arXiv:1812.01243v8 [cs.CV] 13 Jan 2020

Efficient Attention: Attention with Linear Complexities

Shen Zhuoran[†*]
Independent Researcher
4244 University Way NE #85406, Seattle, WA 98105, United States
cmsflash99@gmail.com

Zhang Mingyuan[†], Zhao Haiyu, Yi Shuai
SenseTime International
71 Robinson Road #14-128, Singapore
zhangmingyuan,zhaohaiyu,yishuai@sensetime.com

Li Hongsheng
The Chinese University of Hong Kong
Sha Tin, Hong Kong
hsli@ee.cuhk.edu.hk

Abstract

The attention mechanism has seen wide applications in computer vision and natural language processing. Recent works developed the dot-product attention mechanism and applied it to various vision and language tasks. However, the memory and computational costs of dot-product attention grows quadratically with the spatiotemporal size of the input. Such growth prohibits the application of the mechanism on large inputs, e.g.*, long sequences, high-resolution images, or large videos. To remedy this drawback, this paper proposes a novel efficient attention mechanism, which is equivalent to dot-product attention but has substantially less memory and computational costs. The resource efficiency allows more widespread and flexible incorporation of efficient attention modules into a neural network, which leads to improved accuracies. Empirical evaluations on object recognition and image classification demonstrated the effectiveness of its advantages. Models with efficient attention achieved state-of-the-art performance on MS-COCO 2017 and significant improvement on ImageNet. Further, the resource efficiency of the mechanism democratizes attention to complicated models, which were unable to incorporate original dot-product attention due to prohibitively high costs. As an exemplar, an efficient attention-augmented model achieved state-of-the-art accuracies for stereo depth estimation on the Scene Flow dataset. Code is available at* https://github.com/cmsflash/efficient-attention.

1. Introduction

Attention is a mechanism in neural networks that focuses on long-range dependency modeling, a key challenge to deep learning that convolution and recurrence struggle to solve. A recent series of works developed the highly successful dot-product attention mechanism, which facilitates easy integration into a deep neural network. The mechanism computes the response at every position as a weighted sum of features at all positions in the previous layer. In contrast to the limited spatial and temporal receptive fields of convolution and recurrence, dot-product attention expands the receptive field to the entire input in one pass. Using dot-product attention to efficiently model long-range dependencies allows convolution and recurrence to focus on local dependency modeling, in which they specialize. Dot-product attention-based models now hold state-of-the-art records on nearly all tasks in natural language processing [25, 20, 6, 21]. The non-local module [26], an adaptation of dot-product attention for computer vision, achieved state-of-the-art performance on video classification [26] and generative adversarial image modeling [29, 3] and demonstrated significant improvements on object detection [26], instance segmentation [26], person re-identification [14], and image de-raining [12], *etc*.

However, global dependency modeling on large inputs, *e.g*. long sequences, high-definition images, and large videos, remains an unsolved problem. The quadratic[1] memory and computational complexities with respect to the input size of existing dot-product attention modules inhibit their application on such large inputs. For instance, a non-local module uses over 1 GB of GPU memory and over 50

[*] Work during internship at SenseTime.
[†] Equal contributions.

[1] The complexities are quadratic with respect to the spatiotemporal size of the input, which is quartically w.r.t. the side length of an input image or sextically w.r.t. the dimension of an input video.

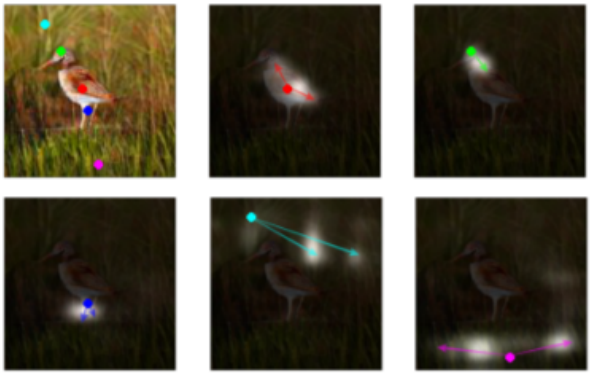

Figure 1. **An illustration of the learned attention maps in a non-local module.** The first image identifies five query positions with colored dots. Each of the subsequent images illustrates the attention map for one of the positions. Adapted from [29].

GMACC[2] of computation for a 64-channel 128×128 feature map or over 68 GB and over 3.2 TMACC for a 64-channel $64 \times 64 \times 32$ video feature volume. The high memory and computational costs constrain the application of dot-product attention to the low-resolution or short-temporal-span parts of models [26, 29, 3] and prohibits its use for resolution-sensitive or resource-hungry tasks.

The need for global dependency modeling on large inputs greatly motivates the exploration for a resource-efficient attention algorithm. An investigation into the non-local module revealed an intriguing phenomenon. The attention maps at each position, despite generated independently, are correlated. As [26] and [29] analyzed, the attention map of a position mainly focuses on semantically related regions. Figure 1 shows the learned attention maps in a non-local module. When generating an image of a bird before a bush, pixels on the legs tend to attend to other leg pixels for structural consistency. Similarly, body pixels mainly attend to the body, and background pixels focus on the bush.

