ABC: ATTENTION WITH BOUNDED-MEMORY CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformer architectures have achieved state-of-the-art results on a variety of sequence modeling 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 apart. 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 finetuning show that our approach outperforms previous efficient attention models; compared to the 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 to modeling in natural language processing (Vaswani et al., 2017), computer vision (Dosovitskiy et al., 2021), computational biology (Jumper et al., 2021), and other application areas. 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 the memory using a softmax-weighted linear combination, with an 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-size 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 to be *not* applicable in causal attention (Tay et al., 2020) or autoregressive decoding. Through the ABC view, we show that it actually *is*, and achieves competitive performance in our machine translation experiments. This finding reveals new insights into the properties of the important Linformer model.

ABC connects existing models that would otherwise seem distinct, reveals new insights into established approaches, and inspires new efficient attention architectures. We explore its applications in transformers, as a drop-in substitute for the canonical softmax attention. This abstraction offers a novel lens that can help future research in the analysis of transformers, where the theoretical insights are still catching up with the 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. Our analysis shows that the efficiency improvements are more prominent for long sequences, suggesting that the asymptotic savings are more appealing in application areas involving working with very long sequences. We will release our code upon publication.

2 An Outer-Product View of Attention

This section reviews softmax attention and presents an outer-product memory perspective of it, which allows for a smooth transition to later discussion.

In attention, a sequence of **query** vectors $\{\mathbf{q}_i\}_{i=1}^N$ 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] Reading from the memory with query $\mathbf{q}$ produces a d-dimensional output vector using a softmax-normalized linear combination: $\text{attn}(\mathbf{q}, \{\mathbf{k}_i\}_{i=1}^N, \{\mathbf{v}_i\}_{i=1}^N) = \mathbf{V}^\top \text{softmax}(\mathbf{K}\mathbf{q}) \in \mathbb{R}^d$. 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:

$$\mathbf{K} = \mathbf{I}\mathbf{K} = \sum_{i=1}^N \mathbf{e}_i \otimes \mathbf{k}_i, \quad \mathbf{V} = \mathbf{I}\mathbf{V} = \sum_{i=1}^N \mathbf{e}_i \otimes \mathbf{v}_i. \tag{1}$$

$\mathbf{I}$ is the identity matrix, and $\otimes$ denotes the outer product between vectors (i.e., $[\mathbf{x} \otimes \mathbf{y}]_{i,j} = x_i y_j$). The N-dimensional $\mathbf{e}_i$ vectors form the standard basis, i.e., $\mathbf{e}_i$ 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 $\mathbf{v}_i$:

$$\mathbf{e}_i \otimes \mathbf{k}_i = \big[\underbrace{0, \dots 0}_{i-1}, 1, \underbrace{0, \dots, 0}_{N-i}\big]^\top \otimes \mathbf{k}_i = \big[\underbrace{\mathbf{0}}_{d\times(i-1)}; \mathbf{k}_i; \underbrace{\mathbf{0}}_{d\times(N-i)}\big]^\top \in \mathbb{R}^{N\times d}. \tag{2}$$

The N-by-d matrix on the right-hand side has its ith row being $\mathbf{k}_i^\top$ and all others being 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 Control

A straightforward way to improve attention's efficiency is to bound its memory size. Our outer-product view of attention (Eq. 1) provides a straightforward way to devise this, by replacing the $\mathbf{e}_i$ vectors with control vectors that select $n \ll N$ vectors to attend to. We denote 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*. $\{\boldsymbol{\phi}_i \in \mathbb{R}^n\}_{i=1}^N$ denotes a sequence of **control** vectors:

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

with the output similarly calculated with a softmax-normalized linear sum, but over $\widetilde{\mathbf{K}}$ and $\widetilde{\mathbf{V}}$:

$$\text{ABC}\left(\mathbf{q}, \{\mathbf{k}_i\}_{i=1}^N, \{\mathbf{v}_i\}_{i=1}^N, \{\boldsymbol{\phi}_i\}_{i=1}^N\right) = \widetilde{\mathbf{V}}^\top \text{softmax}(\widetilde{\mathbf{K}}\mathbf{q}). \tag{4}$$

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

Ways to construct $\{\boldsymbol{\phi}_i\}$ vary, which we will soon discuss. 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, as we will explore in our experiments.

$$\widetilde{\mathbf{K}}_{t+1} = \widetilde{\mathbf{K}}_t + \boldsymbol{\phi}_{t+1} \otimes \mathbf{k}_{t+1}, \quad \widetilde{\mathbf{V}}_{t+1} = \widetilde{\mathbf{V}}_t + \boldsymbol{\phi}_{t+1} \otimes \mathbf{v}_{t+1}. \tag{5}$$

$\widetilde{\mathbf{K}}_t$ and $\widetilde{\mathbf{V}}_t$ can be seen as the **hidden state** that accumulates the information of prefix $\mathbf{x}_{\leq t}$.

