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
182 Cecil Street, #36-02 Frasers Tower, Singapore 069547
zhangmingyuan,zhaohaiyu,yishuai@sensetime.com

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

Abstract

Dot-product attention has wide applications in computer vision and natural language processing. However, its memory and computational costs grow quadratically with the input size. Such growth prohibits its application on high-resolution inputs. To remedy this drawback, this paper proposes a novel efficient attention mechanism equivalent to dot-product attention but with substantially less memory and computational costs. Its resource efficiency allows more widespread and flexible integration of attention modules into a network, which leads to better accuracies. Empirical evaluations demonstrated the effectiveness of its advantages. Efficient attention modules brought significant performance boosts to object detectors and instance segmenters on MS-COCO 2017. Further, the resource efficiency democratizes attention to complex models, where high costs prohibit the use of dot-product attention. As an exemplar, a model with efficient attention 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

Dot-product attention [1, 22, 23] is a prevalent mechanism in neural networks for long-range dependency modeling, a key challenge to deep learning that convolution and recurrence struggle to solve. 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 receptive fields of convolution or the recurrent layer, 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. The non-local module [23], an adaptation of dot-product attention for computer vision, achieved state-of-the-art performance on video classification [23] and generative adversarial image modeling [28, 2] and demonstrated significant improvements on object detection [23], instance segmentation [23], person re-identification [14], image de-raining [13], *etc*.

However, global dependency modeling on large inputs (*e.g*. long sequences, high-resolution images, large videos) remains an open problem. The quadratic[1] memory and computational complexities with respect to the input size of dot-product attention inhibits its application on large inputs. For instance, a non-local module uses over 1 GB of GPU memory and over 25 GMACC[2] of computation for a 64-channel 128×128 feature map or over 68 GB and over 1.6 TMACC for a 64-channel $64 \times 64 \times 32$ 3D feature vol-

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

[1]The complexities are quadratic with respect to the spatiotemporal size of the input, which is quartically w.r.t. the side length of a 2D feature map, or sextically w.r.t. the dimension of a 3D feature volume.

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

ume (*e.g*. for depth estimation or video tasks). The high memory and computational costs constrain the application of dot-product attention to the low-resolution parts of models [23, 28, 2] and prohibits its use for resolution-sensitive or resource-hungry tasks.

The need for global dependency modeling on large inputs motivates the exploration for a resource-efficient attention mechanism. An investigation into the non-local module revealed an intriguing discovery. As Figure 1 shows, putting aside the normalization, dot-product attention involves two consecutive matrix multiplications. The first one ($\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{K}^\mathsf{T}$) computes pairwise similarities between pixels and forms per-pixel attention maps. The second ($\boldsymbol{D} = \boldsymbol{S}\boldsymbol{V}$) aggregates the values $\boldsymbol{V}$ by the per-pixel attention maps to produce the output. Since matrix multiplication is associative, switching the order from $(\boldsymbol{Q}\boldsymbol{K}^\mathsf{T})\boldsymbol{V}$ to $\boldsymbol{Q}(\boldsymbol{K}^\mathsf{T}\boldsymbol{V})$ has no impact on the effect but changes the complexities from $O(n^2)$ to $O(d_k d_v)$, for n the input size and d_k, d_v the dimensionalities of the keys and the values, respectively. This change removes the $O(n^2)$ terms in the complexities of the module, making it linear in complexities. Further, $d_k d_v$ is significantly less than n^2 in practical cases, hence this new term will not become a new bottleneck. Therefore, switching the order of multiplication to $\boldsymbol{Q}(\boldsymbol{K}^\mathsf{T}\boldsymbol{V})$ results in a substantially more efficient mechanism, which this paper names *efficient attention*.

The new mechanism is mathematically equivalent to dot-product attention with scaling normalization and approximately equivalent with softmax normalization. Experiments empirically verified that when the equivalence is approximate, it does not impact accuracies. In addition, experiments showed that its efficiency allows the integration of more attention modules into a network and integration into high-resolution parts of a network, which lead to significantly higher accuracies. Further, experiments demonstrated that efficient attention democratizes attention to tasks where dot-product attention is inapplicable due to resource constraints.