This observation inspired the design of the efficient attention mechanism that this paper proposes. Similar to dot-product attention, the mechanism first generates a key feature map, a query feature map, and a value feature map from an input. It interprets each channel of the key feature map as a global attention map. Using each global attention map as the weights, efficient attention aggregates the value feature map to produce a global context vector that summarizes an aspect of the global features. Then, at each position, the module regards the query feature as a set of coefficients over the global context vectors. Finally, the module computes a sum of the global context vectors with the query feature as the weights to produce the output feature at the position. This algorithm avoids the generation of the pairwise attention matrix, whose size is quadratic in the spatiotemporal size of the input. Therefore, it achieves linear complexities with respect to input size and obtains significant efficiency improvements.

[2]MACC stands for multiply-accumulation. 1 MACC means 1 multiplication or addition operation.

The principal contribution of this paper is the efficient attention mechanism, which:

1. has linear memory and computational complexities with respect to the spatiotemporal size of the input;

2. possesses the same representational power as the widely adopted dot-product attention mechanism;

3. allows the incorporation of significantly more attention modules into a neural network, which brings substantial performance boosts to tasks such as object detection and instance segmentation (on MS-COCO 2017) and image classification (on ImageNet); and

4. facilitates the application of attention on resource-hungry tasks, such as stereo depth estimation (on the Scene Flow dataset).

2. Related Works

2.1. Dot-Product Attention

[1] proposed the initial formulation of the dot-product attention mechanism to improve word alignment in machine translation. Successively, [25] proposed to completely replace recurrence with attention and named the resultant architecture the Transformer. The Transformer architecture is highly successful on sequence tasks. They hold the state-of-the-art records on virtually all tasks in natural language processing [6, 21, 28] and is highly competitive on end-to-end speech recognition [7, 19]. [26] first adapted dot-product attention for computer vision and proposed the non-local module. They achieved state-of-the-art performance on video classification and demonstrated significant improvements on object detection, instance segmentation, and pose estimation. Subsequent works applied it to various fields in computer vision, including image restoration [16], video person re-identification [14], and notably generative adversarial image modeling, where SAGAN [29] and BigGAN [3] substantially advanced the state-of-the-art using the non-local module.

Efficient attention mainly builds upon the version of dot-product attention in the non-local module. Following [26], the team conducted most experiments on object detection and instance segmentation. The paper compares the resource efficiency of the efficient attention module against the non-local module under the same performance and their performance under the same resource constraints.

2.2. Scaling Attention

Besides dot-product attention, there are a separate set of techniques the literature refers to as attention. This section refers to them as scaling attention. While dot-product attention is effective for global dependency modeling, scaling attention focuses on emphasizing important features and suppressing uninformative ones. For example, the squeeze-and-excitation (SE) module [11] uses global average pooling and a linear layer to compute a scaling factor for each channel and then scales the channels accordingly. SE-enhanced models achieved state-of-the-art performance on image classification and substantial improvements on scene segmentation and object detection. On top of SE, CBAM [27] added global max pooling beside global average pooling and an extra spatial attention submodule. It further improved SE's performance.

Despite both names containing attention, dot-product attention and scaling attention are two completely separate sets of techniques with very different goals. When appropriate, one might take both techniques and let them work in conjunction. Therefore, it is unnecessary to make any comparison of efficient attention with scaling attention techniques.

2.3. Object Detection

Since the introduction of R-CNN [8], deep learning methods have achieved remarkable progress on object detection. Mask R-CNN [9] is one of the most successful deep detector recently. It is efficient and accurate and supports instance segmentation in addition to object detection. Because of these advantages and that [26] used Mask R-CNN as their baseline, this work also selected Mask R-CNN as the baseline.

Feature pyramid [15] is a module specifically for object detection that aggregates features across scales and network depths to improve feature expressiveness. It is lightweight but highly effective. Therefore, this work incorporated it into the baseline model.

MegDet [18], the winning entry of the MS-COCO Detection Challenge 2017, reported that large batch sizes and cross-GPU synchronous batch normalization is significantly beneficial to the performance of a deep detector. The baseline incorporated these two techniques.

3. Method

This section introduces the efficient attention mechanism. It is mathematically equivalent with the widely adopted dot-product attention mechanism in computer vision (*i.e.*, the attention mechanism in the Transformer [25] and the non-local module [26]). However, efficient attention has linear memory and computational complexities with respect to the number of pixels or words (hereafter referred to as positions).

Section 3.1 reviews the dot-product attention mechanism and identifies its critical drawback on large inputs to motivate efficient attention. The introduction of the efficient attention mechanism is in Section 3.2. Section 3.3 shows the equivalence between dot-product and efficient attention. Section 3.4 discusses the interpretation of the mechanism. Section 3.5 analyzes its efficiency advantage over dot-product attention.

3.1. A Revisit of Dot-Product Attention

Modeling long-range dependencies has been a central challenge for natural language processing and computer vision. [1] initially proposed dot-product attention for machine translation. Subsequently, the Transformer [25] adopted the mechanism to model long-range temporal dependencies between words. [26] introduce dot-product attention for the modeling of long-range dependencies between pixels in image and video understanding.

