AFFINE SELF CONVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Attention mechanisms, and most prominently self-attention, are a powerful building block for processing not only text but also images. These provide a parameter efficient method for aggregating inputs. We focus on self-attention in vision models, and we combine it with convolution, which as far as we know, are the first to do. What emerges is a convolution with data dependent filters. We call this an Affine Self Convolution. While this is applied differently at each spatial location, we show that it is translation equivariant. We also modify the Squeeze and Excitation variant of attention, extending both variants of attention to the roto-translation group. We evaluate these new models on CIFAR10 and CIFAR100 and show an improvement in the number of parameters, while reaching comparable or higher accuracy at test time against self-trained baselines.

1 INTRODUCTION

Computer vision has seen great success thanks to the use of the convolution operation in Convolutional Neural Networks (CNNs) (LeCun et al., 1989; Krizhevsky et al., 2012; He et al., 2017; Mnih et al., 2013). This operation takes advantage of the translational symmetry in visual perception tasks such as image classification. Meanwhile, in tasks that require sequence processing, attention (Chorowski et al., 2015; Bahdanau et al., 2014) and self-attention (Vaswani et al., 2017) have emerged as a powerful technique.

One of the peculiarities of CNNs is that filters are defined independently of the data. At the same time, self-attention is data dependent, but does not provide a template matching scheme, as does the convolution operation, since it merely reweights neighborhoods. While there is work towards using attention in CNNs, the current models use them independently, sequentially, or in parallel.

We unify convolution and self-attention, taking the best of both worlds. Our method provides a translationally equivariant convolution, where the filters are also dependent on the input. These data dependent filters more efficiently describe the relations present in the input. Moreover, by formulating it as a special convolution, it can be extended to be equivariant to other groups of transformations. As a result, we apply the rich literature on group equivariant neural networks (Cohen & Welling, 2016) and develop the roto-translation equivariant counterpart. This module can be used as a replacement for standard convolutional layers and we call it an Affine Self Convolution.

Another variant of attention in computer vision is Squeeze and Excitation (Hu et al., 2018). This provides global attention and we extend it to the roto-translation variant in order to compare it to our module. We plan to release code for all the experiments soon.

The contributions of this work are:

- We introduce the Affine Self Convolution (ASC), merging convolution and self-attention.
- We prove ASC is translation equivariant.
- We extend ASC to roto-translation equivariant ASC.
- We develop group Squeeze and Excitation.
- We evaluate these modules on CIFAR10 and CIFAR100.

2 BACKGROUND

In order to combine convolution and self-attention we first look at the group convolutions (Cohen et al., 2018a) and then at the self-attention mechanism (Vaswani et al., 2017; Parmar et al., 2018).

2.1 GROUP CONVOLUTION

Group equivariant convolutional neural networks extend the operation of convolution. Cohen & Welling (2016) show that we can consider the convolution operation to be defined on a group and that this allows for a natural generalization to other objects that have a group structure.

Translation equivariant convolution The set of points in $\mathbb{Z}^2$ with the operation of vector addition forms a group. For each value x in this group, the translation operator $\mathcal{L}_{(x)}$ translates the domain of a function: $\mathcal{L}_{(x)}[f](y) = f(-x + y)$. Therefore, given two functions $f, \psi : \mathbb{Z}^2 \to \mathbb{R}$, the convolution between the image f and the filter ψ can be written in terms of translations by elements of the group $(\mathbb{Z}^2, +)$ (we depict the convolution in Figure 1):

$$[f \star_{\mathbb{Z}^2} \psi](x) = \sum_{y \in \mathbb{Z}^2} f(y)\mathcal{L}_{(x)}[\psi](y) \tag{1}$$

Where the convolution operation $\star_{\mathbb{Z}^2}$ is indexed by the domain of the input and the filter. Most importantly, the convolution is equivariant to translations: $\mathcal{L}_{(z)}[f] \star_{\mathbb{Z}^2} \psi = \mathcal{L}_{(z)}[f \star_{\mathbb{Z}^2} \psi]$. This property connects a transformation of the input image $\mathcal{L}_{(z)}[f]$ with a precise transformation of the activations $\mathcal{L}_{(z)}[f \star_{\mathbb{Z}^2} \psi]$. This is desirable because a model that implements such an operation can also be invariant to translations by taking a max over spatial positions at the end.

Roto-translation equivariant convolution A natural extension of the group of translations is the group of planar rotations and translations (Weiler et al., 2018b; Diaconu & Worrall, 2019). The elements of this group have 2 components, a (proper) rotation R and a translation y:

$$SE(2) = \{(R, y) \in (R \in \mathbb{R}^{2\times 2}, y \in \mathbb{R}^2) | R^\top R = RR^\top = I, \det(R) = 1)\} \tag{2}$$

Where I is the identity matrix of order 2 and $(I, 0)$ is the identity element of $SE(2)$. Throughout this work we will use P, R, S to denote rotation elements and x, y, z translation elements of the group $SE(2)$. From here, we denote the $G = SE(2)$. Similarly to Diaconu & Worrall (2019), the operator $\mathcal{L}_{(P,x)}$ inversely transforms the coordinates of a function f with domain $\mathbb{R}^2$ by $\mathcal{L}_{(P,x)}[f](y) = f(P^{-1}(y - x))$. As a result, we can evaluate the planar convolution at points on G, $[f \star_{\mathbb{R}^2} \psi](P, x)$. This operation is called a lifting layer and outputs activations with domain G. The operator $\mathcal{L}_{(P,x)}$ transforms such functions by $\mathcal{L}_{(P,x)}[f](R, y) = f(P^{-1}R, P^{-1}(y-x))$. We can preserve G equivariance of these functions by replacing the planar convolution with G convolutions between $f, \psi : G \to \mathbb{R}$:

$$[f \star_G \psi](P, x) = \sum_{(R,y) \in G} f(R, y)\mathcal{L}_{(P,x)}[\psi](R, y) \tag{3}$$

In practice we are limited by the sampling grid, as real world images are functions on $\mathbb{Z}^2$. This has as symmetries translations by integer values, which form the group $\mathbb{Z}^2$ and rotations by multiples of $90°$, which form the group C_4. By replacing the groups $\mathbb{R}^2$ and $SO(2)$ in the definition of $SE(2)$ with $\mathbb{Z}^2$ and C_4 we get the group $p4$. This is a subgroup of $SE(2)$, and all the relations in this work hold for both.