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

3.1 LINFORMER

Linformer (Wang et al., 2020b) is an established efficient transformer variant that proves 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 *sequence length* dimension to an n-by-d dimensional memory,

$$\widetilde{\mathbf{K}}^{\text{LF}} = \mathbf{W}^{\text{LF}}\mathbf{K}, \quad \widetilde{\mathbf{V}}^{\text{LF}} = \mathbf{W}^{\text{LF}}\mathbf{V}, \tag{6}$$

which are then used for attention computation with Eq. 4. This yields a linear complexity in the sequence length. Linformer is an ABC instance where $\boldsymbol{\phi}_i^{\text{LF}} = \mathbf{W}^{\text{LF}}_{:,i}$ (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 sequence length. The ABC lens reveals that, in fact, Linformer *is* applicable in causal attention. Like all ABC models, it admits the linear complexity recurrent computation as in 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 is indeed applicable to causal attention, with 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. To reduce the effective context size, it clusters keys/values, and only attends to the centroids. Formally, $\{\mathbf{k}_i\}$ are grouped into $n < N$ clusters $\{\widetilde{\mathbf{k}}^{\text{CL}}_j\}_{j=1}^n$, and $\{\mathbf{v}_i\}$ into $\{\widetilde{\mathbf{v}}^{\text{CL}}_j\}_{j=1}^n$.[3] Let an N-by-n binary matrix $\mathbf{M}$ denote the cluster membership matrix, and it is 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 centroids for keys and values are respectively

$$\widetilde{\mathbf{k}}^{\text{CL}}_j = \sum_{i=1}^N \frac{M_{i,j}}{\sum_{\ell=1}^N M_{\ell,j}} \mathbf{k}_i, \quad \widetilde{\mathbf{v}}^{\text{CL}}_j = \sum_{i=1}^N \frac{M_{i,j}}{\sum_{\ell=1}^N M_{\ell,j}} \mathbf{v}_i. \tag{7}$$

Attention over the centroids then proceeds as Eq. 4, in the same way as the softmax attention, but over $\{\widetilde{\mathbf{k}}^{\text{CL}}_j\}_{j=1}^n$ and $\{\widetilde{\mathbf{v}}^{\text{CL}}_j\}_{j=1}^n$: $(\widetilde{\mathbf{V}}^{\text{CL}})^\top \operatorname{softmax}(\widetilde{\mathbf{K}}^{\text{CL}}\mathbf{q})$, where $\widetilde{\mathbf{K}}^{\text{CL}} = [\widetilde{\mathbf{k}}^{\text{CL}}_1, \ldots, \widetilde{\mathbf{k}}^{\text{CL}}_n]^\top =$

$$\sum_{j=1}^n \mathbf{e}_j \otimes \widetilde{\mathbf{k}}^{\text{CL}}_j = \sum_{j=1}^n \mathbf{e}_j \otimes \sum_{i=1}^N \frac{M_{i,j}}{\sum_{\ell=1}^N M_{\ell,j}} \mathbf{k}_i = \sum_{i=1}^N \left(\sum_{j=1}^n \mathbf{e}_j \frac{M_{i,j}}{\sum_{\ell=1}^N M_{\ell,j}} \right) \otimes \mathbf{k}_i. \tag{8}$$

By the last line, 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.

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

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

Model	Section	ϕ_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	Appx. A.1	$\mathbf{e}_i$ if $\mathbf{x}_t$ is the ith global token	
ABC$_{\text{RD}}$	Appx. A.2	$\mathbf{e}_{i_t}$, where $i_t \sim \text{unif}\{1, n\}$	
Comp. Trans.	Appx. 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	$\exp\left(\mathbf{W}_{\boldsymbol{\phi}} \mathbf{x}_t\right) / \sum_{i=1}^{t} \exp\left(\mathbf{W}_{\boldsymbol{\phi}} \mathbf{x}_t\right)$	

Table 1: A comparison of memory-organization strategies 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.