For each input feature vector $\boldsymbol{x}_i \in \mathbb{R}^d$ that corresponds to the i-th position, dot-product attention first uses three linear layers to convert $\boldsymbol{x}_i$ into three feature vectors, *i.e.*, the query feature $\boldsymbol{q}_i \in \mathbb{R}^{d_k}$, the key feature $\boldsymbol{k}_i \in \mathbb{R}^{d_k}$, and the value feature $\boldsymbol{v}_i \in \mathbb{R}^{d_v}$. The query and key features must have the same feature dimension d_k. One can measure the similarity between the i-th query and the j-th key as $\rho(\boldsymbol{q}_i^{\mathrm{T}} \boldsymbol{k}_j)$, where ρ is a normalization function. In general, the similarities are asymmetric, since the query and key features are the outputs of two separate layers. The dot-product attention module calculates the similarities between all pairs of positions. Using the similarities as weights, position i aggregates the value features from all positions via weighted summation to obtain its output feature.

If one represents all n positions' query, key, and value features in matrix forms as $\boldsymbol{Q} \in \mathbb{R}^{n \times d_k}$, $\boldsymbol{K} \in \mathbb{R}^{n \times d_k}$, $\boldsymbol{V} \in \mathbb{R}^{n \times d_v}$, respectively, the output of dot-product attention is

$$\boldsymbol{D}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \rho\left(\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}\right)\boldsymbol{V}. \tag{1}$$

The normalization function has two common choices:

$$\begin{aligned} \text{Scaling: } \rho(\boldsymbol{Y}) &= \frac{\boldsymbol{Y}}{n}, \\ \text{Softmax: } \rho(\boldsymbol{Y}) &= \sigma_{\text{row}}(\boldsymbol{Y}), \end{aligned} \tag{2}$$

where σ_{row} denotes applying the softmax function along each row of matrix $\boldsymbol{Y}$. An illustration of the dot-product attention module is in Figure 2 (left).

The critical drawback of this mechanism is its resource demands. Since it computes a similarity between each pair of positions, there are n^2 such similarities, which results in $O(n^2)$ memory complexity and $O(d_k n^2)$ computational complexity. Therefore, dot-product attention's resource demands get prohibitively high on large inputs. In practice, application of the mechanism is only possible on low-resolution features.

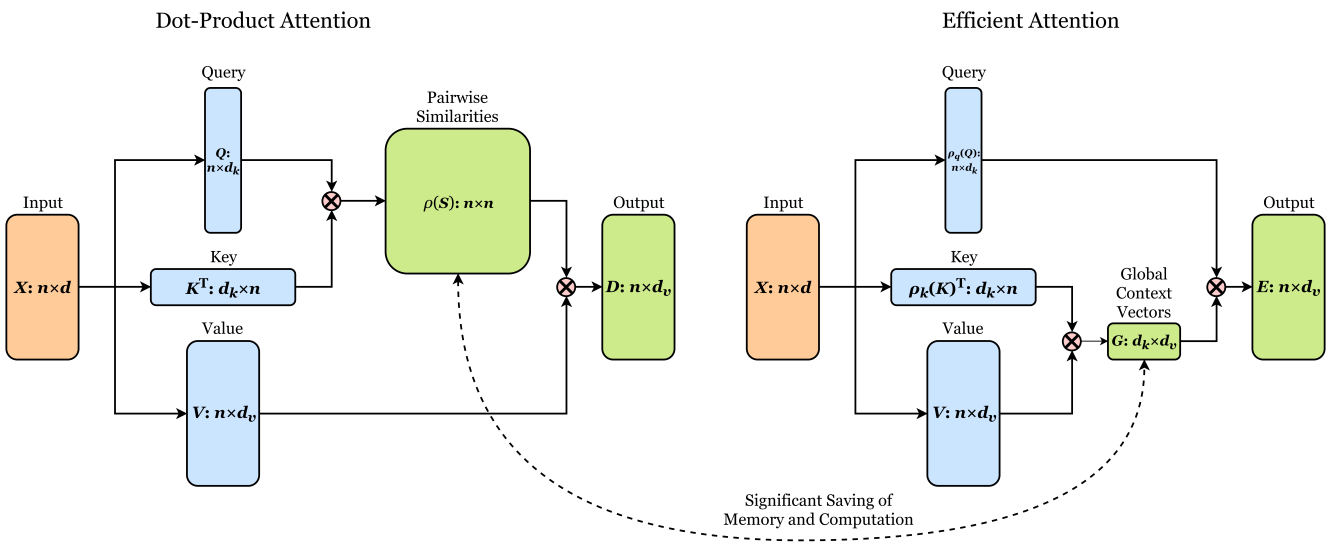

Figure 2. **Illustration of the architecture of dot-product and efficient attention.** Each box represents an input, output, or intermediate matrix. Above each box is the name of the corresponding matrix. The variable name and the size of the matrix are inside each box. $\otimes$ denotes matrix multiplication.