Another discovery is that efficient attention brings a new interpretation to the attention mechanism. Assuming the keys are of dimensionality d_k and the input size is n, one can interpret the $d_k \times n$ key matrix as d_k template attention maps, each corresponding to a semantic aspect of the input. Then, the query at each pixel is d_k coefficients for each of the d_k template attention maps, respectively. Under this interpretation, efficient and dot-product attention differs in that dot-product attention first synthesizes the pixel-wise attention maps from the coefficients and lets each pixel aggregate the values with its own attention map, while efficient attention first aggregates the values by the template attention maps to form template outputs (*i.e*. global context vectors) and lets each pixel aggregate the template outputs.

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

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

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

3. allows the integration of more attention modules into a neural network and into higher-resolution parts of the network, which brings substantial performance boosts to tasks such as object detection and instance segmentation (on MS-COCO 2017); 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, [22] 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 [7, 20, 26] and is highly competitive on end-to-end speech recognition [8, 18]. [23] 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], generative adversarial image modeling [28, 2], image de-raining [13], and few-shot learning [9, 11], *etc*.

Efficient attention mainly builds upon the version of dot-product attention in the non-local module. Following [23], 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

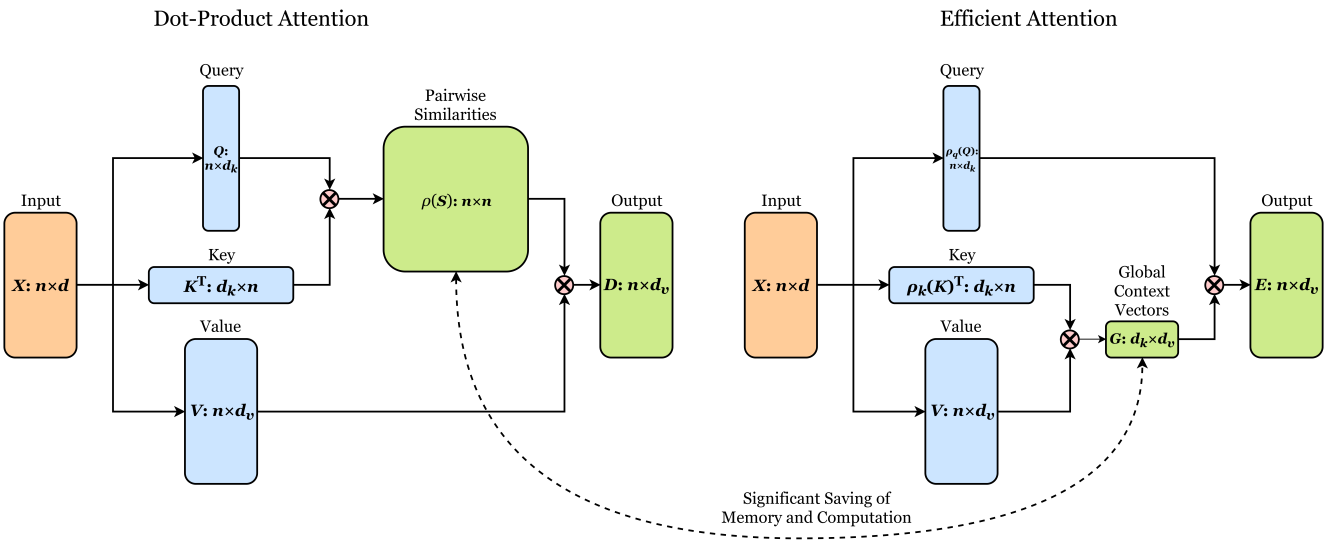

Figure 1. **Illustration of the architecture of dot-product and efficient attention.** Each box represents an input, output, or intermediate matrix. Above it is the name of the corresponding matrix, and inside are the variable name and the size of the matrix. ρ, ρ_q, ρ_k are the normalizers on $\boldsymbol{S}, \boldsymbol{Q}, \boldsymbol{K}$, respectively. n, d, d_k, d_v are the input size and the dimensionalities of the input, the keys, and the values, respectively. $\bigotimes$ denotes matrix multiplication. When ρ, ρ_q, ρ_k implement scaling normalization, the efficient attention mechanism is mathematically equivalent to dot-product attention. When they implement softmax normalization, the two mechanisms are approximately equivalent.

suppressing uninformative ones. For example, the squeeze-and-excitation (SE) module [10] 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 [24] added global max pooling beside global average pooling and an extra spatial attention submodule. GCNet [3] proposes to replace the global average pooling by an adaptive pooling layer, which uses a linear layer to compute the weight for each position. These follow-up methods further improves upon the performance of SE [10].