2.2 SELF-ATTENTION IN COMPUTER VISION

Simplified form We start with a simplified form of local self-attention to highlight each part and to make notation clearer. For an input image $f : \mathbb{Z}^2 \to \mathbb{R}$, self-attention is defined by 2 parts: 1) a score function between center pixel $f(x)$ and a neighbor pixel $f(y)$. This is usually the dot product, $\text{score}(f(x), f(y)) = f(x)f(y)$. Moreover, such that the score values in a neighborhood sum to 1, the scores are normalized with softmax, $\alpha(x, y) = \text{softmax}_y(\text{score}(f(x), f(y))$, 2) an aggregation of the neighbors y based on the score function:

$$\text{Attention}[f](x) = \sum_{y \in \mathbb{Z}^2} \alpha(x, y)f(y) \tag{4}$$

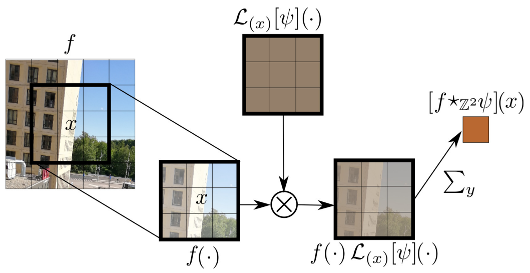
f
x
$\mathcal{L}_{(x)}[\psi](\cdot)$
$[f \star_{\mathbb{Z}^2} \psi](x)$
$\sum_y$
$f(\cdot)$
$f(\cdot)\,\mathcal{L}_{(x)}[\psi](\cdot)$

Figure 1: Convolution between an input f and a 3×3 filter ψ evaluated at x. We denote with $(\cdot)$ all the positions y in the neighborhood of x. This begins by cropping the 3×3 neighborhood in f centered at x. The filter is colored brown for clarity, but it can have different values. The filter is centered in the same neighborhood by translating it $\mathcal{L}_{(x)}[\psi](\cdot)$. The two quantities are then multiplied at each spatial position y, independently. Finally, the sum over the spatial positions aggregates the response of the convolution, concluding how similar the input neighborhood around x is to the filter.

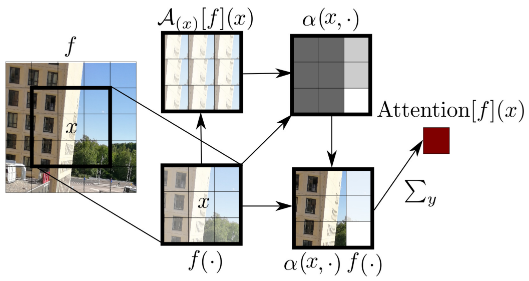

Figure 2: Simple self-attention with extent 3×3 for an input f evaluated at x. Similarly to the convolution $(\cdot)$ denotes the neighborhood of x. The 3×3 region around x is cropped. The score function then compares the center with each neighbor. The normalized score is $\alpha(x, y)$. The scores and the input image neighborhood are then multiplied at each spatial position. This is shown over $\alpha(x, \cdot)f(\cdot)$ by sharpening the color on the building and whitening or removing the color from the sky or trees. By summing over the spatial dimension, we aggregate the neighborhood scaled by how similar the center is to its neighbors.

This operation defines a neighborhood dependent weighting. We depict this in Figure 2. The dot product score function, and therefore Equation 4, does not take account of the relative position between x and y. Simply, this encodes no spatial information. To take advantage of the regular grid in images, recent methods use positional embeddings $\beta : \mathbb{Z}^2 \to \mathbb{R}$. These can be added to neighbors at y, prior to computing the score, and is parametrized by the relative position from the center x, as done in Ramachandran et al. (2019), $\alpha(x, y) = \text{softmax}_y(\text{score}(f(x), f(y) + \beta(y - x)))$. It can also be added to the neighbors score, after computing the score function, as done in Bello et al. (2019); Hu et al. (2019), $\alpha(x, y) = \text{softmax}_y(\text{score}(f(x), f(y)) + \beta(y - x))$

Self-attention In practice, self-attention uses three sets of parameters (Parmar et al., 2018; Wang et al., 2018), W^Q, W^K, W^V, where Q, K, V stands for Query, Key, Value. We turn our attention to an input function (image) $f : \mathbb{Z}^2 \to \mathbb{R}^{d_{\text{in}}}$, with d_{in} input channels and parameters $W^Q, W^K : \mathbb{Z}^2 \to \mathbb{R}^{d_k \times d_{\text{in}}}$ and $W^V : \mathbb{Z}^2 \to \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$. W^Q, W^K, W^V, which are implemented as 1×1 convolutions, have the purpose of mapping to three separate embeddings, Q, K, V and not to process spatial information. These have d_k or d_{out} channels. The spatial information is weighted and mixed by the attention mechanism. In this more general setting, where we denote an arbitrary channel with c, self-attention is defined with: 1) three linear mappings of the input f (defined analogously for K and V): $Q_c(x) = [f \star_{\mathbb{Z}^2} W^Q_c](x)$ 2) a normalized score function, which can use positional embeddings: $\alpha(x, y) = \text{softmax}_y(\text{score}(Q(x), K(y)))$ and 3) an aggregation of the V embeddings based on the score function:

$$\text{Attention}[Q, K, V]_c(x) = \sum_{y \in \mathbb{Z}^2} \alpha(x, y) V_c(y) \tag{5}$$

It is also common to use the multi head mechanism from Transformer (Vaswani et al., 2017) alongside attention. This means that Q, K, V are split along the channel dimension and the self-attention mechanism is evaluated independently for each element of the partition (each head). These are then concatenated and passed through a linear layer.

