Combiner: Full Attention Transformer with Sparse Computation Cost

Hongyu Ren†, Hanjun Dai⋆, Zihang Dai⋄, Mengjiao Yang⋄, Jure Leskovec†, Dale Schuurmans‡, Bo Dai⋄
†Stanford University, {hyren,jure}@cs.stanford.edu
⋆Google Research, Brain Team, {hadai,zihangd,sherry,y.schuurmans,bodai}@google.com
‡University of Alberta

Abstract

Transformers provide a class of expressive architectures that are extremely effective for sequence modeling. However, the key limitation of transformers is their quadratic memory and time complexity $O(L^2)$ with respect to the sequence length in attention layers, which restricts application in extremely long sequences. Most existing approaches leverage sparsity or low-rank assumptions in the attention matrix to reduce cost, but sacrifice expressiveness. Instead, we propose Combiner, which provides full attention capability in each attention head while maintaining low computation and memory complexity. The key idea is to treat the self-attention mechanism as a conditional expectation over embeddings at each location, and approximate the conditional distribution with a structured factorization. Each location can attend to all other locations, either via direct attention, or through indirect attention to abstractions, which are again conditional expectations of embeddings from corresponding local regions. We show that most sparse attention patterns used in existing sparse transformers are able to inspire the design of such factorization for full attention, resulting in the same sub-quadratic cost ($O(L \log(L))$ or $O(L \sqrt{L})$). Combiner is a drop-in replacement for attention layers in existing transformers and can be easily implemented in common frameworks. An experimental evaluation on both autoregressive and bidirectional sequence tasks demonstrates the effectiveness of this approach, yielding state-of-the-art results on several image and text modeling tasks.

1 Introduction

The Transformer [1] is a powerful neural network architecture that has demonstrated state-of-the-art performance in machine translation [2] and many other natural language processing (NLP) tasks via pretraining, using either unidirectional language modeling [3] or bidirectional language modeling [4–8]. It has also achieved excellent results in other domains like image recognition [9], code understanding [10], speech recognition [11], protein [12], music [13] and image [14] generative modeling. The core component of Transformer is the attention mechanism, which computes dependencies between all pairs of positions in a sequence. However, for a sequence of length $L$, the expressiveness of pairwise attention comes at a quadratic cost $O(L^2)$ in both time and memory consumption. This makes the vanilla Transformer [1] prohibitive for applications that involve long sequences, including high-resolution images, protein sequences, or raw speech signals [15], where the sequence length $L$ is often larger than 10,000 [14].

Recently, there have been several attempts to scale up attention to long sequences. A popular class of methods sparsifies the attention matrix with different sparsity patterns, including local

---

*indicates equal contribution. The work was completed during HR’s internship at Google Brain.

window [16, 17], local+stride [14], log-sparse [18], axial [19, 20], or learnable patterns through hashing [21] or clustering [22]. Sparse attention enjoys sub-quadratic cost, but is lossy in capturing all-pair relationships. Generally, sparse attention requires more layers [14, 20, 23] to achieve full autoregressive or bidirectional dependencies (or receptive fields [20]) for each location in a long sequence.

Alternatively, another line of research has tried to achieve scalability with an explicit low-rank assumption [24, 25] on the attention matrix or by using explicit feature maps of some kernels [26]. However these explicit low dimensional approximations might be too restricted for the potentially full rank attention matrix, which uses exponential kernels that are effectively infinite dimensional [27]. The Performer [28] is among the first works that attempts to approximate regular full-rank attention with the random feature trick [29]. However such random-feature based approaches [30] require many more bases to better approximate the exponential kernel [27], and empirically we found it produces inferior results in some sequence modeling tasks, such as density estimation.

In this paper we propose Combiner, a drop-in replacement for the vanilla quadratic attention mechanism with sub-quadratic computation and memory cost. Combiner still achieves full attention capability within each head of Multi-Head Attention, unlike approaches that adopt sparse or low-rank approximations. As we will discuss, the standard attention computed at each location can be seen as the conditional expectation of the value embeddings at all feasible locations given the current location. Based on such an understanding, Combiner explicitly approximates the conditional distribution in through a structured factorization of the probability space. Specifically, given a location \( x \), the probability of attending to location \( y \) can be either directly calculated via the query vector of \( x \) and key vector of \( y \), or indirectly through a local abstraction where \( x \) first attends to the key vector that represents a group of locations containing \( y \), and multiplying the probability of choosing \( y \) within that group. We refer to this model as Combiner since the conditional distributions in attention become a combination between several local attentions and direct attentions. This structured decomposition enables Combiner to take existing sparse attention patterns and convert them into corresponding design choices for probability factorizations that achieve full attention. As shown in Figure 1, Combiner achieves full attention with the same asymptotic complexity as sparse variants. Combiner can be easily implemented in most existing deep learning frameworks without the need for specialized hardware implementation, and is GPU/TPU friendly. In fact, both the fixed and learnable sparse attention patterns from many existing Transformer variants [14, 18, 20, 22] can be enhanced with such structured factorizations, with the same order of time or memory cost.

We validate Combiner on both autoregressive and bidirectional sequence modeling tasks over a variety of domains including text and images. We show that Combiner can achieve better perplexity of runtime, and achieves state of the art performance on density estimation on standard datasets CIFAR-10 (2.77 bits/dim) and ImageNet-64 (3.42 bits/dim), as well as the Long-Range Arena [31]. The implementation of Combiner can be found at https://github.com/google-research/google-research/tree/master/combiner.

## 2 Attention as Conditional Expectation

In this section, we revisit the formulation of the standard Transformer [1] from the perspective of conditional expectation, which inspires the derivation of Combiner.