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

2.3. Efficient non-local operations

Recent literature proposed several methods for efficient non-local operations. LatentGNN [29] proposes to approximate the single $n \times n$ affinity matrix in the non-local [23] module by the product of three lower-rank matrices. In comparison, efficient attention is not an approximation of the non-local module, but is mathematically equivalent (using scaling normalization). In addition, there is a one-to-one mapping between the structural components of the non-local module and the efficient attention module. Therefore, in any field where the non-local module succeeded, one can guarantee the applicability of efficient attention as a drop-in replacement with substantially improved performance-cost trade-off.

CGNL [27] proposes to flatten the height, width, and channel dimensions to a hwc-dimensional vector, applies a kernel function to expand the dimensionality to $hwc \times (p+1)$, for p the degree of Taylor expansion, and models global dependencies in that space. However, after flattening the input into a vector, the feature at each position becomes a scalar, which encodes limited information for interaction modeling. In contrast, efficient attention preserves a vector representation at each pixel and is capable to model richer interactions.

Section 4.1.2 presents empirical comparison between efficient attention and these competing methods in detail, which shows that efficient attention outperforms each of them.

3. Method

3.1. A revisit of dot-product attention

Dot-product attention is a mechanism for long-range interaction modeling in neural networks. 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 $\boldsymbol{q}_i \in \mathbb{R}^{d_k}$, the key $\boldsymbol{k}_i \in \mathbb{R}^{d_k}$, and the value $\boldsymbol{v}_i \in \mathbb{R}^{d_v}$. The queries and

keys must have the same feature dimension d_k. One can measure the similarity between the i-th query and the j-th key as $\rho(\boldsymbol{q}_i^\mathsf{T}\boldsymbol{k}_j)$, where ρ is a normalization function. In general, the similarities are asymmetric, since the queries and keys 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 values from all positions via weighted summation to obtain its output feature.

If one represents all n positions' queries, keys, and values 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

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

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 1 (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.

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 substantially 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 queries $\boldsymbol{Q} \in \mathbb{R}^{n\times d_k}$, keys $\boldsymbol{K} \in \mathbb{R}^{n\times d_k}$, and values $\boldsymbol{V} \in \mathbb{R}^{n\times d_v}$. However, instead of interpreting the keys 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 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:

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

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 $\mathbf{X} \in \mathbb{R}^{h\times w\times d}$, 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

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

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

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

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

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

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 $\boldsymbol{Q}\boldsymbol{K}^\mathsf{T}$. However, they closely approximate the effect of the original softmax function. The critical property of $\sigma_{\text{row}}\left(\boldsymbol{Q}\boldsymbol{K}^\mathsf{T}\right)$ is that each row of it sums up to 1 and represents a normalized attention distribution over all positions. The matrix $\sigma_{\text{row}}(\boldsymbol{Q})\sigma_{\text{col}}(\boldsymbol{K})^\mathsf{T}$ shares this property. Therefore, the softmax variant of efficient attention is a close approximation 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 values $\boldsymbol{V}$ to produce the output at position i.

In contrast, efficient attention does not generate an attention map for each position. Instead, it interprets the keys $\boldsymbol{K} \in \mathbb{R}^{n \times d_k}$ as d_k attention maps $\boldsymbol{k}_j^\mathsf{T}$. Each $\boldsymbol{k}_j^\mathsf{T}$ 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 6 gives several concrete examples. Efficient attention uses each $\boldsymbol{k}_j^\mathsf{T}$ to aggregate the values $\boldsymbol{V}$ and produce a global context vector $\boldsymbol{g}_j$. Since $\boldsymbol{k}_j^\mathsf{T}$ 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 $\boldsymbol{g}_0, \boldsymbol{g}_1, \dots, \boldsymbol{g}_{d_k - 1}$. 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.

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 $O(dn + d^2)$ memory and $O(d^2 n)$ computational complexities, assuming the common setting of $d_v = d, d_k = \frac{d}{2}$. 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 substantial. Firstly, the input size n 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 4.2.2 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.

