Exploring Self-attention for Image Recognition

Hengshuang Zhao
CUHK

Jiaya Jia
CUHK

Vladlen Koltun
Intel Labs

Abstract

Recent work has shown that self-attention can serve as a basic building block for image recognition models. We explore variations of self-attention and assess their effectiveness for image recognition. We consider two forms of self-attention. One is pairwise self-attention, which generalizes standard dot-product attention and is fundamentally a set operator. The other is patchwise self-attention, which is strictly more powerful than convolution. Our pairwise self-attention networks match or outperform their convolutional counterparts, and the patchwise models substantially outperform the convolutional baselines. We also conduct experiments that probe the robustness of learned representations and conclude that self-attention networks may have significant benefits in terms of robustness and generalization.

1. Introduction

Convolutional networks have revolutionized computer vision. Thirty years ago, they were applied successfully to recognizing handwritten digits [19]. Building directly on this work, convolutional networks were scaled up in 2012 to achieve breakthrough accuracy on the ImageNet dataset, outperforming all prior methods by a large margin and launching the deep learning era in computer vision [18, 29]. Subsequent architectural improvements yielded successively larger and more accurate convolutional networks for image recognition, including GoogLeNet [31], VGG [30], ResNet [12], DenseNet [16], and squeeze-and-excitation [15]. These architectures in turn serve as templates for applications in computer vision and beyond.

All these networks, from LeNet [19] onwards, are based fundamentally on the discrete convolution. The discrete convolution operator $*$ can be defined as follows:

$$(F * k)(\mathbf{p}) = \sum_{\mathbf{s}+\mathbf{t}=\mathbf{p}} F(\mathbf{s})\, k(\mathbf{t}). \tag{1}$$

Here F is a discrete function and k is a discrete filter. A key characteristic of the convolution is its translation invariance: the same filter k is applied across the image F. While the convolution has undoubtedly been effective as the basic operator in modern image recognition, it is not without drawbacks. For example, the convolution lacks rotation invariance. The number of parameters that must be learned grows with the footprint of the kernel k. And the stationarity of the filter can be seen as a drawback: the aggregation of information from a neighborhood cannot adapt to its content. Is it possible that networks based on the discrete convolution are a local optimum in the design space of image recognition models? Could other parts of the design space yield models with interesting new capabilities?

Recent work has shown that self-attention may constitute a viable alternative for building image recognition models [13, 27]. The self-attention operator has been adopted from natural language processing, where it serves as the basis for powerful architectures that have displaced recurrent and convolutional models across a variety of tasks [33, 7, 6, 40]. The development of effective self-attention architectures in computer vision holds the exciting prospect of discovering models with different and perhaps complementary properties to convolutional networks.

In this work, we explore variations of the self-attention operator and assess their effectiveness as the basic building block for image recognition models. We explore two types of self-attention. The first is *pairwise* self-attention, which generalizes the standard dot-product attention used in natural language processing [33]. Pairwise attention is compelling because, unlike the convolution, it is fundamentally a set operator, rather than a sequence operator. Unlike the convolution, it does not attach stationary weights to specific locations ($\mathbf{s}$ in equation (1)) and is invariant to permutation and cardinality. One consequence is that the footprint of a self-attention operator can be increased (e.g., from a 3×3 to a 7×7 patch) or even made irregular without any impact on the number of parameters. We present a number of variants of pairwise attention that have greater expressive power than dot-product attention while retaining these invariance properties. In particular, our weight computation does not collapse the channel dimension and allows the feature aggregation to adapt to each channel.

Next, we explore a different class of operators, which we term *patchwise* self-attention. These operators, like the convolution, have the ability to uniquely identify specific locations within their footprint. They do not have the permu-

tation or cardinality invariance of pairwise attention, but are strictly more powerful than convolution.

Our experiments indicate that both forms of self-attention are effective for building image recognition models. We construct self-attention networks that can be directly compared to convolutional ResNet models [12], and conduct experiments on the ImageNet dataset [29]. Our pairwise self-attention networks match or outperform their convolutional counterparts, with similar or lower parameter and FLOP budgets. Controlled experiments also indicate that our vectorial operators outperform standard scalar attention. Furthermore, our patchwise models substantially outperform the convolutional baselines. For example, our mid-sized SAN15 with patchwise attention outperforms the much larger ResNet50, with a 78% top-1 accuracy for SAN15 versus 76.9% for ResNet50, with a 37% lower parameter and FLOP count. Finally, we conduct experiments that probe the robustness of learned representations and conclude that self-attention networks may have significant benefits in terms of robustness and generalization.

2. Related Work

Convolutional networks. Convolutional networks have come to dominate computer vision. More than two decades after their pioneering application to recognizing handwritten digits [19], ConvNets became mainstream after their successful application to image recognition on the ImageNet dataset [18, 29]. A succession of increasingly powerful convolutional architectures for image recognition followed [31, 30, 12, 16, 15]. These serve as the basis for models developed for other computer vision tasks, such as semantic segmentation [22, 3, 42, 44] and object detection [10, 9, 28, 21].

Self-attention. Self-attention models have revolutionized machine translation and natural language processing more broadly [33, 37, 7, 6, 40]. This has inspired applications of self-attention and related ideas to image recognition [5, 34, 15, 14, 45, 46, 13, 1, 27], image synthesis [43, 26, 2], image captioning [39, 41, 4], and video prediction [17, 35].