3.3 Sliding-Window Attention

In some applications, being able to remove entries from the memory could be beneficial. For example, older context can be removed to clear up slots for more recent contexts, 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 (Parmar et al., 2018; Beltagy et al., 2020; Zaheer et al., 2020, *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:

$$\widetilde{\mathbf{K}}^{\text{WD}}_t = [\mathbf{k}_{t-n+1}, ..., \mathbf{k}_t]^\top, \quad \widetilde{\mathbf{V}}^{\text{WD}}_t = [\mathbf{v}_{t-n+1}, ..., \mathbf{v}_t]^\top, \tag{9}$$

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:

$$\mathbf{U}\widetilde{\mathbf{K}}^{\text{WD}}_t = \mathbf{U}\big[\underbrace{\mathbf{k}_{t-n+1}, \ldots, \mathbf{k}_t}_{n}\big]^\top = \big[\underbrace{\mathbf{k}_{t-n+2}, \ldots, \mathbf{k}_{t-1}, \mathbf{k}_t}_{n-1}, \mathbf{0}\big]^\top \in \mathbb{R}^{n \times d}. \tag{10}$$

Then the most recent token can be put into the slot freed up:

$$\widetilde{\mathbf{K}}^{\text{WD}}_{t+1} = \mathbf{U}\widetilde{\mathbf{K}}^{\text{WD}}_t + \mathbf{e}_n \otimes \mathbf{k}_{t+1}. \tag{11}$$

$\mathbf{U}$ and n-dimensional $\boldsymbol{\phi}_t = \mathbf{e}_n$ ensure a first-in-first-out queue. Dilated and stride convolution patterns (Beltagy et al., 2020) can be similarly recovered, but with a double-ended queue (Appendix A.4).

Recurrently multiplying $\mathbf{U}$ simulates the discrete pop operation (Grefenstette et al., 2015; Joulin & Mikolov, 2015; Yogatama et al., 2018) in a differentiable way. This is reminiscent of recurrent neural networks, while in this case $\mathbf{U}$ is fixed and *never* updated as parameters. It is exciting to explore learning such a recurrent matrix, but is beyond the scope of this work.

Discussion. In addition to 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 (Appendix A).

ABC provides a unified perspective of these approaches, and at the same time points out their limitations: their control strategies are context-agnostic. In response to this, we propose to learn a contextualized strategy from data in §4. Closing this section, Table 1 analyzes various ABC models.

4 Learned Memory Control

The ABC abstraction connects several existing approaches that would otherwise seem distinct. This could inspire the design of new architectures. We hypothesize that learning a contextualized strategy could achieve better performance in practice. This section introduces ABC$_{\text{MLP}}$, which parameterizes $\boldsymbol{\phi}$ with a single-layer multi-layer perceptron (MLP) that takes as input the token representation, and

determines which slots to write it into and how much, if at all.

$$\boldsymbol{\alpha}_i = \mathbf{exp}\ (\mathbf{W}_{\boldsymbol{\phi}}\mathbf{x}_i)\,, \quad \boldsymbol{\phi}_i = \boldsymbol{\alpha}_i \Big/ \sum_{j=1}^{N} \boldsymbol{\alpha}_j. \tag{12}$$

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$_{\text{MLP}}$ 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. Goyal et al. (2021) and Ma et al. (2021), concurrently to this work, also 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$_{\text{MLP}}$, 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 amount of additional parameters ABC$_{\text{MLP}}$ incurs is small: we tie $\boldsymbol{\phi}$-MLP's parameters across different layers, which adds less than 1% parameters to the models. This sharing is inspired by the both best-performing and most parameter-efficient configuration by Wang et al. (2020b).

ABC$_{\text{MLP}}$: context-agnostic then context-dependent attention. We now dissect ABC$_{\text{MLP}}$ and show that it can be seen as a cascade of two attention mechanisms: one with a learned context-agnostic 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$_{\text{MLP}}$ with a *single* memory slot ($n = 1$). It is parameterized with a learned vector $\mathbf{w}_\phi$, and $\phi_i = \exp(\mathbf{w}_\phi \cdot \mathbf{x}_i) / \sum_{j=1}^{N} \exp(\mathbf{w}_\phi \cdot \mathbf{x}_j)$. Since ϕ_i is a scalar here, $\phi_i \otimes \mathbf{k}_i = \phi_i \mathbf{k}_i^\top$.

