LUNA reveal the entire sequence to the control vectors, by constructing $\boldsymbol{\phi}$ as a function of previous layer's memory. Although both admit the recurrent computation as all ABC models do, they are ill-suited for causal attention and autoregressive decoding, since future information is "leaked" to $\boldsymbol{\phi}_i$ from the previous layer. LUNA resorts to a variant of Katharopoulos et al. (2020) in causal attention (Ma et al., 2021). In contrast, ABC_{MLP} never conditions $\boldsymbol{\phi}_i$ on previous layer's memory, but only on the current layer's input.

B More Details about ABC-MLP

B.1 Normalization in Causal Attention

An equivalent implementation to Eq. 7 is to normalize $\widetilde{\mathbf{K}}$ and $\widetilde{\mathbf{V}}$ instead of $\boldsymbol{\phi}_i$ vectors:

$$\boldsymbol{\alpha}_i = \exp\left(\mathbf{W}_{\boldsymbol{\phi}}\mathbf{x}_i\right), \quad \boldsymbol{\phi}_i = \boldsymbol{\alpha}_i,$$
$$\bar{\mathbf{K}} = \widetilde{\mathbf{K}} \Big/ \sum_{j=1}^{N} \boldsymbol{\alpha}_j. \quad \bar{\mathbf{V}} = \widetilde{\mathbf{V}} \Big/ \sum_{j=1}^{N} \boldsymbol{\alpha}_j.$$
$$\text{output} = \bar{\mathbf{V}}^\top \text{softmax}(\bar{\mathbf{K}}\mathbf{q}).$$

$\mathbf{M}/\mathbf{z}$ divides the ℓth row of matrix $\mathbf{M}$ by vector $\mathbf{z}$'s ℓth dimension. This admits a linear complexity computation graph for the causal variant of ABC_{MLP}.

B.2 Higher-Dimensional Case of Example 1

This section generalizes Example 1 to higher dimensional cases. Assume that the constant-sized memory has n slots. $\boldsymbol{\phi}_i$ is cauculated as in Eq. 7. Then $\widetilde{\mathbf{K}} = \sum_{i=1}^{N} \boldsymbol{\phi}_i \otimes \mathbf{k}_i \in \mathbb{R}^{n\times d}$. Each row of $\widetilde{\mathbf{K}}$ can be seen as a separate attention mechanism with a pseudo query. Let $[\cdot]_\ell$ denote the ℓth row/dimension of a matrix/vector. Then for any $\ell = 1, \ldots, n$,

$$\begin{aligned}
\left[\widetilde{\mathbf{K}}\right]_\ell &= \sum_{i=1}^{N} [\boldsymbol{\phi}_i]_\ell \otimes \mathbf{k}_i \\
&= \sum_{i=1}^{N} \frac{\exp([\mathbf{W}_\phi]_\ell \cdot \mathbf{x}_i)}{\sum_{j=1}^{N} \exp([\mathbf{W}_\phi]_\ell \cdot \mathbf{x}_j)} \mathbf{k}_i^\top \\
&= \text{attn}\left([\mathbf{W}_\phi]_\ell, \{\mathbf{x}_i\}_{i=1}^N, \{\mathbf{k}_i\}_{i=1}^N\right)^\top \in \mathbb{R}^{1\times d}.
\end{aligned}$$

In other words, there are n attention mechanisms in total, each with a separately-parameterized pseudo-query $[\mathbf{W}_\phi]_\ell$. They summarize the context for n times in parallel, each producing a d-dimensional vectors. These output vectors are then stacked into n-by-d memory $\widetilde{\mathbf{K}}$. $\widetilde{\mathbf{V}}$ is similar.

C Experimental Details

C.1 Language Modeling

We closely build on Baevski and Auli (2019) and Kasai et al. (2021b). The hyperparameters are summarized in Table 10. All models are trained on 4 A100 GPUs.

C.2 Machine Translation

We experiment with a sentence-level (WMT14 EN-DE, Bojar et al., 2014) and a document-level benchmark (IWSLT14 ES-EN, Cettolo et al., 2014) to assess model performance over various sequence lengths. The preprocessing and data splits of WMT14 EN-DE follow Vaswani et al. (2017). A 32,768 byte pair encoding (BPE; Sennrich et al., 2016) vocabulary is shared between source and target languages. For IWSLT14, we follow Miculicich et al. (2018) and use the *dev2010* subset for development and *tst2010-2012* for testing. The tokenization is also the same as Miculicich et al. (2018): we tokenize and truecase Spanish and English with Moses (Koehn et al., 2007) and run byte-pair encoding with 30k splits, shared between the two languages. The final dataset contains 1421, 8, and 42 documents for training, development, and testing. On average, each document contains 126.7 sentences, and each sentence contains 21.7(ES)/22.5(EN) BPE subwords. We use a sliding window with length-4 and stride-one to generate our dataset. During inference, we use predicted context on the target side.

We average the checkpoints from the last five epochs to obtain the final model (Vaswani et al., 2017). In inference, we apply beam search with size 5 and length penalty 0.6. Other hyperparameters are summarized in Table 11. All models are trained on 4 RTX 2080 Ti GPUs.