Until very recently, applications of self-attention in computer vision were complementary to convolution: forms of self-attention were primarily used to create layers that were used in addition to, to modulate the output of, or otherwise in combination with convolutions. In channelwise attention models [34, 15, 14], attention weights reweight activations in different channels. Other approaches [4, 36, 8] adopted both spatial and channel attention. A number of methods learned to reweight convolutional activations or offset the taps of convolutional kernels [5, 15, 34, 36, 46], thus retaining the basic principle of convolutional feature construction. Others applied self-attention in specific modules that were appended to convolutional structures [35, 45]. Bello et al. [1] combined convolutional and self-attention processing streams, but found that the global self-attention they used was not sufficiently powerful to replace convolutions entirely. Jia et al. [17] explored dynamic filter networks, which generalized convolutions, but the construction incurred significant memory and computational costs and was not scaled up to high-resolution images and larger datasets.

Most closely related to our work are the recent results of Hu et al. [13] and Ramachandran et al. [27]. One of their key innovations is restricting the scope of self-attention to a local patch (for example, 7×7 pixels), in contrast to earlier constructions that applied self-attention globally over a whole feature map [35, 1]. Such local attention is key to limiting the memory and computation consumed by the model, facilitating successful application of self-attention throughout the network, including early high-resolution layers. Our work builds on these results and explores a broader variety of self-attention formulations. In particular, our primary self-attention mechanisms compute a *vector* attention that adapts to different channels, rather than a shared scalar weight. We also explore a family of patchwise attention operators that are structurally different from the forms used in [13, 27] and constitute strict generalizations of convolution. We show that all the presented forms of self-attention can be implemented at scale, with favorable parameter and FLOP budgets.

3. Self-attention Networks

In convolutional networks for image recognition, the layers of the network perform two functions. The first is feature aggregation, which the convolution operation performs by combining features from all locations tapped by the kernel. The second function is feature transformation, which is performed by successive linear mappings and nonlinear scalar functions: these successive mappings and nonlinear operations shatter the feature space and give rise to complex piecewise mappings.

One observation that underlies our construction is that these two functions – feature aggregation and feature transformation – can be decoupled. If we have a mechanism that performs feature aggregation, then feature transformation can be performed by perceptron layers that process each feature vector (for each pixel) separately. A perceptron layer consists of a linear mapping and a nonlinear scalar function: this pointwise operation performs feature transformation. Our construction therefore focuses on feature aggregation.

The convolution operator performs feature aggregation by a fixed kernel that applies pretrained weights to linearly combine feature values from a set of nearby locations. The weights are fixed and do not adapt to the content of the features. And since each location must be processed with a dedicated weight vector, the number of parameters scales linearly with the number of aggregated features. We present a number of alternative aggregation schemes and construct high-performing image recognition architectures that interleave feature aggregation (via self-attention) and feature transformation (via elementwise perceptrons).

3.1. Pairwise Self-attention

We explore two types of self-attention. The first, which we refer to as *pairwise*, has the following form:

$$\mathbf{y}_i = \sum_{j \in \mathcal{R}(i)} \alpha(\mathbf{x}_i, \mathbf{x}_j) \odot \beta(\mathbf{x}_j), \tag{2}$$

where $\odot$ is the Hadamard product, i is the spatial index of feature vector $\mathbf{x}_i$ (i.e., its location in the feature map), and $\mathcal{R}(i)$ is the local footprint of the aggregation. The footprint $\mathcal{R}(i)$ is a set of indices that specifies which feature vectors are aggregated to construct the new feature $\mathbf{y}_i$.

The function β produces the feature vectors $\beta(\mathbf{x}_j)$ that are aggregated by the adaptive weight vectors $\alpha(\mathbf{x}_i, \mathbf{x}_j)$. Possible instantiations of this function, along with feature transformation elements that surround self-attention operations in our architecture, are discussed later in this section.

The function α computes the weights $\alpha(\mathbf{x}_i, \mathbf{x}_j)$ that are used to combine the transformed features $\beta(\mathbf{x}_j)$. To simplify exposition of different forms of self-attention, we decompose α as follows:

$$\alpha(\mathbf{x}_i, \mathbf{x}_j) = \gamma(\delta(\mathbf{x}_i, \mathbf{x}_j)). \tag{3}$$

The relation function δ outputs a single vector that represents the features $\mathbf{x}_i$ and $\mathbf{x}_j$. The function γ then maps this vector into a vector that can be combined with $\beta(\mathbf{x}_j)$ as shown in Eq. 2.

The function γ enables us to explore relations δ that produce vectors of varying dimensionality that need not match the dimensionality of $\beta(\mathbf{x}_j)$. It also allows us to introduce additional trainable transformations into the construction of the weights $\alpha(\mathbf{x}_i, \mathbf{x}_j)$, making this construction more expressive. This function performs a linear mapping, followed by a nonlinearity, followed by another linear mapping; i.e., γ=$\{$`Linear`$\rightarrow$`ReLU`$\rightarrow$`Linear`$\}$. The output dimensionality of γ does not need to match that of β as attention weights can be shared across a group of channels.