Without loss of generality, we use a single sequence in the self-attention scenario. Given a sequence of \( L \) embeddings \( X = [x_1, x_2, \ldots, x_L] \), where \( X \in \mathbb{R}^{L \times d} \) and each embedding \( x_i \in \mathbb{R}^d \) is a \( d \)-dimensional vector, the core component of Transformer is the multi-head attention, where each head \( h \) is a scaled dot-product attention:

\[
A_h(X) = \text{softmax} \left( \frac{Q_h \mathbf{K}_h^\top}{\sqrt{d}} \right) V_h, \quad \left\{ Q_h = XW_h^Q, K_h = XW_h^K, V_h = XW_h^V \right\} \in \mathbb{R}^{L \times d},
\]

and the attention vector from each head \( A_h(X) \) is concatenated and projected:

\[
\text{MultiHeadAttn}(X) = [A_1(X), A_2(X), \ldots, A_H(X)] W^o, \quad W^o \in \mathbb{R}^{Hd \times d}
\]

Here \( H \) is the total number of heads per Transformer layer. In this paper, we focus on how to approximate full attention within each head of multi-head attention. For ease of notation, we drop the head index \( h \) whenever possible, and use lower-case letters \( x_i, q_i, k_i, v_i \in \mathbb{R}^d \) to denote rows in
Figure 1: Attention matrices of several instantiations of Combiner in the autoregressive setting. We transform several sparse attention patterns: Fixed (A) [14], Logsparse (B) [18] and Axial (C) [20] to Combiner-Fixed (D), Combiner-Logsparse (E) and Combiner-Axial (F). Combiner approximates the conditional expectation (3) with a combination of direct expectation (blue) and local expectation (yellow). Our instantiations (D)(E)(F) achieve full attention with the same sub-quadratic complexity.

\[ X, Q, K, V \] respectively, which corresponds to a location \( i \) in the original sequence of length \( L \). We use \( \{n\} \) to denote the set of positive integers \( \{1, 2, \ldots, n\} \).

For a position \( i \in \{1, 2, \ldots, L\} \), the attention formulation (1) can be viewed as conditional expectation of rows in \( V \). Specifically, since softmax outputs a probability distribution, we can rewrite (1) as

\[
A(x_i) = \mathbb{E}_{p(j|i)}[v_j], \quad p(j|i) = \frac{1}{Z(x_i)} \exp \left( \frac{q_i^T k_j^T}{\sqrt{d}} \right),
\]

where \( p(j|i) \) denotes the conditional probability at position \( j \) given the token at position \( i \) and the partition function \( Z(x_i) = \sum_{j \in \Omega_i} \exp \left( \frac{q_i^T k_j^T}{\sqrt{d}} \right) \) over support \( \Omega_i \). The support \( \Omega_i \) of \( p(j|i) \) defines the set of valid locations that the \( i \)-th token can attend to. For instance, the support set in autoregressive language modeling (LM) consists of all previous tokens, i.e., \( \Omega_{LM}^i = \{i\} \); in masked language modeling (MLM) the support consists of all tokens in the sequence, i.e., \( \Omega_{MLM}^i = \{L\} \). That is, \( \Omega_{LM} \) and \( \Omega_{MLM} \) represent the full attention capability respectively in the LM and MLM setting.

3 Combiner: Full Attention via Structured Conditional Expectation

The complexity of \( p(j|i) \) is the bottleneck of the computation for \( A(x_i) \). Generally, in existing sparse transformers, the support of \( p(j|i) \) is sparsified to reduce the computation and memory complexity, e.g., \( \Omega_{Sparse}^i \subset \Omega_{LM}^i \) for LM and \( \Omega_{Sparse}^i \subset \Omega_{MLM}^i \) for MLM, but this can lead to either reduced capacity or limited applicability. We defer detailed discussion of the full capacity of the model to Appendix A. In this section we introduce the Combiner, which achieves \( \Omega_{Combiner}^i = \Omega_{LM}^i \) for LM and \( \Omega_{Combiner}^i = \Omega_{MLM}^i \) for MLM, while still maintaining sub-quadratic computation and memory cost. Below we denote \( \Omega_i \) as the support for full attention if there is no ambiguity or need to distinguish between LM or MLM. We introduce the main design framework in Section 3.1 and possible parameterizations in Section 3.2. Then in Section 3.3 we analyze the trade-off of Combiner.

3.1 Local Factorization for Conditional Expectation

The main idea of Combiner is to exploit a hierarchical structure for conditional probability modeling in (3), which provides the opportunity for reducing computation complexity while maintaining the
same support. Specifically, we introduce support variables Ω
r
i
, for r = 0, . . . , n
i
and i ∈ [L]. The support variables are disjoint, i.e., Ω
r
i
∩ Ω
s
i
= \emptyset, ∀ r ≠ s, and \bigcup_{r=0}^{n_i} Ω
r
i
= Ω
i
. Then we can factorize

\[ p(j|i) = \sum_{r=0}^{n_i} p(j, Ω
r
i
|i) = \sum_{r=0}^{n_i} p(j|Ω
r
i
,i)p(Ω
r
i
|i) = p(j|Ω
i
,r,i)p(Ω
r
i
|i), \]  

(4)

where r
j
 denotes the index of the support to which j belongs. The last equation arises from the fact that the Ω
r
i
 are disjoint from each other (Ω
r
i
∩ Ω
s
i
 = \emptyset, ∀ r ≠ s). Therefore, there is only one support, Ω
i
, containing j. The remaining terms, where j \notin Ω
r
i
 for r \neq r
j
, are all zero since p(j|Ω
r
i
,i) = 0.