Additional machine translation results. In addition to the results presented in §5.2, Table 8 further compares, on the WMT14 EN-DE dataset, the clustering-based (§3.2) and sliding-window (§3.3) models of ABC, as well as ReLU and sigmoid variants of ABC_{MLP}. Clustering and sliding-window ABC variants *underperform* ABC_{MLP} with the same memory sizes by more than 0.5 BLEU. Both ReLU and sigmoid *underperform* their exp counterpart.

MLP-exp-all replaces the encoder's softmax attention modules with ABC, in addition to the decoder's. It underperforms ABC_{MLP} by only 0.3 BLEU.

Model	ϕ	Cross n	Causal n	Encoder n	BLEU
BASE	-	-	-	-	27.2
ABC	Window	32	32	-	26.3
	Cluster	32	32	-	26.8
	MLP-ReLU	32	8	-	-
	MLP-ReLU	32	32	-	26.4
	MLP-sigmoid	32	8	-	26.8
	MLP-sigmoid	32	32	-	27.0
	MLP-exp	32	8	-	27.1
	MLP-exp	32	32	-	**27.3**
	MLP-exp-all	32	32	32	27.0

Table 8: ABC variants' performance (SacreBLEU) on the WMT14 EN-DE test set for sentence-level machine translation. MLP-ReLU with 32/8 memory sizes fails to converge. MLP-exp-all applies ABC in both the encoder and the decoder, while others only in the decoders.

Figure 1b compares ABC_{MLP}'s (32-8 memory sizes) attention memory overhead with softmax attention's. Following Kasai et al. (2021b), we consider a synthetic sequence-to-sequence generation task with varying sequence lengths. A batch size of 16 and greedy decoding is used. The models are of the same size as those in §5.2.

C.3 Masked Language Model Finetuning

Our data for continued pretraining is a concatenation of BookCorpus (Zhu et al., 2015), English Wikipedia, OpenWebText (Gokaslan and Cohen, 2019), and RealNews (Zellers et al., 2019). Our data differs from RoBERTa's pretraining data, which we do *not* have access to. We replace their CC-News (Nagel, 2016) with RealNews, and drop Stories (Trinh and Le, 2018). At the time of this project, the public access to the Stories dataset is broken.[10] Our machine does *not* have a large enough memory to load all the data. We therefore split the training data into 20 shards, after shuffling. Other preprocessing is the same as Liu et al. (2019).[11] The hyperparameters for continued pretraining follow base-sized RoBERTa, part of which are summarized in Table 12. All models are trained on a single TPU v3 accelerator.

For downstream task finetuning, we use the same hyperparameters as Liu et al. (2019).[12] Table 13 briefly describes the tasks. The readers are referred to Wang et al. (2019) for futher details.

[10] https://console.cloud.google.com/storage/browser/commonsense-reasoning/reproduce/stories_corpus?pli=1

[11] https://github.com/pytorch/fairseq/blob/master/examples/roberta/README.pretraining.md

[12] https://github.com/pytorch/fairseq/blob/master/examples/roberta/README.glue.md

Data	Train	Dev.	Test	Vocab.	Sent./doc
WikiText-103	103M	218K	246K	268K	-
WMT14 EN-DE	4.5M	3K	3K	32K	-
IWSLT14 ES-EN	1713	8	56	30K	121.5

Table 9: Statistics for the datasets. WikiText-103 split sizes are in number of tokens, WMT14 in number of sentences, and IWSLT14 in number of documents.

Hyperprams.	B&A	Kasai
# Layers	16	32
# Heads	8	8
Embedding Size	1024	1024
Head Size	128	128
FFN Size	4096	4096
Batch Size	64	64
Learning Rate	1.0	1.0
Dropout	0.3	0.3
Layer Dropout	-	0.2
Memory size	$[32, 64]$	64

Table 10: Hyperparameters used in the language modeling experiments. B&A: Baevski and Auli (2019); Kasai: Kasai et al. (2021b).

Hyperprams.	WMT14	IWSLT14
# Layers	6	6
# Heads	8	8
Embedding Size	512	512
Head Size	64	64
FFN Size	2048	1024
Warmup Steps	6000	4000
Dropout	0.1	0.3
Cross Attn. n	32	128
Causal Attn. n	8	64

Table 11: Hyperparameters used in the machine translation experiments.

Hyperprams.	Values
# Layers	12
# Heads	12
Embedding Size	768
Head Size	64
FFN Size	3072
Dropout	0.1
Memory Size	$[64, 128]$

Table 12: Hyperparameters for continued pretraining in the masked language model finetuning experiments.

Data	Task	Train	Dev.
MNLI	Entailment	392K	9.8K
QNLI	Entailment	105K	5.5K
QQP	Paraphrase	363K	40K
SST-2	Sentiment	67K	873

Table 13: GLUE datasets and statistics. MNLI: Williams et al. (2018); QNLI is compiled by GLUE's authors using Rajpurkar et al. (2016); QQP: Csernai (2017, accessed September 1, 2020); SST-2: Socher et al. (2013).