We explore multiple forms for the relation function δ:

Summation: $\delta(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) + \psi(\mathbf{x}_j)$

Subtraction: $\delta(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) - \psi(\mathbf{x}_j)$

Concatenation: $\delta(\mathbf{x}_i, \mathbf{x}_j) = [\varphi(\mathbf{x}_i), \psi(\mathbf{x}_j)]$

Hadamard product: $\delta(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \odot \psi(\mathbf{x}_j)$

Dot product: $\delta(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^\top \psi(\mathbf{x}_j)$

Here φ and ψ are trainable transformations such as linear mappings, and have matching output dimensionality. With summation, subtraction, and Hadamard product, the dimensionality of $\delta(\mathbf{x}_i, \mathbf{x}_j)$ is the same as the dimensionality of the transformation functions. With concatenation, the dimensionality of $\delta(\mathbf{x}_i, \mathbf{x}_j)$ will be doubled. With the dot product, the dimensionality of $\delta(\mathbf{x}_i, \mathbf{x}_j)$ is 1.

Position encoding. A distinguishing characteristic of pairwise attention is that feature vectors $\mathbf{x}_j$ are processed independently and the weight computation $\alpha(\mathbf{x}_i, \mathbf{x}_j)$ cannot incorporate information from any location other than i and j. To provide some spatial context to the model, we augment the feature maps with position information. The position is encoded as follows. The horizontal and vertical coordinates along the feature map are first normalized to the range $[-1, 1]$ in each dimension. These normalized two-dimensional coordinates are then passed through a trainable linear layer, which can map them to an appropriate range for each layer in the network. This linear mapping outputs a two-dimensional position feature $\mathbf{p}_i$ for each location i in the feature map. For each pair (i, j) such that $j \in \mathcal{R}(i)$, we encode the relative position information by calculating the difference $\mathbf{p}_i - \mathbf{p}_j$. The output of $\delta(\mathbf{x}_i, \mathbf{x}_j)$ is augmented by concatenating $[\mathbf{p}_i - \mathbf{p}_j]$ prior to the mapping γ.

3.2. Patchwise Self-attention

The other type of self-attention we explore is referred to as *patchwise* and has the following form:

$$\mathbf{y}_i = \sum_{j \in \mathcal{R}(i)} \alpha(\mathbf{x}_{\mathcal{R}(i)})_j \odot \beta(\mathbf{x}_j), \tag{4}$$

where $\mathbf{x}_{\mathcal{R}(i)}$ is the patch of feature vectors in the footprint $\mathcal{R}(i)$. $\alpha(\mathbf{x}_{\mathcal{R}(i)})$ is a tensor of the same spatial dimensionality as the patch $\mathbf{x}_{\mathcal{R}(i)}$. $\alpha(\mathbf{x}_{\mathcal{R}(i)})_j$ is the vector at location j in this tensor, corresponding spatially to the vector $\mathbf{x}_j$ in $\mathbf{x}_{\mathcal{R}(i)}$.

In patchwise self-attention, we allow the construction of the weight vector that is applied to $\beta(\mathbf{x}_j)$ to refer to and incorporate information from all feature vectors in the footprint $\mathcal{R}(i)$. Note that, unlike pairwise self-attention, patchwise self-attention is no longer a set operation with respect to the features $\mathbf{x}_j$. It is not permutation-invariant or cardinality-invariant: the weight computation $\alpha(\mathbf{x}_{\mathcal{R}(i)})$ can index the feature vectors $\mathbf{x}_j$ individually, by location, and can intermix information from feature vectors from different locations within the footprint. Patchwise self-attention is thus strictly more powerful than convolution.

We decompose $\alpha(\mathbf{x}_{\mathcal{R}(i)})$ as follows:

$$\alpha(\mathbf{x}_{\mathcal{R}(i)}) = \gamma(\delta(\mathbf{x}_{\mathcal{R}(i)})). \tag{5}$$

The function γ maps a vector produced by $\delta(\mathbf{x}_{\mathcal{R}(i)})$ to a tensor of appropriate dimensionality. This tensor comprises weight vectors for all locations j. The function δ combines the feature vectors $\mathbf{x}_j$ from the patch $\mathbf{x}_{\mathcal{R}(i)}$. We explore the following forms for this combination:

Star-product: $\delta(\mathbf{x}_{\mathcal{R}(i)}) = [\varphi(\mathbf{x}_i)^\top \psi(\mathbf{x}_j)]_{\forall j \in \mathcal{R}(i)}$

Clique-product: $\delta(\mathbf{x}_{\mathcal{R}(i)}) = [\varphi(\mathbf{x}_j)^\top \psi(\mathbf{x}_k)]_{\forall j,k \in \mathcal{R}(i)}$

Concatenation: $\delta(\mathbf{x}_{\mathcal{R}(i)}) = [\varphi(\mathbf{x}_i), [\psi(\mathbf{x}_j)]_{\forall j \in \mathcal{R}(i)}]$