3.2. Efficient Attention

Observing the critical drawback of dot-product attention, this paper proposes the efficient attention mechanism, which is mathematically equivalent to dot-product attention but much faster and more memory efficient. In efficient attention, the individual feature vectors $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ still pass through three linear layers to form the query features $\boldsymbol{Q} \in \mathbb{R}^{n \times d_k}$, key features $\boldsymbol{K} \in \mathbb{R}^{n \times d_k}$, and value features $\boldsymbol{V} \in \mathbb{R}^{n \times d_v}$. However, instead of interpreting the key features as n feature vectors in $\mathbb{R}^{d_k}$, the module regards them as d_k single-channel feature maps. Efficient attention uses each of these feature maps as a weighting over all positions and aggregates the value features from all positions through weighted summation to form a global context vector. The name reflects the fact that the vector does not correspond to a specific position, but is a global description of the input features.

The following equation characterizes the efficient attention mechanism:

$$\boldsymbol{E}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \rho_q(\boldsymbol{Q}) \left(\rho_k(\boldsymbol{K})^{\mathrm{T}} \boldsymbol{V} \right), \tag{3}$$

where ρ_q and ρ_k are normalization functions for the query and key features, respectively. The implementation of the same two normalization methods as for dot-production attention are

$$\begin{aligned} \text{Scaling: } \rho_q(\boldsymbol{Y}) &= \rho_k(\boldsymbol{Y}) = \frac{\boldsymbol{Y}}{\sqrt{n}}, \\ \text{Softmax: } \rho_q(\boldsymbol{Y}) &= \sigma_{\text{row}}(\boldsymbol{Y}), \\ \rho_k(\boldsymbol{Y}) &= \sigma_{\text{col}}(\boldsymbol{Y}), \end{aligned} \tag{4}$$

where $\sigma_{\text{row}}, \sigma_{\text{col}}$ denote applying the softmax function along each row or column of matrix $\boldsymbol{Y}$, respectively.

The efficient attention module is a concrete implementation of the mechanism for computer vision data. For an input feature map $x \in \mathbb{R}^{h \times w \times d}$, the module flattens it to a matrix $\boldsymbol{X} \in \mathbb{R}^{hw \times d}$, applies the efficient attention mechanism on it, and reshapes the result to $h \times w \times d_v$. If $d_v \neq d$, it further applies a 1x1 convolution to restore the dimensionality to d. Finally, it adds the resultant features to the input features to form a residual structure.

3.3. Equivalence between Dot-Product and Efficient Attention

Following is a formal proof of the equivalence between dot-product and efficient attention when using scaling normalization. Substituting the scaling normalization formula in Equation (2) into Equation (1) gives

$$\boldsymbol{D}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{n} \boldsymbol{V}. \tag{5}$$

Similarly, plugging the scaling normalization formulae in Equation (4) into Equation (3) results in

$$\boldsymbol{E}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \frac{\boldsymbol{Q}}{\sqrt{n}} \left(\frac{\boldsymbol{K}^{\mathrm{T}}}{\sqrt{n}} \boldsymbol{V} \right). \tag{6}$$

Since scalar multiplication is commutative with matrix multiplication and matrix multiplication is associative, we

have

$$
\begin{aligned}
\boldsymbol{E}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \frac{\boldsymbol{Q}}{\sqrt{n}} \left(\frac{\boldsymbol{K}^{\mathrm{T}}}{\sqrt{n}} \boldsymbol{V} \right) \\
&= \frac{1}{n} \boldsymbol{Q} \left(\boldsymbol{K}^{\mathrm{T}} \boldsymbol{V} \right) \\
&= \frac{1}{n} \left(\boldsymbol{Q} \boldsymbol{K}^{\mathrm{T}} \right) \boldsymbol{V} \\
&= \frac{\boldsymbol{Q} \boldsymbol{K}^{\mathrm{T}}}{n} \boldsymbol{V}.
\end{aligned} \tag{7}
$$

Comparing Equations (5) and (7), we get

$$
\boldsymbol{E}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{D}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}). \tag{8}
$$

Thus, the proof is complete.

The above proof works for the softmax normalization variant with one caveat. The two softmax operations on $\boldsymbol{Q}, \boldsymbol{K}$ are not exactly equivalent to the single softmax on $\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}$. However, they closely approximate the effect of the original softmax function. The critical property of $\sigma_{\text{row}}\left(\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}\right)$ is that each row of it sums up to 1 and represents a normalized attention distribution over all positions. The matrix $\sigma_{\text{row}}(\boldsymbol{Q})\sigma_{\text{col}}(\boldsymbol{K})^{\mathrm{T}}$ shares this property. Therefore, the softmax variant of efficient attention is a close approximate of that variant of dot-product attention. Section 4.1 demonstrates this claim empirically.

3.4. Interpretation of Efficient Attention

Efficient attention brings a new interpretation of the attention mechanism. In dot-product attention, selecting position i as the reference position, one can collect the similarities of all positions to position i and form an attention map $\boldsymbol{s}_i$ for that position. The attention map $\boldsymbol{s}_i$ represents the degree to which position i attends to each position j in the input. A higher value for position j on $\boldsymbol{s}_i$ means position i attends more to position j. In dot-product attention, every position i has such an attention map $\boldsymbol{s}_i$, which the mechanism uses to aggregate the value features $\boldsymbol{V}$ to produce the output at position i.