Furthermore, assume Ω
i
 is a sufficient statistic, i.e., j and i are independent given Ω
i
, we obtain

\[ p(j|i) = p(j|Ω
i
,i)p(Ω
i
|i). \]  

(5)

Given the partition \{Ω
r
i
\}_{r=0}^{n_i}, the attention form in (3) can be rewritten as

\[ A(x_i) = \mathbb{E}_{p(j|i)}[v_j] = \sum_{r=0}^{n_i} \sum_{j \in Ω
r
i
} p(j, Ω
r
i
|i)v_j \]  

\[ = \sum_{j \in Ω
i
} \hat{p}(j|i)v_j + \sum_{r=1}^{n_i} p(Ω
r
i
|i) \left( \sum_{j \in Ω
r
i
} \hat{p}(j|i)v_j \right), \]  

(7)

where we consider direct attention in partition Ω
0
i
 and apply the local factorization (5) to the partition r = 1, . . . , n
i
. Here \hat{p}(j|i) \propto p(j|i) but with different normalization constants, which will be explained below. We refer to this model as Combiner since the structured attention (7) combines the direct expectation of Ω
0
i
 and multiple local expectations p(j|Ω
r
i
) and p(Ω
r
i
|i) to form the final conditional expectation.

Equivalently, we can also rewrite the structured attention (7) as

\[ A(x_i) = \sum_{j \in Ω
i
} \left[ \mathbb{I}(j \in Ω
0
i
)\hat{p}(j|i) + \sum_{r=1}^{n_i} \mathbb{I}(j \in Ω
r
i
)p(j|Ω
r
i
,i)p(Ω
r
i
|i)v_j \right], \]  

(8)

where \mathbb{I}() is a binary indicator function. After reordering, one can see from (8) that we obtain the effective conditional probability \hat{p}(j|i) that tries to approximate the original p(j|i). Each probability term depends on both current location i and other location j, and the expectation is still obtained with respect to a valid conditional probability (non-negative and sums up to 1 over Ω
i
).

**Requirement for Sub-quadratic Cost.** We can immediately see the benefit of this formulation from the fact that the local expectation in (7) is independent of the position i. The full dependence is achieved via the multiplier p(Ω
r
i
|i) where j \in Ω
r
i
. If we can design the local factorization such that:

1. the order of number of terms in (7) for p(·|i), \forall i ∈ [L]: \sum_{r=1}^{L} (n
i
+ |Ω
0
i
|) is sub-quadratic; and
2. let \mathcal{U} = \{Ω
r
i
\}_{i \in [L], r \in [1,n_i]} be the unique set of partitions used for local expectation calculation, then the order of |\mathcal{U}| (i.e., the number of unique partitions in \mathcal{U}) is sub-quadratic;
3. the order of total number of unique calculations of local expectations across all locations in (7), \sum_{Ω \in \mathcal{U}} |Ω| is sub-quadratic;

then one can see that the overall computation and memory cost will be sub-quadratic with full attention support Ω
Divider
 i = Ω
i
, \forall i ∈ [L]. We will discuss in detail in Section 4 how to instantiate such a principle by drawing inspiration from existing sparse transformers, and how to convert them into a full attention model almost for free with identical asymptotic complexity.

**Remark (Further Hierarchical Decomposition):** We introduce the local decomposition with a one layer partition of support of p(·|i) for simplicity. In fact, such local decompositions can be stacked further, which introduces a partition tree. Specifically, we can further partition Ω
i
 with disjoint subsets \{Ω
r
k
i
\}_k=1^{n_r}, and consider local decomposition p(j, Ω
r
k
i
|i) = p(j|Ω
r
k
i
,i)p(Ω
r
k
i
|Ω
r
k
i
,i)p(Ω
r
k
i
|i), where k
j
 is the index of sub-region which j belongs to. Thus, we obtain a hierarchical decomposition of p(j|i), which can also be plugged to (6) and yield a new full attention formulation.
3.2 Parameterizing Conditional Probabilities

While we obtained a possible way to speed up the standard Transformer via a combination of direct expectation and local expectations, it is also important to have an efficient design choice for the probability terms in (7), namely \( p(j|i) \) from direct expectation, \( p(j|\Omega'_i) \) from local expectation and \( p(r | \Omega'_i) \) for \( r \in [1, n_i] \). For simplicity we use the scaled dot-product, which means that we will associate positions \( i, j \) and variable sets \( \Omega'_i \) with the corresponding embedding representation, and thus the probability is proportional to the exponential of the embedding inner products. Specifically:

- \( \tilde{p}(j|i) \): As this term is for the direct expectation, we can let \( \tilde{p}(j|i) \propto \exp\left(\frac{q_j^\top k_j^i}{\sqrt{d}}\right) \), which is the same as vanilla attention (3) but with different normalizations, which will be explained in Equation 9.
- \( p(\Omega'_i|i) \): This term aims to capture the joint event probability, i.e., \( p(\Omega'_i|i) \propto \exp\left(\frac{q_i^\top k_{\Omega'_i}}{\sqrt{d}}\right) \). Thus the design choice of \( k_{\Omega'_i} \) should make an abstraction of the corresponding support \( \Omega'_i \). We find \( k_{\Omega'_i} = \max_{j \in \Omega'_i} k_j \) already provides good empirical results without introducing additional parameters; we can also use DeepSets [32] to obtain such abstraction.
- \( p(j|\Omega'_i) \): This term is the probability of getting \( j \) within this local span \( \Omega'_i \). We make \( p(j|\Omega'_i) \propto \exp\left(\frac{q_j k_j^i}{\sqrt{d}}\right) \), where we use max pooling or DeepSets over \( \{q_j\}_{j \in \Omega'_i} \) to obtain \( q_{\Omega'_i} \) similarly.

**Normalizing Probability Terms.** The terms in each local expectation \( p(j|\Omega'_i), \forall j \in \Omega'_i \) can be normalized within the local span; the direct expectation \( \tilde{p}(j|i) \) and the terms in \( p(\Omega'_i|i) \) should be normalized together,

\[
Z(x_i) = \sum_{j \in \Omega'_i} \exp\left(\frac{q_j^\top k_j^i}{\sqrt{d}}\right) + \sum_{r=1}^{n_i} \exp\left(\frac{q_i^\top k_{\Omega'_i}^r}{\sqrt{d}}\right),
\]

and \( Z(x_i) \) is the normalizing constant when calculating \( \tilde{p}(j|i) \) and \( p(\Omega'_i|i) \).