Layers	Output Size	SAN10	SAN15	SAN19
Input	224×224×3	64-d linear		
Transition	112×112×64	2×2, stride 2 max pool → 64-d linear		
SA Block	112×112×64	[3×3, 16-d sa 64-d linear] ×2	[3×3, 16-d sa 64-d linear] ×3	[3×3, 16-d sa 64-d linear] ×3
Transition	56×56×256	2×2, stride 2 max pool → 256-d linear		
SA Block	56×56×256	[7×7, 64-d sa 256-d linear] ×1	[7×7, 64-d sa 256-d linear] ×2	[7×7, 64-d sa 256-d linear] ×3
Transition	28×28×512	2×2, stride 2 max pool → 512-d linear		
SA Block	28×28×512	[7×7, 128-d sa 512-d linear] ×2	[7×7, 128-d sa 512-d linear] ×3	[7×7, 128-d sa 512-d linear] ×4
Transition	14×14×1024	2×2, stride 2 max pool → 1024-d linear		
SA Block	14×14×1024	[7×7, 256-d sa 1024-d linear] ×4	[7×7, 256-d sa 1024-d linear] ×5	[7×7, 256-d sa 1024-d linear] ×6
Transition	7×7×2048	2×2, stride 2 max pool → 2048-d linear		
SA Block	7×7×2048	[7×7, 512-d sa 2048-d linear] ×1	[7×7, 512-d sa 2048-d linear] ×2	[7×7, 512-d sa 2048-d linear] ×3
Classification	1×1×1000	global average pool → 1000-d linear → softmax		

Table 1. Self-attention networks for image recognition. 'C-d linear' means that the output dimensionality of the linear layer is 'C'. 'C-d sa' stands for a self-attention operation with output dimensionality 'C'. SAN10, SAN15, and SAN19 are in rough correspondence with ResNet26, ResNet38, and ResNet50, respectively. The number X in SANX refers to the number of self-attention blocks. Our architectures are based fully on self-attention.

3.3. Self-attention Block

The self-attention operations described in Sections 3.1 and 3.2 can be used to construct residual blocks [12] that perform both feature aggregation and feature transformation. Our self-attention block is illustrated in Figure 1. The input feature tensor (channel dimensionality C) is passed through two processing streams. The left stream evaluates the attention weights α by computing the function δ (via the mappings φ and ψ) and a subsequent mapping γ. The right stream applies a linear transformation β that transforms the input features and reduces their dimensionality for efficient processing. The outputs of the two streams are then aggregated via a Hadamard product. The combined features are passed through a normalization and an elementwise nonlinearity, and are processed by a final linear layer that expands their dimensionality back to C.

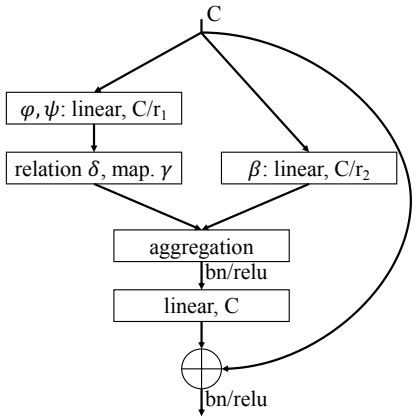

Figure 1. Our self-attention block. C is the channel dimensionality. The left stream evaluates the attention weights α, the right stream transforms the features via a linear mapping β. Both streams reduce the channel dimensionality for efficient processing. The outputs of the streams are aggregated via a Hadamard product and the dimensionality is subsequently expanded back to C.

3.4. Network Architectures

Our network architectures generally follow residual networks, which we will use as baselines [12]. Table 1 presents three architectures obtained by stacking self-attention blocks at different resolutions. These architectures – SAN10, SAN15, and SAN19 – are in rough correspondence with ResNet26, ResNet38, and ResNet50. The number X in SANX refers to the number of self-attention blocks. Our architectures are based fully on self-attention.

Backbone. The backbone of SAN has five stages, each with different spatial resolution, yielding a resolution reduction factor of 32. Each stage comprises multiple self-attention blocks. Consecutive stages are bridged by transition layers that reduce spatial resolution and expand channel dimensionality. The output of the last stage is processed by a classification layer that comprises global average pooling, a linear layer, and a softmax.

Transition. Transition layers reduce spatial resolution, thus reducing the computational burden and expanding receptive field. The transition comprises a batch normalization layer, a ReLU [25], 2×2 max pooling with stride 2, and a linear mapping that expands channel dimensionality.

Footprint of self-attention. The local footprint $\mathcal{R}(i)$ controls the amount of context gathered by a self-attention operator from the preceding feature layer. We set the footprint size to 7×7 for the last four stages of SAN. The footprint is set to 3×3 in the first stage due to the high resolution of that stage and the consequent memory consumption. Note that increasing the footprint size has no impact on the number of parameters in pairwise self-attention. We will study the effect of footprint size on accuracy, capacity, and FLOPs in Section 5.3.

Instantiations. The number of self-attention blocks in each stage can be adjusted to obtain networks with different capacities. In the networks presented in Table 1, the number of self-attention blocks used in the last four stages is the same as the number of residual blocks in ResNet26, ResNet38, and ResNet50, respectively.

4. Comparison

In this section, we relate the family of self-attention operators presented in Section 3 to other constructions, including convolution [19] and scalar attention [33, 35, 27, 13]. Table 2 summarizes some differences between the constructions. These are discussed in more detail below.

Operation	Content adaptive	Channel adaptive
Convolution [19]	✗	✓
Scalar attention [33, 35, 27, 13]	✓	✗
Vector attention (ours)	✓	✓

Table 2. The convolution does not adapt to the content of the image. Scalar attention produces scalar weights that do not vary along the channel dimension. Our operators efficiently compute attention weights that adapt across both spatial dimensions and channels.