$$\widetilde{\mathbf{K}}^\top = \sum_{i=1}^{N} (\phi_i \otimes \mathbf{k}_i)^\top = \sum_{i=1}^{N} \frac{\exp(\mathbf{w}_\phi \cdot \mathbf{x}_i)}{\sum_{j=1}^{N} \exp(\mathbf{w}_\phi \cdot \mathbf{x}_j)} \mathbf{k}_i = \mathrm{attn}\left(\mathbf{w}_\phi, \{\mathbf{x}_i\}_{i=1}^N, \{\mathbf{k}_i\}_{i=1}^N\right). \tag{13}$$

In other words, $\widetilde{\mathbf{K}}$ uses $\mathbf{w}_\phi$ as a "pseudo-query" to attend to $\{\mathbf{x}_i\}$ and $\{\mathbf{k}_i\}$. Likewise, $\widetilde{\mathbf{V}}^\top = \mathrm{attn}(\mathbf{w}_\phi, \{\mathbf{x}_i\}_{i=1}^N, \{\mathbf{v}_i\}_{i=1}^N)$. Despite its similarity to the standard softmax attention, Example 1 has a more efficient linear complexity in sequence lengths. $\mathbf{w}_\phi$'s being context-independent is the key to the savings. Appendix B.3 compares its complexity to softmax attention's.

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 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. Appendix 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

[4] We also experiment with other non-negative activation functions: ReLU and $\mathrm{sigmoid}$ (Appendix 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. Appendix B.1 details a linear complexity computation graph of causal $\boldsymbol{\phi}_i$.

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

and attend 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 works "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_{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_{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 neural Turing machines (NTM; Graves et al., 2014). ABC_{MLP} computes the control vectors $\boldsymbol{\phi}_i$ 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 & 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 finetuning (§5.3). Dataset statistics and implementation details are summarized in Appendix C.

5.1 LANGUAGE MODELING

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

- ABC_{MLP} as described in §4, learns a contextualized exp-MLP as the $\boldsymbol{\phi}$ function.
- Linformer (Wang et al., 2020b), as described in §3.1.
- ABC_{RD} stores each token in a randomly-selected memory slot with $\boldsymbol{\phi}_t = \mathbf{e}_{i_t}$. 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 control.

We consider two model size settings:

- The first one follows Baevski & Auli (2019). It compares models with 16 layers and around 242M parameters. They train with 512-token segments; at evaluation time, the context size is 0 or 480, i.e., a 0 or 480 length prefix is attached to each evaluation segment.
- The second one follows Kasai et al. (2021b), aiming to compare ABC_{MLP} head-to-head to several recently-proposed kernel-based efficient attention variants: ELU (Katharopoulos et al., 2020), RFA (Peng et al., 2021), and T2R (Kasai et al., 2021b). This setup uses 32 layers, applies layer dropout (Fan et al., 2020), and evaluates with a 256 context size; otherwise it is the same as the 16-layer setting. All models have around 484M parameters.

Results. Table 2a compares ABC variants using Baevski & Auli (2019)'s 16-layer setting. Among ABC models, ABC_{MLP} achieves the best performance for both context sizes. With a memory size $n = 64$, ABC_{MLP} outperforms both Linformer and ABC_{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_{MLP}-32 outperforms its

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) Baevski & Auli (2019)'s 16-layer setting. 0 and 480 indicate context sizes at evaluation time (§5.1).

Model	n	Dev.	Test
†BASE	-	17.9	18.5
†ELU	128	22.0	22.8
†RFA	64	20.4	21.3
†T2R	64	20.1	20.8
ABC$_{\text{MLP}}$	64	**18.6**	**19.1**

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

Table 2: WikiText-103 language modeling perplexity (**lower is better**). n denotes the memory size. Bold numbers perform the best among linear-complexity models.

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

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 transformer baseline, with ABC$_{\text{MLP}}$-64 having the smallest gap of 0.5 perplexity. ABC$_{\text{MLP}}$-64 outperforms kernel-based methods by more than 1.7 test perplexity, using Kasai et al. (2021b)'s 32-layer setting (Table 2b).