In contrast, efficient attention does not generate an attention map for each position. Instead, it interprets the key features $\boldsymbol{K} \in \mathbb{R}^{n \times d_k}$ as d_k attention maps $\boldsymbol{k}_j^{\mathrm{T}}$. Each $\boldsymbol{k}_j^{\mathrm{T}}$ is a global attention map that does not correspond to any specific position. Instead, each of them corresponds to a semantic aspect of the entire input. For example, one such attention map might cover the persons in the input. Another might correspond to the background. Section 4.3 gives several concrete examples. Efficient attention uses each $\boldsymbol{k}_j^{\mathrm{T}}$ to aggregate the value features $\boldsymbol{V}$ and produce a global context vector $\boldsymbol{g}_j$. Since $\boldsymbol{k}_j^{\mathrm{T}}$ describes a global, semantic aspect of the input, $\boldsymbol{g}_j$ also summarizes a global, semantic aspect of the input. Then, position i uses $\boldsymbol{q}_i$ as a set of coefficients over $\boldsymbol{g}_0, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_{d_k-1}$. Using the previous example, a person pixel might place a large weight on the global context vector for persons to refine its representation. A pixel at the boundary of an object might have large weights on the global context vectors for both the object and the background to enhance the contrast.

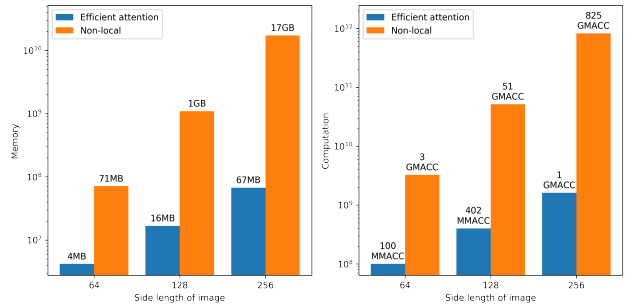

Figure 3. **Resource requirements under different input sizes.** The blue and orange bars depict the resource requirements of the efficient attention and non-local modules, respectively. The calculation assumes $d = d_v = 2d_k = 64$. The figure is in log scale.

3.5. Efficiency Advantage

This section analyzes the efficiency advantage of efficient attention over dot-product attention in memory and computation. The reason behind the efficiency advantage is that efficient attention does not compute a similarity between each pair of positions, which would occupy $O(n^2)$ memory and require $O(d_k n^2)$ computation to generate. Instead, it only generates d_k global context vectors in $\mathbb{R}^{d_v}$. This change eliminates the $O(n^2)$ terms from both the memory and computational complexities of the module. Consequently, efficient attention has $O((d_k + d)n + d_k d)$ memory complexity and $O((d_k d + d^2)n)$ computational complexities, assuming the common setting of $d_v = d$. Table 1 shows complexity formulae of the efficient attention module and the non-local module (using dot-product attention) in detail. In computer vision, this complexity difference is very significant. Firstly, n itself is quadratic in image side length and often very large in practice. Secondly, d_k is a parameter of the module, which the designer of a network can tune to meet different resource requirements. Section 4.2.3 shows that, within a reasonable range, this parameter has minimal impact on performance. This result means that an efficient attention module can typically have a small d_k, which further increases its efficiency advantage over dot-product attention. Table 2 compares the complexities of the efficient attention module with the ResBlock. The table shows that the resource demands of the efficient attention module are on par with (less than in most cases) the ResBlock, which gives an intuitive idea on the level of efficiency of he module.

The rest of this section will give several concrete examples comparing the resource demands of the efficient atten-

Metric	Efficient attention module	Non-local module
Memory (floats)	$(2d_k + 3d)n + d_k d$	$(2d_k + 3d)n + n^2$
Computation (MACC)	$(8d_k d + 2d^2 + d)n$	$(4d_k d + 2d^2 + d)n + (2d_k + 2d)n^2$
Memory complexity	$O((d_k + d)n + d_k d)$	$O((d_k + d)n + n^2)$
Computational complexity	$O((d_k d + d^2)n)$	$O((d_k d + d^2)n + (d_k + d)n^2)$

Table 1. **Comparison of resource usage of the efficient attention and non-local modules.** This table assumes that $d_v = d$, which is the setting for all experiments in Section 4 and also a common setting in the literature for dot-product attention.

Module	ResBlock	EA module
Memory (floats)	$5dn$	$4dn + \frac{d^2}{2}$
Computation (MACC)	$(36d^2 + 11d)n$	$(6d^2 + d)n$
Mem. complexity	$O(dn)$	$O(dn + d^2)$
Comp. complexity	$O(d^2 n)$	$O(d^2 n)$

Table 2. **Comparison of resource usage of the efficient attention module and the ResBlock.** Since the ResBlock does not have parameters d_k, d_v, this table sets $d_v = d, d_k = \frac{d}{2}$, the typical values for these parameters.