Convolution. The regular convolution operator has fixed kernel weights that are independent of the content of the image. It does not adapt to the input content. The kernel weights can vary across channels.

Scalar attention. Scalar attention, as used in the transformer [33] and related constructions in computer vision [35, 27, 13], typically has the following form:

$$\mathbf{y}_i = \sum_{j\in\mathcal{R}(i)} \left(\varphi(\mathbf{x}_i)^\top \psi(\mathbf{x}_j)\right)\beta(\mathbf{x}_j) \tag{6}$$

(A softmax and other forms of normalization can be added.) Unlike the convolution, the aggregation weights can vary across different locations, depending on the content of the image. On the other hand, the weight $\varphi(\mathbf{x}_i)^\top\psi(\mathbf{x}_j)$ is a *scalar* that is shared across all channels. (Hu et al. [13] explored alternatives to the dot product, but these alternatives operated on scalar weights that were likewise shared across channels.) This construction does not adapt the attention weights at different channels. Although this can be mitigated to some extent by introducing multiple heads [33], the number of heads is a small constant and scalar weights are shared by all channels within a head.

Vector attention (ours). The operators presented in Section 3 subsume scalar attention and generalize it in important ways. First, within the pairwise attention family, the relation function δ can produce *vector* output. This is the case for the summation, subtraction, Hadamard, and concatenation forms. This vector can then be further processed and mapped to the right dimensionality by γ, which can also take position encoding channels as input. The mapping γ produces a vector that has compatible dimensionality to the transformed features β. This gives the construction significant flexibility in accommodating different relation functions and auxiliary inputs, expressive power due to multiple linear mappings and nonlinearities along the computation graph, ability to produce attention weights that vary along both spatial and channel dimensions, and computational efficiency due to the ability to reduce dimensionality by the mappings γ and β.

The patchwise family of operators generalizes convolution while retaining parameter and FLOP efficiency. This family of operators produces weight vectors for all positions along a feature map that also vary along the channel dimension. The weight vectors are informed by the entirety of the footprint of the operator.

5. Experiments

We conduct experiments on ImageNet classification [29]. The dataset contains 1.28 million training images and 50K validation images from 1000 different classes. For comparisons of self-attention networks with convolutional networks such as ResNet, we train on the original training set and report accuracy (single center crop) on the original validation set (referred to as 'val-original'). For controlled experiments and ablation studies on self-attention networks, we split a separate validation set out of the original training set by randomly sampling 50 images from the training set for each category: this is referred to as 'val-split'. This ensures that architectural and hyperparameter choices are not made on the same set that is used for comparisons with external baselines.

5.1. Implementation

We train all models from scratch for 100 epochs. We use the cosine learning rate schedule with base learning rate 0.1 [23]. We apply standard data augmentation on ImageNet, including random cropping to 224×224 patches [31], random horizontal flipping, and normalization. We use synchronous SGD with minibatch size 256 on 8 GPUs. We

Method	ResNet26 vs. SAN10				ResNet38 vs. SAN15				ResNet50 vs. SAN19			
	top-1	top-5	Params	Flops	top-1	top-5	Params	Flops	top-1	top-5	Params	Flops
Convolutional	73.6	91.7	13.7M	2.4G	76.0	93.0	19.6M	3.2G	76.9	93.5	25.6M	4.1G
SAN, pairwise	74.9	92.1	10.5M	2.2G	76.6	93.1	14.1M	3.0G	76.9	93.4	17.6M	3.8G
SAN, patchwise	77.1	93.5	11.8M	1.9G	78.0	93.9	16.2M	2.6G	78.2	93.9	20.5M	3.3G

Table 3. Comparison of self-attention networks and convolutional residual networks on ImageNet classification. Single-crop testing on the val-original set.

use label smoothing regularization with coefficient 0.1 [32]. Momentum and weight decay are set to 0.9 and 1e-4, respectively [12, 38, 11].

Our convolutional network baselines are ResNet26, ResNet38, and ResNet50 [12]. ResNet38 and ResNet26 are constructed by taking ResNet50 as the starting point and removing one or two residual blocks from each stage, respectively. For self-attention blocks, we use $r_1 = 16$ and $r_2 = 4$ by default (see Figure 1 for notation). The number of channels sharing the same attention weight is set to 8.

5.2. Comparison to Convolutional Networks

Table 3 reports the results of the main comparison of the presented self-attention networks to convolutional counterparts. For pairwise self-attention, we use the subtraction relation. For patchwise self-attention, we use concatenation. These decisions are based on the controlled experiments reported in Section 5.3. The pairwise models match or outperform the convolutional baselines, with similar or lower parameter and FLOP budgets. The patchwise models perform even better. For example, the patchwise SAN10 outperforms not only ResNet26 but also ResNet38, with a 40% lower parameter count and a 41% lower FLOP count versus the latter. Likewise, the patchwise SAN15 outperforms not only ResNet38 but also ResNet50 (78% top-1 accuracy for SAN15 versus 76% for ResNet38 and 76.9% for ResNet50), with a 37% lower parameter count and a 37% lower FLOP count versus the latter.