3.3 Trade-offs in Combiner

Combiner achieves full attention with reduced cost without making explicit sparsity or low-rank assumptions over the attention matrix. However this efficiency gain is not free. In this section we discuss the limitations of the simplification made by Combiner, and provide a simple workaround.

**Structured Attention Approximation.** We obtain the local decomposition (5) under the conditional independence assumption. Therefore, the local expectation in (7) is independent of the position \( i \), this suggests that any two locations \( i_1 \) and \( i_2 \) with \( \Omega'_i = \Omega'_j = \Omega \) would have linearly dependent attention scores over the region \( \Omega \). Formally, the probabilities formed by the effective conditional distribution \( \tilde{a}(\Omega)_{i_1} = [q(j_1|i_1), q(j_2|i_1), \ldots, q(j_{n_1} | \Omega'_1|i_1)] = \frac{p(i_1)}{p(i_1|i_2)} \tilde{a}(\Omega)_{i_2} \). In other words, the rank of the sub-matrix over the same partition in the resulting attention matrix is 1, therefore, the attention matrix is locally low-rank based on the partition. On the other hand, the direct expectation fully attends to each position in sub-support \( \Omega'_i \), which ensures the full-rank block. These two attention schemes make the attention matrix of Combiner structured. Compared with the low-rank approximation for each position \( i \) in sub-support \( \Omega'_i \), which is inspired from random features [29] in the kernel community, a structured approximation that exploits both the locally low-rank and full-rank blocks has proved more powerful theoretically and empirically in large-scale kernel machines [27].

**Improving Expressiveness Using a Mixture Model.** One way to further improve the expressiveness of the local factorization is to use a mixture model. This idea is adapted from the mixture of softmaxes [33] to obtain high-rank softmax layer in language modeling. Let \( \omega \) be a certain partition of the support (i.e., collection of \( \Omega'_i \) of \( \Omega \), then one can easily use \( A(x_i) = \frac{1}{M} \sum_{m=1}^{M} A(x_i; \omega_m) \) to compute the attention, where each component of the mixture \( A(x_i; \omega_m) \) is the term (7) using a specific factorization plan \( \omega_m \). Empirically we find two components are already sufficient to improve performance.

4 Combiner Instantiations

In this section we show several local factorization schemes satisfying the requirements in Section 3.1. As we will see, Combiner is able to convert several sparse transformers [14, 18, 20–22] into full
attention, with the same order of computation and memory consumption. One can also design other factorization patterns, which can be easily instantiated in Combiner.

4.1 Combiner-Fixed

The Sparse Transformer [14] is one of the most representative variants that can achieve $O(L\sqrt{L})$ computation and memory cost with sparse attention. Here we show how to convert this fixed pattern proposed in [14] (Figure 1(A)) into a factorization plan, and instantiate a full attention variant named the Combiner-Fixed (Figure 1(D)).

In the fixed-sparse attention, the support is $\Omega_i^{\text{sparse MLM}} = \{j : j \mod s = 0\} \cup \{j : j \equiv i \mod (\text{div} s)\}$, where $s$ is a hyper-parameter, div is integer division, and $j \equiv i \mod (\text{div} s)$ denotes that the quotients of $i$ and $j$ w.r.t. $s$ are the same. In the autoregressive case, $\Omega_i^{\text{sparse LM}} = \Omega_i^{\text{sparse MLM}} \cap [i]$. Please refer to Figure 1(A) for an illustration of the LM version.

Our design of $\omega_{\text{fixed}}^{\text{LM}}$ has the following form:

$$
\Omega_i^t = \{j : j \equiv i \mod (\text{div} s)\}, \quad \Omega_r^j = \{j : j \text{ div } s = r, j \not\in \Omega_0^t\}, \quad \forall r \in \left[\text{L div s}\right], \quad \forall i \in \left[\text{L}\right] \tag{10}
$$

where each local expectation is performed in each span of size $s$, and there are totally $L \text{ div s}$ spans across all locations. For each position $i \in \left[\text{L}\right]$, there are $(s + (L \text{ div s}))$ terms in (7); the local expectation has $(L \text{ div s})$ terms. The overall complexity is $O(L \cdot (s + 2(L \text{ div s})))$. The optimal $s$ is $O(\sqrt{L})$, and we can achieve $O(L \sqrt{L})$ computation and memory complexity, which is the same as [14] but here we gain full attention capability in each attention head. For the LM case, we can simply have $\omega_{\text{fixed}}^{\text{LM}} : \Omega_r^j \cap [i] \in \omega_{\text{fixed}}^{\text{LM}}$, which has the same $O(L \sqrt{L})$ optimal complexity.

4.2 Combiner-Logsparse

The Logsparse Transformer is proposed in [18] and can theoretically achieve $O(L \log L)$ cost. The general idea is to make the size of support $\Omega_i^{\text{Sparse}}$ no larger than $\lceil \log_2 i \rceil$. For the ease of notation, we first define bits$(n) = [b_1, b_2, \ldots, b_{\lceil \log_2 n \rceil}]$ to be the binary representation of integer $n$, with $b_t \in \{0, 1\}$ the coefficient of basis $2^t$. Thus we have $n = \sum_{t=1}^{\lceil \log_2 n \rceil} b_t \cdot 2^t$. One of the possible design choices to make Logsparse in the LM case is $\Omega_i^{\text{Sparse LM}} = \{\text{suff}_{t_i} := \sum_{t=1}^{\lceil \log_2 i-1 \rceil} b_t \cdot 2^t \} \cup \{i\}$, i.e., attend to the location indices that equal to the suffix sum of the weighted bits $(i-1)$, as well as location $i$ itself. This serves as our base sparse version as shown in Figure 1(B).