5.2 MACHINE TRANSLATION

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

- Sentence-level translation with WMT14 EN-DE (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. Here we compare the ABC variants described in §5.1. Appendix C.2 further compares to the clustering-based (§3.2) and sliding-window (§3.3) variants of ABC. The BASE model that they are built on is our implementation of the base-sized transformer (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 (Peng et al., 2021; Kasai et al., 2021a); other components are kept the same. Appendix C.2 studies a model that replaces *all* softmax attention with ABC$_{\text{MLP}}$. We evaluate with SacreBLEU (Post, 2018).

Results. Table 3a summarizes sentence-level machine translation results on WMT14 EN-DE test set. Overall ABC$_{\text{MLP}}$ performs on par with BASE, with either 32-32 cross-causal memory sizes or

Model	n	MNLI	QNLI	QQP	SST-2	Avg.	Speed	Mem.
RoBERTa	-	87.6	92.8	91.9	94.8	91.8	1.0×	1.0×
RoBERTa-Ours	-	87.2	92.4	91.7	94.3	91.4	1.0×	1.0×
Linformer	64	85.3	91.8	90.8	92.4	90.1	1.7×	0.5×
Linformer	128	86.1	91.9	91.4	93.7	90.8	1.5×	0.6×
ABC$_{\text{MLP}}$	64	85.6	91.8	91.7	93.8	90.7	1.5×	0.5×
ABC$_{\text{MLP}}$	128	**87.1**	**92.6**	**91.8**	**94.4**	**91.5**	1.3×	0.6×

Table 4: Text classification development set accuracy. RoBERTa is the base-sized RoBERTa model, and its numbers are due to Liu et al. (2019). The second block continues pretraining RoBERTa based 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 RoBERTa-Ours. Inference speed (higher is better) and memory consumption (lower is better) are relative to RoBERTa's.

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 3b), except that ABC$_{\text{MLP}}$ slightly *underperforms* BASE by 0.2 BLEU. This result 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 varying-length sequences, especially long text decoding.

5.3 Masked Language Model Finetuning

Setting. We compare the ABC variants described in §5.1. It is interesting to pretrain 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 tranformer, and continue pretraining with the masked language modeling (MLM) objective on a concatenation of BookCorpus (Zhu et al., 2015), English Wikipedia, OpenWebText (Gokaslan & Cohen, 2019), and RealNews (Zellers et al., 2019).[7] Then the models are finetuned and evaluated on downstream classification datasets from the GLUE benchbark (Wang et al., 2018). This is an appealing setting: being able to convert a pretrained transformer into its efficient alternatives could avoid the majority of the expensive pretraining cost. In preliminary experiments, *all* ABC models fail on downstream datasets *without* continued pretraining.

Results. Table 4 compares ABC models against RoBERTa on downstream text classification accuracy. Following standard practice (Liu et al., 2019), we report results on development data. Continued training on our data, RoBERTa-Ours slightly *underperforms* RoBERTa. This could possibly be due to overfitting, or a discrepancy between our data and RoBERTa's. Linformer achieves competitive accuracy, aligned with Wang et al. (2020b)'s results. ABC$_{\text{MLP}}$ outperforms Linformer, and performs on par with or better than RoBERTa-Ours, affirming the benefits of using contextualized memory organization in masked language modeling. With sequence length 512 and batch size 16, both ABC$_{\text{MLP}}$ and Linformer achieve inference time speed gains and memory savings. Linformer is faster, since its linear projection is cheaper to compute than ABC$_{\text{MLP}}$'s MLP.[8] Their memory overhead is similar. ABC$_{\text{RD}}$ fails to converge in this experiment.

6 Analysis

Decoding efficiency varying sequence lengths. Scaling linearly in the sequence length, ABC's efficiency improvements could be more prominent for longer sequences. We study ABC$_{\text{MLP}}$'s decoding overhead varying sequence lengths. We follow Peng et al. (2021)'s setting, and consider

[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 & Le, 2018), whose public access is broken at the time of this work.

[8] Inference speed is measured on the same V100 GPU. Wang et al. (2020b) use batch sizes as large as the hardware allows, while we use the same 16 batch size for all models.

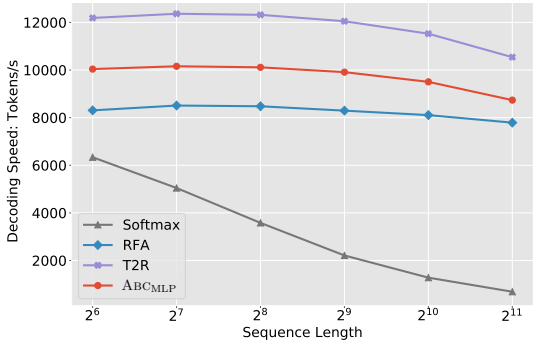

Figure 1: Decoding speed with varying sequence lengths. The setting follows Peng et al. (2021). All models run on a v2 TPU, with greedy decoding and batch size 16.