5.3. Controlled Experiments

Relation function. Table 4 reports the results of a controlled comparison of different relation functions on the val-split set. For pairwise self-attention, summation, subtraction, and Hadamard product achieve similar accuracy. These relation functions outperform concatenation and dot product. In particular, these experiments indicate that vector self-attention outperforms scalar self-attention. For patchwise self-attention, concatenation achieves slightly higher accuracy than star-product and clique-product.

We also attempted a controlled comparison with the self-attention configuration of Ramachandran et al. [27]. Unfortunately, their implementation has not been released at the time of writing, and there are many subtle differences that can impact results, from the configuration of the input stem, to positional encoding, to architectural hyperparameters, to

Method		top-1	top-5	Params	Flops
Conv.-ResNet26		76.0	92.8	13.7M	2.4G
SAN10-pair.	summation	77.4	93.3	10.5M	2.2G
	subtraction	77.4	93.3	10.5M	2.2G
	concatenate	76.4	92.6	10.6M	2.5G
	Had. product	77.4	93.4	10.5M	2.2G
	dot product	77.0	93.0	10.5M	1.8G
SAN10-patch.	star-product	78.7	94.0	10.9M	1.7G
	clique-product	79.1	94.2	11.5M	1.9G
	concatenation	79.3	94.2	11.8M	1.9G

Table 4. Controlled comparison of different relation functions on the val-split set.

data augmentation and the training schedule. We attempted to control for extraneous differences as much as possible by using the same overall network architecture (SAN10) and training setup (Section 5.1). Within this framework, we reproduced the self-attention block of Ramachandran et al. as closely as possible. In particular, we used their grouped dot-product attention, added position information, and set r_1 and r_2 (the bottleneck dimension reduction factor) to 4. This yielded top-1 accuracy of 71.7% and top-5 accuracy of 89.9%, lower than our self-attention configurations with the same setup and lower than the results reported by Ramachandran et al. (The number of parameters is 13.9M, the number of FLOPs is 2.3G.) Considered in conjunction with our controlled experiments, this appears to support the conclusion that vector self-attention is a useful building block for self-attention networks in computer vision. Our results also indicate that patchwise self-attention may be particularly powerful and merits further study. Finally, the difficulties in reproducing results reported in related work highlight the importance of timely release of reference implementations. We will release our full implementation and experimental setup open-source to facilitate comparison and assist future work in this area.

Mapping function. We conduct an ablation study on the number of linear layers in the attention mapping function γ. The results are listed in Table 5. For pairwise models, using two linear layers yields the highest accuracy. For patchwise models, different settings yield similar accuracy. Using only one linear layer for attention mapping incurs significant memory and computation costs in the patchwise setting. Multiple layers enable the introduction of bottlenecks that

reduce dimensionality and thus reduce memory and computation costs. Considering all the factors, we use two linear layers (the intermediate setting in Table 5) as our default for all models.

Method		top-1	top-5	Params	Flops
Conv.-ResNet26		76.0	92.8	13.7M	2.4G
SAN10-pair.	`L`	75.8	92.3	10.5M	1.8G
	`L`→`R`→`L`	77.4	93.3	10.5M	2.2G
	`L`→`R`→`L`→`R`→`L`	77.0	93.0	10.6M	2.5G
SAN10-patch.	`L`	79.3	94.2	53.5M	9.5G
	`L`→`R`→`L`	79.3	94.2	11.8M	1.9G
	`L`→`R`→`L`→`R`→`L`	79.5	94.3	12.7M	2.0G

Table 5. Controlled comparison of different mapping functions on the val-split set. `L` and `R` denote `Linear` and `ReLU` layers, respectively.

Transformation functions. We now evaluate whether the use of three distinct transformation functions (φ, ψ, and β) is helpful. The results are reported in Table 6. Using three distinct learnable transformations is generally the best choice. An additional advantage is that a distinct β transformation enables the use of different bottleneck dimension reduction factors r_1 and r_2, which can be used to lower FLOP consumption. For $\varphi = \psi = \beta$, we set $r_1 = r_2 = 4$, which yields comparable accuracy to $\varphi = \psi \neq \beta$ but at higher FLOP counts.

Method		top-1	top-5	Params	Flops
Conv.-ResNet26		76.0	92.8	13.7M	2.4G
SAN10-pair.	$\varphi = \psi = \beta$	76.5	92.8	9.5M	3.0G
	$\varphi = \psi \neq \beta$	76.3	92.6	10.0M	2.1G
	$\varphi \neq \psi \neq \beta$	77.4	93.3	10.5M	2.2G
SAN10-patch.	$\varphi = \psi = \beta$	78.9	94.1	13.4M	2.2G
	$\varphi = \psi \neq \beta$	79.0	94.0	11.3M	1.8G
	$\varphi \neq \psi \neq \beta$	79.3	94.2	11.8M	1.9G

Table 6. Controlled evaluation of the use of distinct transformation functions.

Footprint size. We now assess the impact of the size of the footprint $\mathcal{R}(i)$ of the self-attention operator. The results are reported in Table 7. In convolutional networks, larger footprint sizes incur significant memory and computation costs. In self-attention networks, the accuracy initially increases with footprint size and then saturates. For pairwise self-attention, increasing the footprint size has no impact on the number of parameters. Taking all factors into account, we set the footprint size to 7×7 as our default for all models.