To exploit this scheme in the Combiner framework, we can define $[\log_2 n]$ non-overlapping supports, where $\Omega_i^t = \{\text{suff}_t\} \setminus \{\text{suff}_{t+1}\}$ with the boundary case $[\text{suff}_{\log_2 i-1}] = 0$. Note that for the ease of notation, some of the $\Omega_i^t$ are empty which will be ignored. In this case, the direct attention set $\Omega_i^t$ includes $\{i\}$, as well as $\{i-1\}$ when $i$ is an even number. Such a factorization leads to Combiner-Logsparse, as shown in Figure 1(E). From the Figure, we observe that in total we will have span summaries for every $2, 4, 8, \ldots, 2^{\lceil \log_2 L \rceil}$ locations, resulting in total $\sum_{t=1}^{\lceil \log_2 L \rceil} \frac{1}{2^t}$ or $O(L)$ summaries. Each location $i$ will select at most $O(\log(i))$ non-overlapping spans to cover the full support $\Omega_i$, and thus, the total cost will be $O(L \log L)$. We leave the design of MLM case to Appendix B.

4.3 Combiner-Axial

The Axial Transformer [20] builds the attention along each axis of the input data. Without loss of generality, we focus on 2D case where the input sequence is reshaped into a matrix of size $n \times m = L$. Specifically, the location $i$ in original sequence will be in $\text{row}_i = (i - 1) \text{ div } m + 1$ and $\text{col}_i = (i - 1) \mod m + 1$. We show how to simply enable full attention with factorization on 2D matrix, hence Combiner-Axial.

The sparse axial has $\Omega_i^{\text{Sparse LM}} = \{j : j - 1 \equiv i - 1 \mod m\} \cup \{j : j - 1 \equiv i - 1 \mod (\text{div} m)\}$, and $\Omega_i^{\text{Sparse LM}} = \Omega_i^{\text{Sparse MLM}} \cap [i]$, which all have at most $O(m + n)$ entries for each $i$, as illustrated in Figure 1(C). We propose several factorization schemes to make it an attention with full support.

- $\omega_{\text{axial-vertical}}^{\text{LM}} : \Omega_r^t = \Omega_i^{\text{Sparse LM}}$, and $\Omega_r^t = \{j : j \equiv r \mod m\} \cap [i - \text{col}_i], \text{for } r \in [m] \\setminus \{\text{col}_i\}$. As depicted in Figure 2(A), $\Omega_r^t$ corresponds to the column $r$ above $\text{row}_i$, where we use max pooling to
We evaluate Combiner with different full attention patterns on both autoregressive and bidirectional work for MLM. LM requires the following definition:

\[ TPUs \text{ (see illustration in Appendix D).} \]

variants (including the Routing Transformer) require a

gather

variants of Combiner-Axial. Blue and yellow correspond to direct and local attention respectively for location \(i\) (purple). Locations connected by arrows correspond to the same support \(\Omega^r\).

obtain the abstraction. To obtain such abstraction for all the locations, we can leverage the \text{cummax} operator for each column to efficiently obtain the prefix-max.

• \(\omega^{\text{LM}}_{\text{axial-horizontal}}\): similar as \(\omega_{\text{axial-vertical}}\) except that each \(\Omega^r_i\) summarizes the row \(r\) before \(row_i\) and excludes \(col_i\) (Figure 2(B)).

• \(\omega^{\text{LM}}_{\text{axial-rowmajor}}\): \(\Omega^0_i = \{j : j - 1 \equiv i - 1 (\text{div} m)\} \cap [i]\), \(i.e.,\) elements in the same row are directly attended, while \(\Omega^r_i = \{j : j \equiv r (\text{div} m)\} \cap [i - col_i]\) captures the rows before \(row_i\). This structure is similar to Combiner-Fixed, except for the way that the abstraction (and thus the local expectation) is computed. Combiner-Fixed computes the abstraction only based on \(r\) of partition \(\Omega^r_i\), where \(\omega_{\text{axial-rowmajor}}\) depends on both \(r\) and the column \(col_i\) (Figure 1(F)).

In all cases above, the cost is similar to the Axial Transformer [20], which is \(O(L \sqrt{L})\) if we reshape the sequence to a 2D matrix with \(n, m = O(\sqrt{L})\). We defer the MLM case to Appendix C.

4.4 Combiner-Learnable

Inspired by the Reformer [21] and Routing Transformer [22], we can also learn the factorization plan \(\omega\) from the data. We illustrate this with Routing Transformer and provide a way to enable full attention in Routing Transformer following the Combiner principle.

For a specific layer, suppose we have a learned disjoint region (or cluster in Routing Transformer) \(\{\Omega^r\}_{n_i=1}^{n_r}\), where \(\cup_r \Omega^r = [L]\). In Routing Transformer, we simply have \(\Omega^{\text{sparse MLM}} = \Omega^r\), where \(\Omega^r\) denotes the region where position \(i\) belongs to. To define the Combiner factorization, we let

\[
\omega^{\text{routing MLM}} : \Omega^0_i = \Omega^r_i, \quad \Omega^r_i = \Omega^r \setminus \Omega^0_i, \quad \forall r \in [n_i].
\]

(11)

Note that \(n_i = n\) (\(i.e.,\) number of learned clusters) for all locations. The above factorization can only work for MLM. LM requires the following definition:

\[
\omega^{\text{routing LM}} : \Omega^0_i = \Omega^r_i \cap [i], \quad \Omega^r_i = (\Omega^r \setminus \Omega^0_i) \cap [i], \quad \forall r \in [n_i].
\]

(12)

In general, both LM and MLM can have sub-quadratic cost when \(n = O(\sqrt{L})\). However, routing variants (including the Routing Transformer) require a \text{gather} operation, which can be slow on TPUs (see illustration in Appendix D).