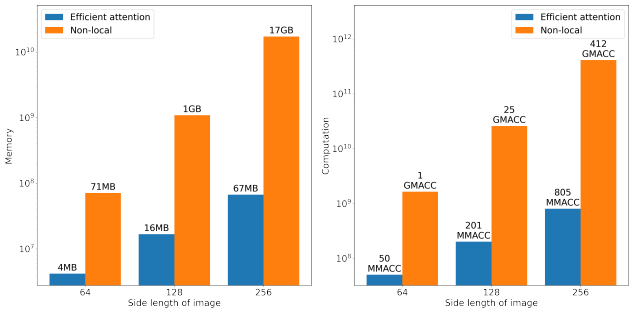

Figure 2. **Resource requirements under different input sizes.** The blue and orange bars depict the resource requirements of the efficient attention and non-local modules, respectively. The calculation assumes $d = d_v = 2d_k = 64$. This is a typical setting of self-attention for computer vision. The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

The rest of this section will give several concrete examples comparing the resource demands of the efficient attention and non-local modules. Figure 2 compares their resource consumption for image features with different sizes. Directly substituting the non-local module on the 64×64 feature map in SAGAN [28] yields a 17-time saving of memory and 33-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 (413 GMACC). With the same input size, an efficient attention module uses 1/257 the memory and 1/513 the computation. The difference is more prominent for 3D features. For a tiny $28 \times 28 \times 4$ feature volume, an efficient attention module uses less than 1/10 the memory and computation in comparison to a non-local module. On a larger $64 \times 64 \times 32$ feature volume, a non-local module requires 513 times the memory and 1025 times the computation of an efficient attention module.

4. Experiments on the MS-COCO task suite

This section presents comparison experiments on the MS-COCO 2017 dataset for object detection and instance

Table 1. **Comparison of resource usage of the efficient attention and non-local modules.** This table assumes that $d_v = d, d_k = \frac{d}{2}$, which is a common setting in the literature for dot-product attention

Metric	Efficient attention module	Non-local module
Memory (floats)	$4dn + \frac{d^2}{2}$	$4dn + n^2$
Computation (MACC)	$(6d^2 + d)n$	$(4d^2 + d)n + 3dn^2$
Memory complexity	$O(dn + d^2)$	$O(dn + n^2)$
Comp. complexity	$O(d^2n)$	$O(d^2n + dn^2)$

segmentation. The baseline is a ResNet-50 Mask R-CNN with a 5-level feature pyramid [15]. More architectural details are in the supplementary materials. The backbones initialize from ImageNet pretrainings. All other modules use random initialization. All models trained for 24 epochs on 32 NVIDIA TITAN Xp GPUs. The batch size is 64. The learning rate is 1.25×10^{-4} at the beginning of training and drops by a factor of 10 at the start of the 18th and 21st epochs. The experiments by default use softmax normalization, $d_k = d_v = 64$, and reprojection to the original number of channels.

4.1. Comparison experiments

4.1.1 Comparison with the non-local module

Table 2 reports the comparison against the non-local module. Efficient attention achieves substantially better performance-cost trade-off. As rows *res3* to *fpn5* 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, while efficient attention uses orders of magnitude less resources. Rows *res3-4+fpn3-5* and *res3-4+fpn1-5* show that under the same resource cap (TITAN Xp GPU, 12 GB VRAM), efficient attention achieves significantly better performance. Note that *res3-4+fpn3-5* is the best configuration that fits in memory for non-local modules. Further inserting non-local modules to *fpn1* or *fpn2* would require gigabytes of memory *per example*.

4.1.2 Comparison with competing methods

Table 3 shows the comparison of absolute performance and performance improvement with competing approaches on MS-COCO 2017 object detection and instance segmentation. EA models has the highest performance and performance improvement in all settings while using the least resources. Note that EA's baseline models are significantly stronger, which make the improvements more valuable.

4.2. Ablation studies

4.2.1 Attention normalization

These experiments empirically compared the two methods Section 3.2 specified, namely scaling and softmax normalization. Table 4 reports the experimental outcomes. The results demonstrate that the effectiveness does not depend on the specific normalization method. Following [23], all other experiments used softmax normalization.