Position encoding. Finally, we evaluate the importance of position encoding in pairwise self-attention. The results are reported in Table 8. Position encoding has a significant effect. Without position encoding, top-1 accuracy drops by 5 percentage points. Absolute position encoding [20] is better than none, but accuracy is still low. Relative position encoding, as described in Section 3.1, is much more effective.

Method		top-1	top-5	Params	Flops
Conv.-ResNet26	3×3	76.0	92.8	13.7M	2.4G
	5×5	77.4	93.6	22.7M	4.0G
	7×7	77.9	93.7	36.1M	6.5G
SAN10-pair.	3×3	75.3	92.0	10.5M	1.7G
	5×5	76.6	92.9	10.5M	1.9G
	7×7	77.4	93.3	10.5M	2.2G
	9×9	77.8	93.5	10.5M	2.5G
	11×11	77.6	93.3	10.5M	3.0G
SAN10-patch.	3×3	77.4	93.4	10.7M	1.6G
	5×5	78.7	94.0	11.2M	1.7G
	7×7	79.3	94.2	11.8M	1.9G
	9×9	79.3	94.1	12.7M	2.1G
	11×11	79.4	94.1	13.8M	2.3G

Table 7. Controlled assessment of the impact of footprint size.

Method		top-1	top-5	Params	Flops
Conv.-ResNet26		76.0	92.8	13.7M	2.4G
SAN10-pair.	none	72.3	90.3	10.5M	2.1G
	absolute	74.7	91.7	10.5M	2.2G
	relative	77.4	93.3	10.5M	2.2G

Table 8. The importance of position encoding in pairwise self-attention.

5.4. Robustness

We now conduct two experiments that probe the robustness of the representations learned by self-attention networks, as compared to convolutional baselines.

Zero-shot generalization to rotated images. The first experiment tests trained networks on rotated and flipped images. In this experiment, ImageNet images from the val-original set are rotated and flipped in one of four ways: clockwise 90°, clockwise 180°, clockwise 270°, and upside-down flip about the horizontal axis. This is zero-shot testing: such manipulations were not performed at training time.

The results are reported in Table 9. Our hypothesis was that pairwise self-attention models will be more robust to this kind of manipulation than convolutional networks (or patchwise self-attention), given that pairwise self-attention is fundamentally a set operator. Indeed, we see that pairwise self-attention models are less vulnerable than convolutional or patchwise self-attention networks, although all networks suffer from the domain shift. For example, when images are rotated by 180°, the performance of pairwise SAN19 drops by 18.9 percentage points, which is 5.1 percentage points lower than the drop suffered by ResNet50. The pairwise SAN10 model achieves 54.7% top-1 accuracy in this

Method	no rotation		clockwise 90°		clockwise 180°		clockwise 270°		upside-down	
	top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5
ResNet26	73.6	91.7	49.1(24.5)	72.7(19.0)	50.6(23.0)	75.4(16.3)	49.2(24.4)	72.8(18.9)	50.5(23.1)	75.4(16.3)
SAN10-pair.	74.9	92.1	51.8(**23.1**)	74.6(**17.5**)	54.7(**20.2**)	78.5(**13.6**)	51.7(23.2)	74.5(**17.6**)	54.7(**20.2**)	78.5(**13.6**)
SAN10-patch.	77.1	93.5	53.1(24.0)	75.7(17.8)	54.6(22.5)	78.4(15.1)	53.3(23.8)	76.0(17.5)	54.7(22.4)	78.3(15.2)
ResNet38	76.0	93.0	51.2(24.8)	74.2(18.8)	52.2(23.8)	76.9(16.1)	51.6(24.4)	74.6(18.4)	52.2(23.8)	76.8(16.2)
SAN15-pair.	76.6	93.1	54.5(**22.1**)	77.1(**16.0**)	57.9(**18.7**)	80.8(**12.3**)	54.8(**21.8**)	77.0(**16.1**)	58.0(**18.6**)	80.8(**12.3**)
SAN15-patch.	78.0	93.9	53.7(24.5)	76.1(17.8)	56.0(22.2)	79.5(14.4)	53.9(24.3)	76.2(17.7)	56.0(22.2)	79.4(14.5)
ResNet50	76.9	93.5	52.6(24.3)	75.3(18.2)	52.9(24.0)	77.4(16.2)	52.6(24.3)	75.5(18.0)	53.0(23.9)	77.3(16.2)
SAN19-pair.	76.9	93.4	54.7(**22.2**)	77.1(**16.3**)	58.0(**18.9**)	80.4(**13.0**)	55.0(**21.9**)	77.1(**16.3**)	57.9(**19.0**)	80.4(**13.0**)
SAN19-patch.	78.2	93.9	54.2(24.0)	76.3(17.6)	56.2(22.0)	79.5(14.4)	54.1(24.1)	76.4(17.5)	56.3(21.9)	79.5(14.4)

Table 9. Robustness of trained networks to rotation and flipping of images at test time. Zero-shot testing on the val-original set. Numbers in the brackets show the relative performance drop compared to testing on original images with no manipulation (lower is better). Pairwise self-attention models are less vulnerable than convolutional networks or patchwise self-attention.

regime, which is higher than the accuracy of the much larger ResNet50 (52.9%).