5 Experimental Evaluation

We evaluate Combiner with different full attention patterns on both autoregressive and bidirectional sequence modeling tasks, covering a wide range of input data from images to texts. All tasks considered involve long sequences for up to 12,000 in length, some of which prevent the applicability of the vanilla transformer. We compare Combiner with state-of-the-art Transformers. We also perform a series of ablation studies where all of the models being compared use the \textit{exact same architecture} that only differ in the attention module, avoiding individual tricks employed in the original works (\textit{e.g.}, using both learnable and fixed patterns in Routing Transformer [22]). Details to reproducing all experimental results can be found in Appendix E.
Table 1: Ablation results in Bits per Dimension (Bits/Dim) on CIFAR-10 and ImageNet-64.

<table>
<thead>
<tr>
<th>Model</th>
<th>Layers</th>
<th>CIFAR-10</th>
<th>ImageNet-64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reformer [21]</td>
<td>6</td>
<td>-</td>
<td>3.740</td>
</tr>
<tr>
<td>Performer [28]</td>
<td>6</td>
<td>3.335</td>
<td>3.719</td>
</tr>
<tr>
<td>Logsparse [18]</td>
<td>6</td>
<td>4.253</td>
<td>4.351</td>
</tr>
<tr>
<td>Combiner-Logsparse (Ours)</td>
<td>6</td>
<td>3.366</td>
<td>3.795</td>
</tr>
<tr>
<td>Combiner-Fixed (Ours)</td>
<td>6</td>
<td>3.321</td>
<td>3.654</td>
</tr>
<tr>
<td>Combiner-Axial (Ours)</td>
<td>6</td>
<td>3.050</td>
<td><strong>3.585</strong></td>
</tr>
<tr>
<td>Combiner-Mixture (Ours)</td>
<td>6</td>
<td><strong>3.040</strong></td>
<td><strong>3.585</strong></td>
</tr>
<tr>
<td>Reformer [21]</td>
<td>12</td>
<td>-</td>
<td>3.710</td>
</tr>
<tr>
<td>Performer [28]</td>
<td>12</td>
<td>3.310</td>
<td>3.636</td>
</tr>
<tr>
<td>Routing Transformer [22]</td>
<td>12</td>
<td>2.950</td>
<td>-</td>
</tr>
<tr>
<td>Combiner-Mixture (Ours)</td>
<td>12</td>
<td><strong>2.885</strong></td>
<td><strong>3.504</strong></td>
</tr>
</tbody>
</table>

5.1 Autoregressive Sequence Modeling

In this subsection, we first perform density estimation on text and image using Combiner.

5.1.1 Language Modeling

For language modeling, we focus on the Wiki-40B-En dataset [34], which consists of clean Wikipedia pages in English. We use a sentence piece model with vocabulary size 32K to tokenize the text and measure the perplexity at the sentence piece level. To ensure fair comparison, all models being compared again have the same number of layers and hidden sizes, are are implemented under the same code base.

Table 2 shows the results of the comparison. As we can see, under 2k sequence length, Combiner variants are consistently better than their corresponding baselines, and are very close to the standard Transformer. When sequence length goes to 8k, the standard Transformer runs out of memory, whereas Combiner continues to achieve improved perplexity, surpassing the result of Transformer-2k. If we further use DeepSets to calculate the summarization terms $\theta_{ji}$ and $k_{ji}$, we may further achieve lower perplexity as shown in Table 3.

Table 2: LM Perplexity on Wiki-40B (Main).

<table>
<thead>
<tr>
<th>Model</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer-2k [1]</td>
<td>17.26</td>
</tr>
<tr>
<td>Performer-2k [28]</td>
<td>19.66</td>
</tr>
<tr>
<td>Routing-2k [22]</td>
<td>20.85</td>
</tr>
<tr>
<td>Fixed-2k [14]</td>
<td>18.04</td>
</tr>
<tr>
<td>Combiner-Fixed-2k (Ours)</td>
<td>17.70</td>
</tr>
<tr>
<td>Axial-2k [20]</td>
<td>20.82</td>
</tr>
<tr>
<td>Combiner-Axial-2k (Ours)</td>
<td>17.56</td>
</tr>
<tr>
<td>Combiner-Fixed-8k (Ours)</td>
<td>16.60</td>
</tr>
<tr>
<td>Combiner-Axial-8k (Ours)</td>
<td>16.49</td>
</tr>
</tbody>
</table>

Table 3: LM Perplexity on Wiki-40B (Ablation).

<table>
<thead>
<tr>
<th>Model</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer-2k [1]</td>
<td>17.26</td>
</tr>
<tr>
<td>Combiner-DeepSets-Max-8k (Ours)</td>
<td>16.29</td>
</tr>
<tr>
<td>Combiner-DeepSets-Mean-8k (Ours)</td>
<td>16.48</td>
</tr>
<tr>
<td>Combiner-Max-8k (Ours)</td>
<td>16.60</td>
</tr>
<tr>
<td>Combiner-Mean-8k (Ours)</td>
<td>16.54</td>
</tr>
</tbody>
</table>

5.1.2 Image Generative Models

CIFAR-10. We first perform a sanity check where we compare sparse attention baselines against Combiner with full attention under the same architecture on the CIFAR-10 dataset. The sequence length is 3072. For all the methods, we use a same 6-layer transformer with 8 attention heads and 512 embedding dimensions. We train all models for 500k iterations using batch size 32 on TPU v2. As shown in Table 1, given the same model architecture, Combiner-X performs significantly better than the base model X under the bits per dimension (BPD) metric on the 10,000 test images. In particular, Combiner significantly decreases BPD by 0.887, 0.087, and 0.626 compared to the base models Logsparse, Fixed and Axial, respectively. Note that all of the Combiner variants achieve better performance than the best of the base models. This demonstrates the advantage of Combiner over the baselines given the same 6-layer architecture. We observe a similar trend under a 12-layer architecture.
Table 4: Bits per Dimension (Bits/Dim) on CIFAR-10 and ImageNet-64.