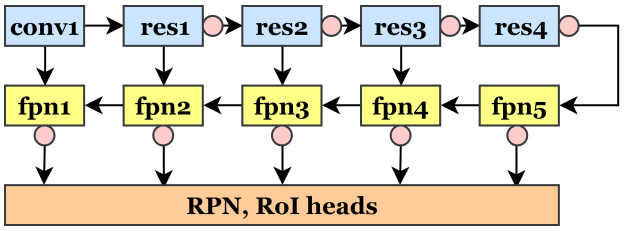

Figure 4. **Architecture of the Mask R-CNN FPN baseline.** Red dots are potential locations for the insertion of efficient attention or non-local modules.

tion and non-local modules. Figure 3 compares their resource consumption for image features with different sizes. Directly substituting the non-local module on the 64×64 feature map in SAGAN [29] yields a 17-time saving of memory and 32-time saving of computation. The gap widens rapidly with the increase of the input size. For a 256×256 feature map, a non-local module would require impractical amounts of memory (17.2 GB) and computation (412 GMACC). With the same input size, an efficient attention module uses 1/260 the amount of memory and 1/515 the amount of computation. The difference is more prominent for videos. Replacing the non-local module on the tiny $28 \times 28 \times 4$ feature volume in res3 of the non-local I3D-ResNet-50 network [26] results in 2-time memory and computational saving. On a larger $64 \times 64 \times 32$ feature volume, an efficient attention module requires 1/32 the amount of memory and 1/1025 the amount of computation.

4. Experimental Results

4.1. Comparison with the Non-Local Module and the State-of-the-Art

4.1.1 MS-COCO Task Suite

This section presents comparison experiments on the MS-COCO 2017 dataset for object detection and instance segmentation. The section first compares the efficient attention module with the non-local module, which uses dot-product attention. Then, it compares models using efficient attention with other state-of-the-art methods. The baseline is a ResNet-50 Mask R-CNN with a 5-level feature pyramid [15]. Figure 4 illustrates the architecture. The backbones initialize from ImageNet pretrainings. All other modules use random initialization. All models trained for 24 epochs on 32 NVIDIA GTX 1080 Ti GPUs. The batch size is 64. The learning rate is 1.25e-4 at the beginning of training and drops by a factor of 10 at the start of the 18th and 21st epochs. All experiments used softmax normalization and $d_k = 64$.

Layer(s)	EA module	Non-local	Input size
None	39.4/35.1	39.4/35.1	N/A
res3	40.2/36.0	40.3/35.9	56×80
fpn1	39.9/35.8	OOM	224×320
fpn2	39.7/35.7	OOM	112×160
fpn3	39.7/35.5	39.8/35.5	56×80
Best	**41.2/36.7**	40.3/35.9	N/A

Table 3. **Comparison between the efficient attention and non-local modules.** Each numeric cell is in the format box AP/mask AP. The best configuration for the efficient attention module inserted at FPN layers 1-5 and backbone layers res3,4. The best for the non-local module inserted at res3.

Table 3 reports the comparison against the non-local module. As rows res3 and fpn3 show, inserting an efficient attention module or a non-local module at the same location in a network has nearly identical effects on the performance. However, at layers where the input is large, inserting a non-local module leads to out-of-memory errors, while inserting an efficient attention module improved performance significantly and consumed little additional resources. Consequently, using the best insertion schemes for each module, the efficient attention module outperforms the non-local module by 0.9 box AP and 0.8 mask AP.

Table 4 compares efficient attention-augmented detectors with other state-of-the-art methods on the MS-COCO 2017 dataset. EA Mask R-CNN with a ResNeXt-101

Model	Backbone	AP
TDM Faster R-CNN [22]	Inc.-ResNet-v2	36.8
Mask R-CNN [9]	ResNeXt-101	39.8
Soft-NMS [2]	Algn.-Inc.-ResNet	40.9
LH R-CNN [13]	ResNet-101	41.5
Fitness NMS [24]	ResNet-101	41.8
EA Mask R-CNN	ResNet-101	43.1
EA Mask R-CNN	ResNeXt-101	**44.9**

Table 4. **Comparison with the state-of-the-art on MS-COCO 2017.** *Inc.-ResNet-v2* and *Algn.-Inc.-ResNet* represent Inception-ResNet-v2 and Aligned-Inception-ResNet, respectively.

Model	EPE
PSMNet (original)	1.09
PSMNet (baseline)	0.513
EA-PSMNet	**0.477**
Nonlocal-PSMNet	OOM

Table 5. **Experiments on the Scene Flow dataset.** EA-PSMNet is the proposed approach. *OOM* indicates out-of-memory errors.

Model	EPE
iResNet-i2 [17]	1.40
PSMNet [4]	1.09
EdgeStereo [23]	1.12
CSPN [5]	0.78
EA-PSMNet	**0.48**

Table 6. **Comparison with the state-of-the-art on the Scene Flow dataset.** *EPE* stands for end-point error and is lower the better. The table rounded EPE for EA-PSMNet to the second decimal place following the format of previous works.

backbone set a new state-of-the-art. The version with a ResNet-101 backbone also outperformed all other ResNet-101-based models.