3 Method

In the following section we look at the local relative type of attention, self-attention and we formulate it similarly to convolution. This allows us to merge the two and then develop the roto-translational

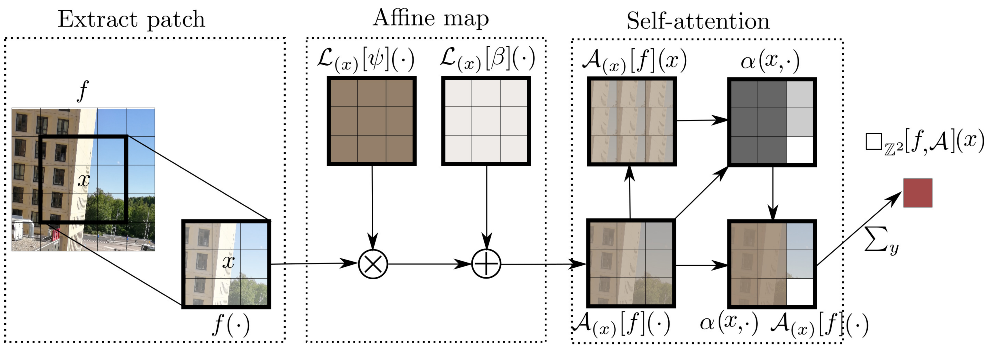
Extract patch
Affine map
Self-attention
f
x
$f(\cdot)$
$\mathcal{L}_{(x)}[\psi](\cdot)$
$\mathcal{L}_{(x)}[\beta](\cdot)$
$\mathcal{A}_{(x)}[f](x)$
$\alpha(x,\cdot)$
$\mathcal{A}_{(x)}[f](\cdot)$
$\alpha(x,\cdot)\ \mathcal{A}_{(x)}[f](\cdot)$
$\Box_{\mathbb{Z}^2}[f,\mathcal{A}](x)$
$\sum_y$

Figure 3: Simple ASC for an input f and an affine map $\mathcal{A}$ with extent 3×3 evaluated at x. As for Figures 1 and 2, $(\cdot)$ denotes the neighborhood of x. The neighborhood is first extracted from the input, then the affine map parameters are centered in the neighborhood of x. For each spatial position, f is transformed affinely by $\mathcal{L}_{(x)}[\psi](\cdot)$ and $\mathcal{L}_{(x)}[\beta](\cdot)$, which results in $\mathcal{A}_{(x)}[f](\cdot)$. If we were to sum now over spatial positions, this would be an affine convolution. Nonetheless, the affinely transformed input is now passed through the self-attention mechanism. This consists of evaluating a similarity score between the center of the neighborhood and the neighbors $\alpha(x, y)$ and then multiplying the input with the score function. As a result, the ASC module can focus on the parts of the image that contain information relevant to the center. Similarly to both previous methods, the neighbors' information is aggregated as the final step. Intuitively, what we get is a data dependent convolution between for the image around x.

variant. We also derive the roto-translational variant of the Squeeze and Excite module, which defines global attention.

3.1 AFFINE SELF CONVOLUTION (ASC)

Affine map We have seen how simplified self-attention in Equation 4 is applied to an input image and that it was further developed using relative positional embeddings β in order to include spatial information. We note that these are additive terms and extend them by a multiplicative term ψ that is also spatially dependent. This results in a affine map $\mathcal{A}$, which depends on $\beta, \psi : \mathbb{Z}^2 \to \mathbb{R}$. We define the map $\mathcal{A}$ relative to a center x: $\mathcal{A}_{(x)} : (\mathbb{Z}^2, \mathbb{R}) \to \mathbb{R}$ and we apply it using the translation operator $\mathcal{L}_{(x)}$ as $\mathcal{A}_{(x)}(y, r) = r\mathcal{L}_{(x)}[\psi](y) + \mathcal{L}_{(x)}[\beta](y)$. Furthermore, we can describe the affine map as acting on an image $f : \mathbb{Z}^2 \to \mathbb{R}$:

$$\mathcal{A}_{(x)}[f](y) \coloneqq \mathcal{A}_{(x)}(y, f(y)) = f(y)\mathcal{L}_{(x)}[\psi](y) + \mathcal{L}_{(x)}[\beta](y) \tag{6}$$

Simplified form Applying the affine map $\mathcal{A}$, then simplified self-attention from Equation 4 to f: Attention$[\mathcal{A}_{(x)}[f]](x) = \sum_y \alpha(x, y)\mathcal{A}_{(x)}[f](y)$ is how we define the simplified ASC. We denote this with $\Box_{\mathbb{Z}^2}$. This is applied to two functions, an input and an affine filter, and we index it by the domain of the two functions, similarly to the convolution operation:

$$\Box_{\mathbb{Z}^2}[f, \mathcal{A}](x) \coloneqq \sum_y \alpha(x, y)\mathcal{A}_{(x)}[f](y) = \sum_{y\in\mathbb{Z}^2} \underbrace{\alpha(x, y)}_{\text{score}}(f(y)\underbrace{\mathcal{L}_{(x)}[\psi](y)}_{\text{filter}} + \underbrace{\mathcal{L}_{(x)}[\beta](y)}_{\text{positional emb}}) \tag{7}$$