4.2.2 Dimensionality of the keys

These experiments tested the impact of the dimensionality of the keys on the effect of efficient attention. As in Table 5, decreasing the dimensionality of the keys from 128 to 32 caused minimal accuracy change. This result reinforces the hypothesis in Section 1 that most attention maps are expressible as linear combinations of a limited set of template attention maps. Therefore, researchers can reduce the dimensionality of the keys and queries in efficient attention modules to further save resources.

5. Experiments on other tasks

5.1. Stereo depth estimation

The experiments on efficient attention for stereo depth estimation used the Scene Flow dataset, a large-scale synthesized dataset with 39824 stereo frame pairs. The baseline is PSMNet [4], a clean model with near state-of-the-art performance. The experiments empirically determined the optimal hyperparamters, which significantly outperform the setting in [4] (batch size is 24, learning rate is 2×10^{-3}, training length is 100 epochs, and the rest is the same as in [4]), as in Table 7. On top of the strong baseline, inserting an efficient attention module after the last 3D hourglass leads to further improvement and achieves a new state-of-the-art. In comparison, inserting a non-local module at the same place would require an astronomical 9.68 TB of memory, prohibiting any attempt to verify its effectiveness. Table 8 compares EA-PSMNet with other state-of-the-art approaches and shows that it substantially outperforms all competing methods.

5.2. Temporal action localization

This section presents experiments for temporal action localization on the THUMOS14 [12] dataset. The baseline is R-C3D [25]. The experiment added two efficient attention

Table 2. **Comparison between the efficient attention and non-local modules on MS-COCO 2017 object detection and instance segmentation.** *Box*, *mask*, *mem.*, and *comp.* stand for box AP, mask AP, memory (in bytes), and computation (in MACC), respectively. *Mem.* and *comp.* only count the attention module(s). *res*$\{x\}$ and *fpn*$\{x\}$ indicate inserting attention modules after the x-th ResBlock group or FPN level x, respectively. *res*$\{x$-$y\}$ and *fpn*$\{x$-$y\}$ similarly mean inserting after every ResBlock group or FPN level within the range $[x, y]$

	EA module				Non-local module				
Layer(s)	Box	Mask	Mem.	Comp.	Box	Mask	Mem.	Comp.	Input size
None	39.4	35.1	0	0	39.4	35.1	0	0	N/A
res3	40.2	36.0	41.3 M	1.21 G	40.3	35.9	122 M	3.74 G	56×80
res4	40.2	35.9	19.5 M	596 M	40.1	36.0	24.5 M	748 M	28×40
fpn1	39.9	35.8	220 M	5.28 G	OOM	OOM	20.8 G	662 G	224×320
fpn2	39.7	35.7	55.1 M	1.32 G	OOM	OOM	1.34 G	42.3 G	112×160
fpn3	39.7	35.5	13.8 M	330 M	39.8	35.5	94.0 M	2.86 G	56×80
fpn4	39.7	35.4	3.46 M	82.6 M	39.5	35.3	8.46 M	234 M	28×40
fpn5	39.6	35.3	877 K	20.6 M	39.4	35.2	1.17 M	28.4 M	14×20
res3-4+fpn3-5	40.6	36.2	78.9 M	2.24 G	**40.7**	**36.3**	250 M	7.62 G	N/A
res3-4+fpn1-5	**41.2**	**36.7**	354 M	8.85 G	OOM	OOM	22.4 G	712 G	N/A

Table 3. **Comparison vs. competing methods on MS-COCO 2017 object detection and instance segmentation.** For each model, the number outside the parentheses is the AP, and the number inside is the AP improvement over baseline. The table reports number of parameters and amount of computation as a percentage increase over the baseline Mask R-CNN. The team obtained these metrics by measuring the official open-source implementations of [29, 27]. The table does not report results for CGNL with ResNet-101 and ResNeXt-101 since [27] did not report such results. The Table omits parameters and computation for instance segmentation since all methods modified the backbone, which the bounding box and the instance mask branches share. Therefore, the table reports the total parameter and computation change only in the rows for object detection to avoid repetition