4.1.2 Stereo Depth Estimation

This section will present the experimental results of efficient attention-augmented models for stereo depth estimation. The experiments used the Scene Flow dataset. The baseline is the state-of-the-art published model, PSMNet [4]. The experiments empirically determined an optimal set of hyperparamters, which includes a batch size of 24, a learning rate of 2e-3, and 100 training epochs. Other hyperparamters remain the same as in [4].

As Table 5 shows, the proposed EA-PSMNet demonstrated significant improvement over the highly competitive baseline, which already surpassed the former state-of-the-art by a substantial margin. Table 6 compares EA-PSMNet with other state-of-the-art methods. EA-PSMNet set a new state-of-the-art on the Scene Flow dataset.

Layer(s)	EA module	Nonlocal
Baseline	76.052/92.952	76.052/92.952
res_1	76.932/93.252	OOM
res_2	76.532/93.274	OOM
$\text{res}_{1,2}$	**77.312/93.650**	OOM

Table 7. **Experiments on ImageNet.** All results are in the form top-1/top-5. All experiments used single-crop testing.

Method	Box AP	Mask AP
None	NaN	NaN
Scaling	**40.2**	35.9
Softmax	**40.2**	**36.0**

Table 8. **Experiments for attention normalization methods**. All experiments inserted a single efficient attention module at res3.

4.1.3 Image Classification

This section presents the experimental results of efficient attention-augmented models on the ImageNet dataset for image classification. The baseline is ResNet-50 [10]. As in Table 7, insertions at res1 and res2 both lead to significant improvements. Inserting simultaneously at both layers resulted in the largest gain of performance. In comparison, experiments with the same settings but with the non-local module all encountered OOM errors.

4.2. Ablation Studies

4.2.1 Attention Normalization

These experiments empirically validated the necessity of attention normalization and compared the two methods Section 3.2 specified, namely scaling and softmax normalization. Table 8 reports the experimental outcomes. The results demonstrate that while attention normalization is critical for the training of efficient attention-augmented models, the effectiveness does not depend on the specific normalization technique. Following [26], all other experiments used softmax normalization.

4.2.2 Layer of Insertion

Table 9 shows experiments contrasting models with EA modules inserted at different layers. When adding a single module, the reader can infer a general trend that among the same type of layers (*e.g.*, among FPN levels or among ResNet levels), the higher the resolution at a layer, the more performance gain the insertion of an efficient attention module brings. This result reiterates the significance of efficient attention. Since attention is more beneficial on higher-resolution inputs, the efficiency advantage of efficient attention is highly impactful. The layer res4 is the only one where insertion lead to degraded performance. A hypothesis is that the low resolution and large gap between the

Layer(s)	Box AP	Mask AP	Input size
None	39.4	35.1	-
res3	**40.2**	**36.0**	56×80
res4	39.1	34.8	28×40
fpn1	39.9	35.8	224×320
fpn2	39.7	35.7	112×160
fpn3	39.7	35.5	56×80
fpn4	39.7	35.4	28×40
fpn5	39.6	35.3	14×20
fpn1-5	40.6	36.2	-
fpn1-5 & res3	40.8	36.3	-
fpn1-5 & res3,4	**41.2**	**36.7**	-

Table 9. **Comparison between insertions at different layers.**

dimensionality of the input (2048) and the keys (64) might be the cause. A defect in the implementation caused the experiments for res1 and res2 to fail, so the results are not available for report in this paper.

Subsequent experiments explored adding multiple efficient attention modules to a network. Adding an efficient attention module to res3, res4, and every FPN level resulted in an improvement of 1.8 box AP and 1.6 mask AP. The results back the hypothesis in Section 1 that a core advantage of attention is the ability to model multi-hop long-range dependencies.

Interestingly, although a single insertion at res4 decreased performance, removing the module at res4 from the fpn1-5 & res3,4 model caused a significant decrease in performance, as the row of fpn1-5 & res3 shows. This result further reinforces the importance of multi-hop dependency modeling, as even efficient attention modules which cannot improve performance independently can make contributions when collaborating with other efficient attention modules. The low complexities of the efficient attention module enable multiple insertion and realize this previously hypothetical advantage of attention.

4.2.3 Dimensionality of the Keys

These experiments tested the impact of the dimensionality of the keys on the effect of efficient attention. As in Table 10, decreasing the dimensionality of the keys from 128 to 32 caused minimal performance change. This result reinforces the hypothesis in Section 1 that most attention maps are expressible as linear combinations of a limited set of basis attention maps. Therefore, researchers can reduce the dimensionality of the keys and queries in efficient attention modules, if there is a need to further save resources.

d_k	Box AP	Mask AP
32	40.4	36.1
64	**40.6**	**36.2**
128	40.3	36.1

Table 10. **Experiments with different dimensionalities of the keys.** All experiments inserted efficient attention modules at FPN levels 1-5.