This uses the self-attention score from Equation 4, the convolutional filter from Equation 1 and also adds positional embeddings. Moreover, we can distribute α and view $\alpha(x, y)\mathcal{L}_{(x)}[\psi](y)$ and $\alpha(x, y)\mathcal{L}_{(x)}[\beta](y)$ as the parameters of a normalized affine map. Intuitively, this not only performs template matching through ψ, which is independent of the information in the image, but scales the template relative to what is in the image through $\alpha(x, y)$. By unifying the convolution and self-attention, these data dependent filters can more efficiently describe the relations in the image because they are applied differently at each location. We call this an Affine Self Convolution and we depict it in Figure 3. It is possible to recover the usual convolution by setting the scaling coefficients $\alpha(x, y)$ to 1 and β to 0. By setting ψ to 1, we recover a simplified self-attention, where positional embeddings are used for computing the score and for the aggregated term.

Translation equivariance We prove that this is translation equivariant in the Appendix 21 and therefore: $\square_{\mathbb{Z}^2}[\mathcal{L}_{(z)}[f], \mathcal{A}](x) = \mathcal{L}_{(z)}[\square_{\mathbb{Z}^2}[f, \mathcal{A}]](x)$. This means that this model can detect objects regardless of their position in the image, as does the standard convolution.

Affine Self Convolution Similarly to the general form of self-attention in Equation 5, we use three sets of parameters, $(W^Q, \mathcal{A}^Q)$, $(W^K, \mathcal{A}^K)$, $(W^V, \mathcal{A}^V)$ for an input $f : \mathbb{Z}^2 \to \mathbb{R}^{d_{\text{in}}}$. This is depicted in Appendix Figure 5. By contrast to the simple variant, we now also have the affine maps. These process spatial information and are 0 outside the extent, which is controlled by a hyperparameter: kernel size. Moreover, each affine map, can be implemented with $\psi^Q, \psi^K : \mathbb{Z}^2 \to \mathbb{R}^{d_k \times d_k}$, mapping from all the channels in the input to all the channels in the output or $\psi^Q, \psi^K : \mathbb{Z}^2 \to \mathbb{R}^{d_k}$, mapping from one channel in the input to the same channel in the output. The same is true for ψ^V, if we replace d_k with d_{out}. Our experiments use the latter, due to computational constraints. The additive term of the map is defined as $\beta^Q, \beta^K : \mathbb{Z}^2 \to \mathbb{R}^{d_k}$, where for β^V, d_k is replaced with d_{out}. We denote an arbitrary channel with c and we define ASC as:

1) three linear mappings of the input f (defined analogously for K and V):

$$Q_c(x) = [f \star_{\mathbb{Z}^2} W_c^Q](x) \tag{8}$$

2) three affine maps $\mathcal{A}^Q, \mathcal{A}^K, \mathcal{A}^V$ for the Q, K, V terms (defined analogously for K and V):

$$\mathcal{A}^Q_{(x)}[Q]_c(y) = Q_c(y)\mathcal{L}_{(x)}[\psi^Q]_c(y) + \mathcal{L}_{(x)}[\beta^Q]_c(y) \tag{9}$$

3) a score function between center x and neighbor y, which is then normalized with softmax:

$$\alpha(x, y) = \text{softmax}_y(\text{score}(\mathcal{A}^Q_{(x)}[Q](x), \mathcal{A}^K_{(x)}[K](y))) \tag{10}$$

4) an aggregation of the V embeddings based on the score function:

$$\square_{\mathbb{Z}^2}[V, \mathcal{A}^V]_c(x) = \sum_{y \in \mathbb{Z}^2} \alpha(x, y)\mathcal{A}^V_{(x)}[V]_c(y) \tag{11}$$

We also use the multi head mechanism. We note that now we have a separate set of parameters for $\mathcal{A}^Q, \mathcal{A}^K, \mathcal{A}^V$, and that $\mathcal{A}^Q$ is only evaluated at:$\mathcal{A}^Q_{(x)}[Q](x) = Q(x)\psi^Q + \beta^Q$. Therefore, we can learn ψ^Q, β^Q for only one spatial index 0.

By comparison, the positional embeddings in Ramachandran et al. (2019) are represented by the term β^K. The positional embeddings in Hu et al. (2019) are equivalent to learning the product of the embeddings $\beta^Q\beta^K$, which arises when multiplying $\mathcal{A}^Q_x[Q](x)\mathcal{A}^K_x[K](y)$. A difference between this work and their work is that we directly learn these parameters, while Ramachandran et al. (2019); Hu et al. (2019) use a separate network to learn β.

3.2 Roto-translation Affine Self Convolution

We now use the machinery of this new operation and the group theoretic background to develop the roto-translation ASC. Similarly to the standard ASC, we first define a simplified form based on an affine map, then we prove roto-translation equivariance, and finally, we present the general form.

Affine map We now turn to functions on $SE(2)$. We will denote $G := SE(2), H := SO(2), \mathbb{R}^2 = G/H$. Similarly to the relative affine map for functions defined on $\mathbb{Z}^2$ in equation 6, we define a relative affine map $\mathcal{A}$ for functions on G. This map uses affine parameters $\psi, \beta : G \to \mathbb{R}$ and is defined as $\mathcal{A}_{(P,x)} : (G, \mathbb{R}) \to \mathbb{R}$. This acts on a function f with domain G as:

$$\mathcal{A}_{(P,x)}[f](R, y) := f(R, y)\mathcal{L}_{(P,x)}[\psi](R, y) + \mathcal{L}_{(P,x)}[\beta](R, y) \tag{12}$$

This affine map transforms a function f relative to a center (P, x).