<table>
<thead>
<tr>
<th>CIFAR-10</th>
<th>Bits/Dim</th>
</tr>
</thead>
<tbody>
<tr>
<td>PixelCNN [15]</td>
<td>3.03</td>
</tr>
<tr>
<td>PixelCNN++ [36]</td>
<td>2.92</td>
</tr>
<tr>
<td>Image Transformer [16]</td>
<td>2.90</td>
</tr>
<tr>
<td>PixelSNAIL [37]</td>
<td>2.85</td>
</tr>
<tr>
<td>Sparse Transformer [14]</td>
<td>2.80</td>
</tr>
<tr>
<td>Combiner-Axial (ours)</td>
<td><strong>2.77</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ImageNet 64x64</th>
<th>Bits/Dim</th>
</tr>
</thead>
<tbody>
<tr>
<td>PixelCNN [15]</td>
<td>3.57</td>
</tr>
<tr>
<td>Parallel Multiscale [38]</td>
<td>3.70</td>
</tr>
<tr>
<td>Glow [39]</td>
<td>3.81</td>
</tr>
<tr>
<td>SPN [40]</td>
<td>3.52</td>
</tr>
<tr>
<td>Sparse Transformer [14]</td>
<td>3.44</td>
</tr>
<tr>
<td>Axial Transformer [20]</td>
<td>3.44</td>
</tr>
<tr>
<td>Routing Transformer [22]</td>
<td>3.43</td>
</tr>
<tr>
<td>Combiner-Axial (ours)</td>
<td><strong>3.42</strong></td>
</tr>
</tbody>
</table>

Following the 128-layer architecture in Child et al. [14], we apply Combiner-Axial and achieve state-of-the-art performance, 2.77 BPD on CIFAR-10, as listed in Table 4. We run all of the models in Table 4 without data augmentation [35].

**ImageNet-64.** We also evaluate performance under the autoregressive setting on ImageNet-64, where sequence length is 12,288. We first perform the same analysis as CIFAR-10 and compare Combiner-X with the baselines using the same model architecture. As shown in Table 1, Combiner consistently outperforms the baselines with the same attention pattern. We further apply Combiner-Axial to a 30-layer Transformer, which achieves state-of-the-art performance on density estimation on ImageNet-64, demonstrating the effectiveness of full attention achieved by Combiner.

5.2 Bidirectional Sequence Modeling

Besides autoregressive tasks, we also evaluate Combiner on a set of standard bidirectional tasks to show the general applicability of the method.

5.2.1 Long-Range Arena

Long-Range Arena (LRA) is a unified benchmark [31] for probing the capability of efficient transformers on handling long sequences. We evaluate our models on five tasks from LRA: ListOps, Text Classification, Retrieval, Image Classification and Pathfinder. All of the tasks are sequence-level multi-class classification. Please refer to the original LRA paper for more details.

Table 5: Experimental results on Long-Range Arena benchmark.

<table>
<thead>
<tr>
<th>Model</th>
<th>ListOps</th>
<th>Text</th>
<th>Retrieval</th>
<th>Image</th>
<th>Pathfinder</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chance</td>
<td>10.00</td>
<td>50.00</td>
<td>50.00</td>
<td>10.00</td>
<td>50.00</td>
<td>34.00</td>
</tr>
<tr>
<td>Transformer</td>
<td>36.38</td>
<td>64.27</td>
<td>57.46</td>
<td>42.44</td>
<td>88.81</td>
<td>57.87</td>
</tr>
<tr>
<td>Local Attention</td>
<td>15.95</td>
<td>52.98</td>
<td>53.39</td>
<td>41.46</td>
<td>84.64</td>
<td>49.68</td>
</tr>
<tr>
<td>Sparse Tran.</td>
<td>35.78</td>
<td>63.58</td>
<td>59.59</td>
<td>44.24</td>
<td>83.90</td>
<td>57.42</td>
</tr>
<tr>
<td>Longformer</td>
<td>36.03</td>
<td>62.85</td>
<td>56.89</td>
<td>42.22</td>
<td>86.68</td>
<td>56.93</td>
</tr>
<tr>
<td>Linformer</td>
<td>35.49</td>
<td>53.94</td>
<td>52.27</td>
<td>38.56</td>
<td>86.17</td>
<td>53.28</td>
</tr>
<tr>
<td>Reformer</td>
<td>36.30</td>
<td>56.10</td>
<td>53.40</td>
<td>38.07</td>
<td>79.18</td>
<td>52.61</td>
</tr>
<tr>
<td>Sinkhorn Trans.</td>
<td>34.20</td>
<td>61.20</td>
<td>53.83</td>
<td>41.23</td>
<td>73.36</td>
<td>52.76</td>
</tr>
<tr>
<td>Synthesizer</td>
<td>36.50</td>
<td>61.68</td>
<td>54.67</td>
<td>41.61</td>
<td>81.61</td>
<td>55.21</td>
</tr>
<tr>
<td>BigBird</td>
<td>37.08</td>
<td>64.02</td>
<td>59.29</td>
<td>40.83</td>
<td>86.75</td>
<td>57.59</td>
</tr>
<tr>
<td>Linear Trans.</td>
<td>17.15</td>
<td>65.90</td>
<td>53.09</td>
<td>42.34</td>
<td>88.13</td>
<td>53.32</td>
</tr>
<tr>
<td>Performer</td>
<td>36.00</td>
<td>65.40</td>
<td>53.82</td>
<td>42.77</td>
<td>88.76</td>
<td>57.35</td>
</tr>
<tr>
<td>Combiner-Fixed</td>
<td>36.65</td>
<td>64.99</td>
<td>59.81</td>
<td>41.67</td>
<td>88.59</td>
<td>58.34</td>
</tr>
<tr>
<td>Combiner-Axial</td>
<td>36.15</td>
<td>64.36</td>
<td>56.10</td>
<td>41.33</td>
<td>88.43</td>
<td>57.27</td>
</tr>
</tbody>
</table>