Backbone	Baseline APs	With EA modules
ResNet-50	39.4/35.1	41.2/36.7
ResNet-101	41.3/36.6	43.1/37.9
ResNeXt-101	43.5/38.5	44.9/39.5

Table 11. **Experiments with different backbone architectures.** All results have format box AP/mask AP. The *with EA modules* variants inserted the modules at fpn1-5 & res3,4. Experiments for ResNet-101 and ResNeXt-101 used batch sizes of 32, not 64.

Figure 5. **Visualization of global attention maps.** The left-most column displays 4 images from MS-COCO 2017. The other three columns show three of the corresponding global attention maps from the efficient attention module at FPN level 1 for each respective example.

4.2.4 Backbone Architecture

This section reports experiments with different backbone networks. Table 11 reports the results using three selected architectures, ResNet-50, ResNet-101 and ResNeXt-101. The significant and robust performance gains across all three backbones demonstrate the generalizability of efficient attention.

4.3. Visualization

Figure 5 shows visualization of the global attention maps for various examples from the efficient attention module at fpn1 in the model corresponding to the last row in Table 9. The figure illustrates 3 sets of global attention maps each with a distinct, semantic focus. Column 2 tends to capture the foreground, column 3 tends to capture the core parts

of objects, and column 4 tends to capture the peripheral of objects. The semantic distinctiveness of each set of global attention maps supports the analysis in Section 1 that the attention maps are linear combinations of a set of basis attention maps each focusing on a semantically significant area.

5. Conclusion

This paper has presented the efficient attention mechanism, an attention mechanism that is quadratically more memory- and computationally-efficient than the widely adopted dot-product attention mechanism. By dramatically reducing the resource usage, efficient attention enables a large number of new use cases of attention, particularly in domains with tight resource constraints or large inputs.

The experiments verified its effectiveness on four distinct tasks, object detection, instance segmentation, stereo depth estimation, and image classification. It brought significant improvement for each task. On object detection and stereo depth estimation, efficient attention-augmented models have set new states-of-the-art. Besides the tasks this paper evaluated efficient attention on, it has promising potential in other fields where attention has demonstrated effectiveness. These fields include generative adversarial image modeling [29, 3] and most tasks in natural language processing [25, 20, 6, 21]. Future plans include generalizing efficient attention to these fields, as well as other fields where the prohibitive costs have been preventing the application of attention.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR 2015*, 2015. 2, 3

[2] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms: Improving object detection with one line of code. In *ICCV 2017*, 2017. 7

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 1, 2, 9

[4] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *CVPR*, 2018. 7

[5] Xinjing Cheng, Peng Wang, and Ruigang Yang. Learning depth with convolutional spatial propagation network. *arXiv preprint arXiv:1810.02695*, 2018. 7

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1, 2, 9

[7] Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *ICASSP 2018*, 2018. 2

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR 2014*, 2014. 3

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV 2017*, 2017. 3, 7

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR 2016*, 2016. 7

[11] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR 2018*, 2018. 3

[12] Guanbin Li, Xiang He, Wei Zhang, Huiyou Chang, Le Dong, and Liang Lin. Non-locally enhanced encoder-decoder network for single image de-raining. In *ACMMM 2018*, 2018. 1

[13] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Light-head r-cnn: In defense of two-stage object detector. *arXiv preprint arXiv:1711.07264*, 2017. 7

[14] Xingyu Liao, Lingxiao He, and Zhouwang Yang. Video-based person re-identification via 3d convolutional networks and non-local attention. *arXiv preprint arXiv:1807.05073*, 2018. 1, 2

[15] Tsung-Yi Lin, Piotr Dollár, Ross B Girshick, Kaiming He, Bharath Hariharan, and Serge J Belongie. Feature pyramid networks for object detection. In *CVPR 2017*, 2017. 3, 6

[16] Ding Liu, Bihan Wen, Yuchen Fan, Chen Change Loy, and Thomas S Huang. Non-local recurrent network for image restoration. *arXiv preprint arXiv:1806.02919*, 2018. 2

[17] Jiahao Pang, Wenxiu Sun, Jimmy SJ Ren, Chengxi Yang, and Qiong Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *ICCV 2017 Workshops*, 2017. 7

[18] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. In *CVPR 2018*, 2018. 3

[19] Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Muller, and Alex Waibel. Very deep self-attention networks for end-to-end speech recognition. *arXiv preprint arXiv:1904.13377*, 2019. 2

[20] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Blog*, 2018. 1, 9

[21] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019. 1, 2, 9

[22] Abhinav Shrivastava, Rahul Sukthankar, Jitendra Malik, and Abhinav Gupta. Beyond skip connections: Top-down modulation for object detection. *arXiv preprint arXiv:1612.06851*, 2016. 7

[23] Xiao Song, Xu Zhao, Hanwen Hu, and Liangji Fang. Edgestereo: A context integrated residual pyramid network for stereo matching. In *ACCV 2018*, 2018. 7

[24] Lachlan Tychsen-Smith and Lars Petersson. Improving object localization with fitness nms and bounded iou loss. In *CVPR 2018*, 2018. 7

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS 2017*, 2017. 1, 2, 3, 9

[26] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR 2018*, 2018. 1, 2, 3, 6, 7

[27] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *ECCV 2018*, 2018. 3

[28] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019. 2

[29] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018. 1, 2, 6, 9