AP type	Method	ResNet-50	ResNet-101	ResNeXt-101	Parameters	Computation
Box	EA	(**+1.8**) **41.2**	(**+1.8**) **43.1**	(**+1.4**) **44.9**	**+2.9**%	**+5.3**%
	LatentGNN [29]	(+1.7) 39.5	(+1.5) 41.0	(+1.1) 43.2	+11.1%	+7.6%
	CGNL [27]	(+1.2) 35.7	-	-	+21.7%	+5.7%
Mask	EA	(**+1.6**) **36.7**	(**+1.3**) **37.9**	(**+1.0**) **39.5**	-	-
	LatentGNN [29]	(+1.2) 35.4	(**+1.3**) 37.2	(**+1.0**) 38.8	-	-
	CGNL [27]	(+0.8) 31.2	-	-	-	-

Table 4. **Experiments on attention normalization methods on MS-COCO 2017 object detection and instance segmentation.** Experiments inserted efficient attention modules at fpn1-5

Method	Box AP	Mask AP
Scaling	**40.2**	35.9
Softmax	**40.2**	**36.0**

Table 5. **Experiments on the dimensionality of the keys on MS-COCO 2017 object detection and instance segmentation.** Experiments inserted efficient attention modules at res3-4+fpn3-5

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

modules after *res3* and *res4* in the ResNet-50 backbone. Table 6 presents the results. At the table shows, efficient attention substantially improved the performance for this task.

6. Visualization

Figure 3 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 2. 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 template

Table 6. **Experiments on THUMOS14 temporal action localization.** *mAP@x* stands for mean average precision at IoU threshold x. *EA R-C3D* is this paper's model. Both models used ResNet-50 as the backbone

Model	mAP@0.1	mAP@0.2	mAP@0.3	mAP@0.4	mAP@0.5	mAP@0.6	mAP@0.7
R-C3D	54.2	54.1	50.0	45.6	37.3	29.2	18.5
EA R-C3D	**60.3**	**59.8**	**56.8**	**51.3**	**43.4**	**33.2**	**21.8**

Table 7. **Experiments on Scene Flow stereo depth estimation.** *EPE* stands for end-point error and is lower the better. *EA-PSMNet* is this paper's model. *OOM* indicates out of memory. *Memory* only counts the attention module

Model	EPE	Memory
PSMNet (original)	1.09	0
PSMNet (baseline)	0.51	0
EA-PSMNet	**0.48**	796 MB
Nonlocal-PSMNet	OOM	9.68 TB

Table 8. **Comparison with the state-of-the-art on Scene Flow stereo depth estimation.** *EPE* stands for end-point error and is lower the better. *EA-PSMNet* is this paper's model

Model	EPE
iResNet-i2 [17]	1.40
EdgeStereo [21]	1.12
PSMNet [4]	1.09
CSPN [5]	0.78
LEAStereo [6]	0.78
EA-PSMNet	**0.48**

Figure 3. **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.

attention maps each focusing on a semantically significant area.

7. 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, and stereo depth estimation. 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 [28, 2] and most tasks in natural language processing [22, 19, 7, 20]. 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.

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

[3] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *ICCV*, 2019.

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

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

[6] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yuchao Dai, Xiaojun Chang, Tom Drummond, Hongdong Li, and Zongyuan Ge. Hierarchical neural architecture search for deep stereo matching. *arXiv preprint arXiv:2010.13501*, 2020.

[7] 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.

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

[9] Ruibing Hou, Hong Chang, MA Bingpeng, Shiguang Shan, and Xilin Chen. Cross attention network for few-shot classification. In *NeurIPS*, pages 4003–4014, 2019.

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

[11] Tao Hu, Pascal Mettes, Jia-Hong Huang, and Cees GM Snoek. Silco: Show a few images, localize the common object. In *ICCV*, pages 5067–5076, 2019.

[12] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. `http://crcv.ucf.edu/THUMOS14/`, 2014.

[13] 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.

[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.

[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.

[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.

[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.

[18] 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.

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

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

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

[22] 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.

[23] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

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

[25] Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *ICCV*, 2017.

[26] 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.

[27] Kaiyu Yue, Ming Sun, Yuchen Yuan, Feng Zhou, Errui Ding, and Fuxin Xu. Compact generalized non-local network. In *NeurIPS*, 2018.

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

[29] Songyang Zhang, Shipeng Yan, and Xuming He. Latent-GNN: Learning efficient non-local relations for visual recognition. In *ICML*, 2019.