As shown in Table 5, Combiner is able to match the performance of vanilla Transformer and achieves even better performance in some tasks. Following the protocol of LRA, all methods use the same architecture and hyperparameters for a controllable comparison. We use the numbers from Tay et al. [31] for all tasks except for Pathfinder. Since we were unable to reproduce the original Pathfinder results using the default setup in LRA Github repository, we rerun all the baselines using Pathfinder-inter configuration to conduct fair comparison. However, as the benchmark is still of small-scale and the LRA official website discourages hyperparameter tuning, Table 5 should be treated as results for the test bench of expressiveness compared to vanilla Transformer.
Table 6: MLM perplexity on C4 dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Perplexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformer-2k [1]</td>
<td>4.552</td>
</tr>
<tr>
<td>BigBird-2k [41]</td>
<td>4.696</td>
</tr>
<tr>
<td>Performer-2k [28]</td>
<td>10.940</td>
</tr>
<tr>
<td>Fixed-2k [14]</td>
<td>5.279</td>
</tr>
<tr>
<td>Combiner-Fixed-2k (Ours)</td>
<td>5.170</td>
</tr>
<tr>
<td>Axial-2k [20]</td>
<td>5.370</td>
</tr>
<tr>
<td>Combiner-Axial-2k (Ours)</td>
<td>4.809</td>
</tr>
<tr>
<td>Routing-2k [22]</td>
<td>6.703</td>
</tr>
<tr>
<td>Combiner-Routing-2k (Ours)</td>
<td>6.539</td>
</tr>
<tr>
<td>BigBird-8k [41]</td>
<td>4.542</td>
</tr>
<tr>
<td>Combiner-Axial-8k (Ours)</td>
<td>4.190</td>
</tr>
<tr>
<td>Combiner-Fixed-8k (Ours)</td>
<td>4.139</td>
</tr>
</tbody>
</table>

5.2.2 Masked Language Modeling

As the core element of BERT language pretraining [5], masked language modeling (MLM) refers to the task of reconstructing tokens that are randomly masked out in the input sequence. As with the LM task, we use perplexity as the main metric, which correlates relatively well with downstream task performance. Specifically, we use the large scale C4 dataset [8] for training and evaluation, and consider different sequence lengths. Following the original BERT setup, we mask out 15% of the tokens in each input sequence. The comparison is summarized in Table 6. Similar to the LM result, different Combiner variants consistently outperform their corresponding baselines under 2k sequence length. However, apart from the standard Transformer, Combiner-2k also falls behind BigBird-2k. We conjecture that this is related to the special design in BigBird such as all tokens can always attend to the <cls> token directly, which is only applicable in non-causal problems. That said, when we further increase sequence length to 8k, the standard Transformer runs into OOM issue, whereas Combiner not only outperforms BigBird but also substantially surpasses Transformer-2k. This suggests that Combiner can truly benefit from scaling learning to longer sequence lengths.

5.3 Runtime and Memory Usage of Combiner

Here we evaluate the inference runtime and memory usage of five baselines – Transformer, Performer, BigBird, Sparse-Fixed and Sparse-Axial, as well as three variants of Combiner– Combiner-Fixed, Combiner-Axial and Combiner-Mixture. We run inference of all the models on a TPU v3-16 (16 cores x 16GB) with batch size 16, and we test sequences of length from $2^{10}$ to $2^{14}$. As shown in Figure 3, Combiner instantiations achieve comparable runtime and memory usage with their sparse counterpart and Performer. Note Combiner achieves much better empirical performance than the sparse models and Performer. Combiner-Mixture has the same asymptotic complexity with Combiner-Fixed and Combiner-Axial, however, since it requires running two partition plans, it is slower than Combiner-Fixed and Combiner-Axial. Due to the gather operation required by the random attention which is not very TPU/GPU friendly, BigBird is very computationally expensive. And the Transformer model quickly runs out of memory when sequence length increases.

6 Conclusion

Inspired by the conditional expectation view of attention mechanism, we propose Combiner, a drop-in replacement of the attention module. By introducing structured decomposition to the conditional probability, Combiner achieves full attention capability while maintaining sub-quadratic computational and memory cost. We instantiate several Combiner variants converting existing sparse transformers to full attention. Combiner achieves state-of-the-art performance on both autoregressive and bidirectional tasks for image and text modeling, showing benefits in both modeling effectiveness and runtime efficiency. Future work includes additional factorization pattern designs, as well as applications of Combiner in domains like bioinformatics and speech.
Acknowledgments and Disclosure of Funding

We would like to thank Richard Song and David Dohan for the help on introducing Performer codebase and experiment configurations, Yi Tay and Mostafa Dehghani for clarifications on the LRA benchmark, James Lee-Thorp, Joshua Ainslie, and Ilya Eckstein for clarification on their LRA experiment results, Adams Yu for performing internal paper review and helpful suggestions. We also gratefully acknowledge the support of DARPA under Nos. HR00112190039 (TAMI), N660011924033 (MCS); ARO under Nos. W911NF-16-1-0342 (MURI), W911NF-16-1-0171 (DURIP); NSF under Nos. OAC-1835598 (CINES), OAC-1934578 (HDR), CCF-1918940 (Expeditions), IIS-2030477 (RAPID), NIH under No. R56LM013365; Stanford Data Science Initiative, Wu Tsai Neurosciences Institute, Chan Zuckerberg Biohub, Amazon, JPMorgan Chase, Docomo, Hitachi, Intel, JD.com, KDDI, NVIDIA, Dell, Toshiba, Visa, and UnitedHealth Group. Hongyu Ren is supported by the Masason Foundation Fellowship and the Apple PhD Fellowship. Jure Leskovec is a Chan Zuckerberg Biohub investigator.

References