Simplified form To define the score function, we notice that we can split the sum over the group G in Equation 3 into two sums, $\sum_{y \in G/H} \sum_{R \in H} \mathcal{A}_{(P,x)}[f](R, y)$. In this form, for each $P \in H$ the convolution can be seen as a weighted sum of the neighbors y, relative to a center x. We can scale the affine group convolution based on this intuition. Precisely, we add a score function that is based on each center (P, x) and each neighbor y:

$$\alpha((P, x), y) = \text{softmax}_y \left(\left(\sum_{R \in H} \mathcal{A}_{(P,x)}[f](R, x) \right) \left(\sum_{R \in H} \mathcal{A}_{(P,x)}[f](R, y) \right) \right) \tag{13}$$

We note that it might also be possible to define a score function $\alpha((P,x),(R,y))$. Nonetheless, this could be computationally prohibitive and is not required in order to preserve roto-translation equivariance. Using the score in Equation 13 we define the Simple ASC on G:

$$\square_G[f,\mathcal{A}](P,x) = \sum_{y \in G/H} \alpha((P,x),y) \sum_{R \in H} \mathcal{A}_{(P,x)}[f](R,y) \tag{14}$$

We show in the Appendix 24 that we can replace $\sum_{R\in H} \beta(R,x)$ with $\beta(x)$. Therefore, we can learn β as a function with domain G/H.

Roto-translation equivariance We verify that ASC on groups is equivariant to actions of the group G, by checking the equivariance relation in the Appendix 25.

Roto-translation ASC In practice, the input is discrete and we turn to functions on $p4$, $G := p4, H := C_4, \mathbb{Z}^2 = G/H$. For the general roto-translation ASC all quantities are defined analogously to the ASC on $\mathbb{Z}^2$ in Equation 11, using the score function in Equation 13, the aggregation in Equation 14 and replacing the domain $\mathbb{Z}^2$ with G. The details can be found in Appendix D.

3.3 Group Squeeze and Excite

Interactions between filters that are spatially far apart is also tackled in Squeeze and Excitation (SE) (Hu et al., 2018). The SE module proposes to rescale each feature map based on a global aggregation of the spatial dimension. An intuition for this is that, by allowing for channel interactions at all the spatial locations, this effectively enlarges the receptive field maximally. This is also a parameter efficient method for increasing the receptive field.

For a general group G (with $g, h, s \in G$), the squeeze term takes an average of a function $f : G \to \mathbb{R}^{d_{\text{in}}}$ over the group. This is invariant to transformations by actions of the group G, which we show in the Appendix 28. The average is then passed through a one hidden layer MLP ($W_1 \in \mathbb{R}^{d_{\text{in}} \times (d_{\text{in}}/r)}, W_2 \in \mathbb{R}^{(d_{\text{in}}/r) \times d_{\text{in}}}$, where d_{in} is the number of channels of the input and r is the reduction ratio). A sigmoid unit (σ) is then used on the activation.

$$\text{squeeze}(f) = \sigma\left(W_1\left(\text{ReLU}\left(W_2\left(\frac{1}{|G|}\sum_{h\in G} f(h)\right)\right)\right)\right) \tag{15}$$

These are then broadcasted across the domain with an element-wise multiplication, $f'_{c_{\text{in}}}(g) = f_{c_{\text{in}}}(g)\text{squeeze}_{c_{\text{in}}}(f)$. Therefore, multiplying a function $f(g)$ by squeeze(f) preserves the group structure of f:

$$\mathcal{L}_{(s)}[f']_{c_{\text{in}}}(g) = \mathcal{L}_{(s)}[f_{c_{\text{in}}}(g)\text{squeeze}_{c_{\text{in}}}(f)] = \mathcal{L}_{(s)}[f]_{c_{\text{in}}}(g)\text{squeeze}_{c_{\text{in}}}(f) \tag{16}$$

This is added as the last operation in any bottleneck ResNet. In our experiments, the group is either the group of integer translations $\mathbb{Z}^2$ (which is the original operation is Hu et al. (2018) and for which $|G|$ is the height$\times$width of the image) or the discrete roto-translation group $p4$ (for which $|G|$ is height$\times$width$\times 4$, since there are 4 rotation in $p4$).

4 Experiments

In this section we describe the dataset used and motivate our baseline architecture. The results are then divided into models that use convolution only and models that use self-attention, including ASC. We then present the overall trends. We specify various hyperparameters in the Appendix C.

Dataset In our experiments we test the models' performance on the CIFAR10 and CIFAR100 datasets (Krizhevsky et al., 2009). These consist of $60k$ images each, $50k$ for training and $10k$ for testing. We further split the training set into $45k$ images for training and we leave $5k$ for validation. CIFAR10 consists of 10 classes and CIFAR100 consists of 100 classes.

Backbone ResNets (He et al., 2016) are a family of CNNs. They are composed of building blocks which are either basic blocks, which have $2n$ layers per feature map size or bottleneck blocks, which have $3n$ layers per feature map size. For CIFAR the feature map sizes are $(32, 16, 8)$. Based on the choice of n He et al. (2016) define the depth of the network. On top of this, they also count the initial and the final layer. As a result we can choose basic block ResNets with $6n + 2$ layers or bottleneck block ResNets with $9n+2$ layers. In our experiments we use a variant of the ResNet that is appropriate for CIFAR and also uses bottleneck residual blocks. We take this approach because the models using self-attention in the literature use it as a replacement for the 3×3 layer inside the bottleneck residual block and we do the same. As a result, from the standard ResNet20, which is an example of a ResNet with basic residual block ($6n + 2$ layers, with $n = 3$), we arrive at ResNet29, which is an example of a ResNet with bottleneck residual block ($9n + 2$ layers, with $n = 3$).