Robustness to adversarial attacks. Next, we evaluate the robustness of trained networks to adversarial attacks. We subject the trained models to white-box targeted PGD attacks [24]. Hyperparameters of the attacks include the maximal per-pixel perturbation ϵ (under the L_∞ norm), attack step size ρ, and the number of attack iterations n. We test with two sets of hyperparameters: $\{\epsilon, \rho, n\}$ set to {8, 4, 2} and {8, 2, 4}, respectively. The results are reported in Table 10.

The results indicate that self-attention models are much more robust than convolutional networks. For example, with 4 attack iterations, the attack success rate for ResNet50 is 82.5% and top-1 accuracy drops to 11.8%. For the corresponding pairwise and patchwise SAN models, the attack success rate is much lower, at 63.7% and 62.0%, respectively, and the models' accuracy is roughly 2x higher, at 21.8% and 24.8%, respectively. For the ResNet26 baseline, 4 attack iterations essentially destroy the model, with a top-1 accuracy of 1%. In comparison, the top-1 accuracy of the patchwise SAN model is roughly 10x higher at 9.6%. (A random guess baseline would exhibit a top-1 accuracy of 0.1%.)

Both experiments indicate that self-attention networks may have significant benefits in terms of robustness and generalization. These may surpass accuracy gains observed in traditional evaluation procedures and merit further study.

Method	clean	attack $n = 2$		attack $n = 4$	
	top-1	s. rate	top-1	s. rate	top-1
ResNet26	73.6	49.0	26.6	98.2	1.0
SAN10-pair.	74.9	32.8	35.3	90.1	5.3
SAN10-patch.	77.1	24.5	46.4	85.8	9.6
ResNet38	76.0	32.7	39.2	94.1	3.8
SAN15-pair.	76.6	15.5	47.3	67.5	19.6
SAN15-patch.	78.0	13.1	54.8	65.6	22.9
ResNet50	76.9	19.5	49.3	82.5	11.8
SAN19-pair.	76.9	13.1	49.1	63.7	21.8
SAN19-patch.	78.2	12.1	55.1	62.0	24.8

Table 10. Robustness of trained networks to adversarial attacks on the val-original set. n is the number of attack iterations. 's. rate' is the success rate of the attack (lower is better) and 'top-1' is the accuracy under the attack (higher is better). Self-attention models are much more robust than convolutional networks.

6. Conclusion

In this paper, we explored the effectiveness of image recognition models that are based fully on self-attention. We considered two forms of self-attention: pairwise and patchwise. The pairwise form is a set operation and is fundamentally different from convolution in this respect. The patchwise form is a generalization of convolution. For both forms, we introduced *vector* attention that efficiently adapts weights across both spatial dimensions and channels.

Our experiments yield a number of significant findings. First, networks based purely on pairwise self-attention match or outperform convolutional baselines. This indicates that the success of deep learning in computer vision is not inextricably tied to convolutional networks: there is an alternative route to comparable or higher discriminative power, with different and potentially beneficial structural properties such as permutation- and cardinality-invariance. Our second major finding is that patchwise self-attention models substantially outperform convolutional baselines. This suggests that patchwise self-attention, which generalizes convolution, may yield strong accuracy gains across applications in computer vision. Finally, our experiments indicate that *vector* self-attention is particularly powerful and substantially outperforms scalar (dot-product) attention, which has been the predominant formulation to date.

References

[1] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *ICCV*, 2019. 2

[2] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019. 2

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. *ICLR*, 2015. 2

[4] Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, Wei Liu, and Tat-Seng Chua. SCA-CNN: Spatial and channel-wise attention in convolutional networks for image captioning. In *CVPR*, 2017. 2

[5] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017. 2

[6] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *ACL*, 2019. 1, 2

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019. 1, 2

[8] Jun Fu, Jing Liu, Haijie Tian, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *CVPR*, 2019. 2

[9] Ross Girshick. Fast R-CNN. In *ICCV*, 2015. 2

[10] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2

[11] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv:1706.02677*, 2017. 6

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 4, 6

[13] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *ICCV*, 2019. 1, 2, 5

[14] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Andrea Vedaldi. Gather-excite: Exploiting feature context in convolutional neural networks. In *NeurIPS*, 2018. 2

[15] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 1, 2

[16] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017. 1, 2

[17] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *NIPS*, 2016. 2

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[19] Yann LeCun, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1989. 1, 2, 5

[20] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the CoordConv solution. In *NeurIPS*, 2018. 7

[21] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *ECCV*, 2016. 2

[22] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2

[23] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 5

[24] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018. 8

[25] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted Boltzmann machines. In *ICML*, 2010. 5

[26] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *ICML*, 2018. 2

[27] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, 2019. 1, 2, 5, 6

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. ImageNet large scale visual recognition challenge. *IJCV*, 2015. 1, 2, 5

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 2

[31] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1, 2, 5

[32] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 6

[33] 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. 1, 2, 5

[34] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *CVPR*, 2017. 2

[35] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 2, 5

[36] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: Convolutional block attention module. In *ECCV*, 2018. 2

[37] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In *ICLR*, 2019. 2

[38] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 6

[39] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015. 2

[40] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019. 1, 2

[41] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. Stacked attention networks for image question answering. In *CVPR*, 2016. 2

[42] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *ICLR*, 2016. 2

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

[44] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 2

[45] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. PSANet: Point-wise spatial attention network for scene parsing. In *ECCV*, 2018. 2

[46] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable ConvNets v2: More deformable, better results. In *CVPR*, 2019. 2