		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 5: ABC$_{\text{MLP}}$'s SacreBLEU on WMT14 EN-DE development data with varying memory sizes of cross and causal attention. All models apply greedy decoding, *without* checkpoint averaging.

a sequence-to-sequence generation experiment. Three linear-complexity models are compared: RFA (with cross-causal memory sizes of 256-128; Peng et al., 2021), T2R (32-4; Kasai et al., 2021b), and ABC$_{\text{MLP}}$ (32-8). These memory 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-complexity models achieve consistent decoding speed for different lengths (Figure 1), substantially outpacing the softmax attention baseline, especially for long sequences. In particular, ABC$_{\text{MLP}}$ decodes around 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 decodes about 20% faster. 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 decoding memory saving compared to softmax attention. The comparison is summarized in Figure 2 in Appendix C.2.

Memory size's impact on accuracy. Practically, one may want to minimize 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: as previous evidence shows, cross attention is more important than causal attention in machine translation (Michel et al., 2019; You et al., 2020). Our results (Table 5) 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: we showed that Linformer *is* efficiently applicable in causal attention, opposite to what was previously believed. ABC also has practical benefits 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.

REFERENCES

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. ETC: Encoding long and structured inputs in transformers. In *Proc. of EMNLP*, 2020.

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

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

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

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. Findings of the 2014 workshop on statistical machine translation. In *Proc. of WMT*, 2014.

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

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

Kornél Csernai. *First Quora Dataset Release: Question Pairs*, 2017, accessed September 1, 2020. URL https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs.

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. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. of ICLR*, 2021.

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

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

Anirudh Goyal, Aniket Didolkar, Alex Lamb, Kartikeya Badola, Nan Rosemary Ke, Nasim Rahaman, Jonathan Binas, Charles Blundell, Michael Mozer, and Yoshua Bengio. Coordination among neural modules through a shared global workspace. *arXiv:2103.01197*, 2021.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv:1410.5401*, 2014.

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

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

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

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Kathryn Tunyasuvunakool, Olaf Ronneberger, Russ Bates, Augustin Žídek, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Anna Potapenko, Andrew J Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Martin Steinegger, Michalina Pacholska, David Silver, Oriol Vinyals, Andrew W Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. High accuracy protein structure prediction using deep learning. *Nature*, 596:583–589, 2021.

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

Jungo Kasai, Hao Peng, Yizhe Zhang, Dani Yogatama, Gabriel Ilharco, Nikolaos Pappas, Yi Mao, Weizhu Chen, and Noah A. Smith. Finetuning pretrained transformers into RNNs. *arXiv:2103.13076*, 2021b.

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

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

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. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL*, 2007.

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

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

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

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

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

Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. Luna: Linear unified nested attention. *arXiv:2016.01540*, 2021.

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

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Proc. of NeurIPS*, 2019.

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

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

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

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

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

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

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

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

Aurko Roy, Mohammad Taghi Saffar, David Grangier, and Ashish Vaswani. Efficient content-based sparse attention with routing transformers. *arXiv: 2003.05997*, 2020.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *Proc. Int. Conf. on Machine Learning (ICML)*, Virtual only, July 2021.

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

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

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv: 2009.06732*, 2020.

Trieu H. Trinh and Quoc V. Le. A simple method for commonsense reasoning. *arXiv:1806.02847*, 2018.

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

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

Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention. *arXiv: 2007.04825*, 2020.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proc. of BlackboxNLP Workshop*, 2018.

Shuohang Wang, Luowei Zhou, Zhe Gan, Yen-Chun Chen, Yuwei Fang, Siqi Sun, Yu Cheng, and Jingjing Liu. Cluster-former: Clustering-based sparse transformer for long-range dependency encoding. *arXiv:2009.06097*, 2020a.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv: 2006.04768*, 2020b.

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

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

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

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

Weiqiu You, Simeng Sun, and Mohit Iyyer. Hard-coded Gaussian attention for neural machine translation. In *Proc. of ACL*, 2020.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. *arXiv: 2007.14062*, 2020.

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

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proc. of ICCV*, 2015.