Models We present the results on CIFAR10 in Figure 4. The models are divided into convolution only models and self-attention models, including ASC. Convolution models include squeeze and excite models (SE), since these do not change the convolution operation, but adds the SE module at the end of each bottleneck block. We show the baseline ResNet29, the roto-translational baseline $p4$ResNet29, together with these models with SE. We now turn to models that use self-attention, for which we replace the 3×3 convolutions in ResNet29s bottleneck with self-attention. This means that we still leave a convolution, the convolution in the stem (this is the first layer in all ResNets). We experiment with several models. We replicate the strategy for positional embeddings in Ramachandran et al. (2019) and we only learn a factorized variant of β^K, instead of learning an affine map for each of the Q, K, V terms. These parameters are factorized over the spatial dimensions. This means that for each head, half of the positional embeddings β^K is invariant to horizontal translations, while the other half is invariant to vertical translations. Models using these parametrization of self-attention are denoted with +SASA. Self-attention models includes ASC models. We use Simple_ASC based on Equation 7. This does not use three separate pairs of parameters, just one. With +ASC we denote the general form of ASC as described in equation 11. We also include ResNet29+ASC+SE as one of the models. This is because ASC and SE are not mutually exclusive and we can add the SE module as for a standard ResNet. We also train a roto-translation ASC, which we denote with $p4$ResNet29+ASC. This uses the general form of roto-translation ASC as described in equation 20.

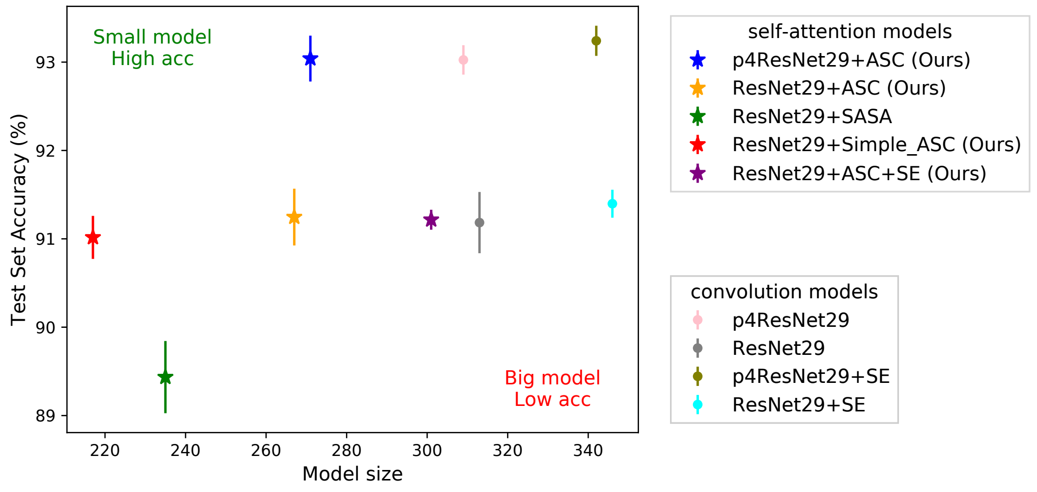

Figure 4: Comparing the model size (in thousands of parameters) to accuracy of several variants of ResNet29 on CIFAR10. Error bars represent 1 standard deviation from 8 runs. Precise numbers are in Appendix Table 2.

Results The overall trade-off between accuracy and parameter count in Figure 4 shows that all the self-attention models use fewer parameters than convolution models, while reaching an accuracy in the same range. This confirms the intuition that data dependent filters are more flexible and a powerful modeling choice. We see that +SE adds a small improvement to convolutional models, but no improvement to +ASC models. This indicates that local self-attention provides enough spatial context. In terms of the self-attention models, +SASA is the only one which does not manage to compete with the baseline. We conclude that a relative affine map is required, additive positional embeddings are not enough. Most insightful, +Simple_ASC drastically decreases the number of parameters ($\approx 30\%$) and reaches accuracy within one standard deviation of the baseline. The three models with the highest accuracy are the roto-translational counterparts to standard models. They

Table 1: We show accuracy mean and 1 standard deviation from 8 runs while varying random seed of various models on CIFAR100. Plot variant in Appendix Figure 4

	Model	Accuracy	#Parameters
translation equivariant	ResNet29	68.34 ± 0.38	$336k$
	ResNet29+Simple_ASC (ours)	68.40 ± 0.78	**240k**
	ResNet29+ASC (ours)	$\mathbf{68.68 \pm 0.77}$	$291k$
roto-translation equivariant	$p4$ResNet29	72.03 ± 0.46	$321k$
	$p4$ResNet29+SE	72.03 ± 0.45	$354k$
	$p4$ResNet29+ASC (ours)	$\mathbf{72.71 \pm 0.51}$	**283k**

show a clear constant increase in performance with little to no extra parameters. This confirms that the theory was applied consistently and that the group theoretic approach benefits attention mechanisms. We ran a subset of the models on Cifar100, results are shown in Table 1. The table shows that both ASC and roto-translation equivariance are more beneficial on this dataset. This indicates that these models generalize better when there is less data available.

The experiments show that self-attention is competitive when including the affine map as done in ASC and that roto-translational equivariance is a robust improvement in all models.

5 Related work

Group equivariant CNNs The theoretically founded approach of group equivariant neural networks has motivated several advances. These works are presented under a unifying framework of group equivariant convolutional networks in Cohen et al. (2018a). Closely related to our work are the developments in planar Euclidean groups in Worrall et al. (2017); Weiler et al. (2018b); Diaconu & Worrall (2019) and the discrete variants of these in LeCun et al. (1989); Cohen & Welling (2016); Dieleman et al. (2016); Hoogeboom et al. (2018) and 3D in Kondor (2018); Cohen et al. (2018b); Esteves et al. (2018a); Worrall & Brostow (2018); Weiler et al. (2018a); Winkels & Cohen (2018); Thomas et al. (2018). We note that this work would benefit from extensions to semigroups (Worrall & Welling, 2019) or curved manifolds (Cohen et al., 2019). Other relevant works include Kondor & Trivedi (2018); Esteves et al. (2019; 2018b); Marcos et al. (2017); Zhou et al. (2017); Bekkers et al. (2018); Jacobsen et al. (2017).

Self-attention Various forms of self-attention for CNNs have been introduced based on non-local means (Buades et al., 2005) or on the Transformer (Vaswani et al., 2017). These models are generally trained for classification/segmentation (Wang et al., 2018; Hu et al., 2019), but have also been used for generative tasks (Parmar et al., 2018; Zhang et al., 2018). Some works (Hu et al., 2019) show that locality of the self-attention mechanism together with softmax helps and that relative positional embeddings are essential. In parallel, Ramachandran et al. (2019) also show improvements using local self-attention with local relative positional embeddings over Bello et al. (2019) which use global self-attention with global positional embeddings. We describe in more detail how our model is related to Ramachandran et al. (2019); Hu et al. (2019) at the end of Section 3.1. It is also worth mentioning that several of these works compare convolutional models with attention based models and show that convolutional models require more floating point operations per second (FLOPS). Self-attention has also been applied to graphs (Veličković et al., 2017). Regardless, they are faster than the attention based models.

Data dependent filters Other works which approach the problem of learning to apply filters differently at each spatial positions are Stanley et al. (2019); Jia et al. (2016); Ha et al. (2016); Jaderberg et al. (2015); Sabour et al. (2017); Kosiorek et al. (2019).

6 Conclusion

In this work we show that there is a mixture of convolution and self-attention that can be used to replace spatial convolutions in CNNs. This module can be described as a convolution with data dependent filters. By retaining all the benefits of self-attention and convolution, what emerges are filters that are translationally equivariant, while being applied differently for each location in the input. The results show that this method is able to achieve comparable if not better performance than the convolutional models, while using fewer parameters and a bigger receptive field. Under simplifying assumptions, we can recover both self-attention and convolution, which allows us to incorporate the group theoretic approach of Group equivariant CNNs. Therefore, we prove the translational equivariance of ASC and we also develop the roto-translation equivariant ASC. The latter, is more robust to transformations of the input while surpassing the other models in accuracy. We expect the most fruitful directions for future work to be: an efficient implementation (because self-attention is slower), efficient parametrization (order and shape of Ws and $\mathcal{A}$s), merge self-attention and convolution for NLP/graphs, and equivariant ASC for manifolds (equivariant self-attention could score transport methods without assuming a predefined geometry).

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Erik J Bekkers, Maxime W Lafarge, Mitko Veta, Koen AJ Eppenhof, Josien PW Pluim, and Remco Duits. Roto-translation covariant convolutional networks for medical image analysis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 440–448. Springer, 2018.

Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. *arXiv preprint arXiv:1904.09925*, 2019.

Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pp. 60–65. IEEE, 2005.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. In *Advances in neural information processing systems*, pp. 577–585, 2015.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, pp. 2990–2999, 2016.

Taco Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant cnns on homogeneous spaces. *arXiv preprint arXiv:1811.02017*, 2018a.

Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral cnn. In *International Conference on Machine Learning*, pp. 1321–1330, 2019.

Taco S. Cohen, Mario Geiger, Jonas Koehler, and Max Welling. Spherical cnns, 2018b.

Nichita Diaconu and Daniel Worrall. Learning to convolve: A generalized weight-tying approach. In *International Conference on Machine Learning*, pp. 1586–1595, 2019.

Sander Dieleman, Jeffrey De Fauw, and Koray Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *International Conference on Machine Learning*, pp. 1889–1898, 2016.

Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 52–68, 2018a.

Carlos Esteves, Avneesh Sud, Zhengyi Luo, Kostas Daniilidis, and Ameesh Makadia. Cross-domain 3d equivariant image embeddings. *arXiv preprint arXiv:1812.02716*, 2018b.

Carlos Esteves, Yinshuang Xu, Christine Allen-Blanchette, and Kostas Daniilidis. Equivariant multi-view networks. *arXiv preprint arXiv:1904.00993*, 2019.

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 preprint arXiv:1706.02677*, 2017.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

Emiel Hoogeboom, Jorn WT Peters, Taco S Cohen, and Max Welling. Hexaconv. *arXiv preprint arXiv:1803.02108*, 2018.

Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. *arXiv preprint arXiv:1904.11491*, 2019.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Yerlan Idelbayev. pytorchresnetcifar10. `https://github.com/akamaster/pytorch_resnet_cifar10`.

Jörn-Henrik Jacobsen, Bert De Brabandere, and Arnold WM Smeulders. Dynamic steerable blocks in deep residual networks. *arXiv preprint arXiv:1706.00598*, 2017.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pp. 2017–2025, 2015.

Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, pp. 667–675, 2016.

Risi Kondor. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials. *arXiv preprint arXiv:1803.01588*, 2018.

Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning*, pp. 2747–2755, 2018.

Adam R Kosiorek, Sara Sabour, Yee Whye Teh, and Geoffrey E Hinton. Stacked capsule autoencoders. *arXiv preprint arXiv:1906.06818*, 2019.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Alex Krizhevsky et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

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*, 1(4):541–551, 1989.

Diego Marcos, Michele Volpi, Nikos Komodakis, and Devis Tuia. Rotation equivariant vector field networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5048–5057, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in neural information processing systems*, pp. 3856–3866, 2017.

Kenneth O Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35, 2019.

Oleg Smery. imgclsmob. `https://github.com/osmr/imgclsmob`.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7794–7803, 2018.

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *Advances in Neural Information Processing Systems*, pp. 10381–10392, 2018a.

Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 849–858, 2018b.

Marysia Winkels and Taco S Cohen. 3d g-cnns for pulmonary nodule detection. *arXiv preprint arXiv:1804.04656*, 2018.

Daniel Worrall and Gabriel Brostow. Cubenet: Equivariance to 3d rotation and translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 567–584, 2018.

Daniel E Worrall and Max Welling. Deep scale-spaces: Equivariance over scale. *arXiv preprint arXiv:1905.11697*, 2019.

Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5028–5037, 2017.

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

Yanzhao Zhou, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Oriented response networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 519–528, 2017.

A QKV ASC PICTURE

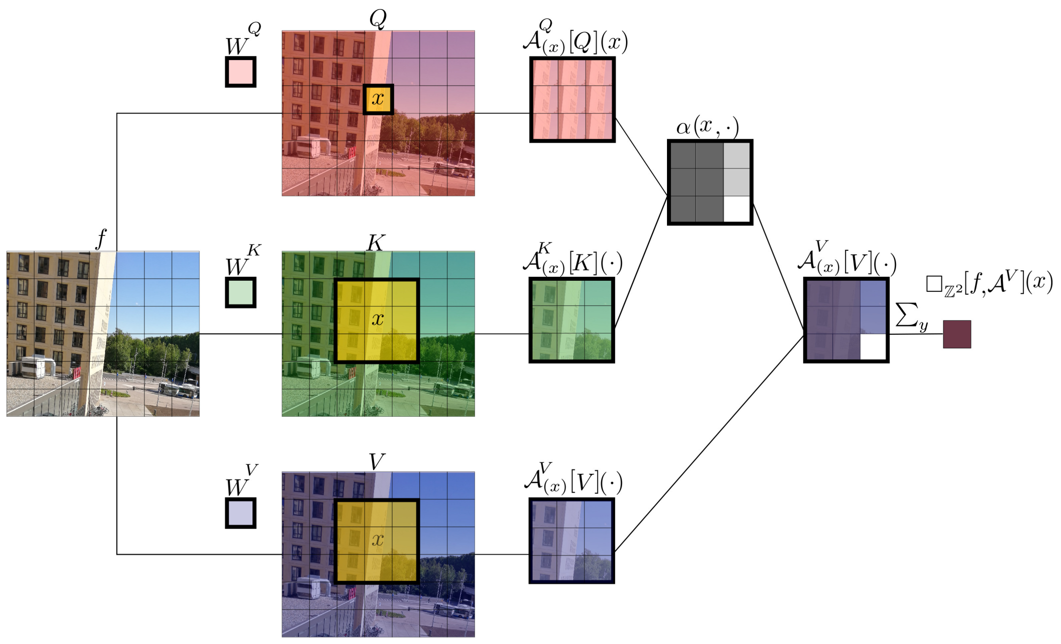
W^Q
Q
x
$\mathcal{A}^Q_{(x)}[Q](x)$
$\alpha(x,\cdot)$
f
W^K
K
x
$\mathcal{A}^K_{(x)}[K](\cdot)$
$\mathcal{A}^V_{(x)}[V](\cdot)$
$\sum_y$
$\square_{\mathbb{Z}^2}[f,\mathcal{A}^V](x)$
W^V
V
x
$\mathcal{A}^V_{(x)}[V](\cdot)$

Figure 5: Affine Self Convolution with extent 3×3 evaluated at x. Yellow depicts the cropped region. For clarity, the affine parameters are not depicted, just the output of the affine map. As opposed to Simple ASC in Figure 3, here we have 3 linear transformations W^Q, W^K, W^V, which do not process spatial information, just output 3 separate embeddings Q, K, V for the same input f. Moreover, three local affine maps $\mathcal{A}^Q, \mathcal{A}^K, \mathcal{A}^V$ are applied to each neighborhood relative to a center x in order to process spatial information. The score $\alpha(x, y)$ is evaluated between the center taken from Q and the neighbors taken from K. The score is then used to aggregate neighbors taken from V. This depicts how self-attention is applied in practice.

B RESULTS

Table 2: We show accuracy mean and 1 standard deviation from 8 runs while varying random seed of various models on CIFAR10. With **bold** we outline the highest accuracy second highest with **blue**. We do the same for the most parameter efficient models. Plot variant in Figure 4.

	Model	Accuracy	#Parameters
convolution models	ResNet29	91.18 ± 0.34	313k
	ResNet29 + SE	91.39 ± 0.15	347k
	$p4$ResNet29	93.02 ± 0.16	310k
	$p4$ResNet29 + SE	**93.24 ± 0.17**	342k
self-attention models	ResNet29+SASA	89.43 ± 0.40	**235k**
	ResNet29+Simple_ASC (ours)	91.01 ± 0.24	**217k**
	ResNet29+ASC (ours)	91.24 ± 0.32	268k
	ResNet29+ASC+SE (ours)	91.21 ± 0.11	301k
	$p4$ResNet29+ASC (ours)	**93.03 ± 0.25**	272k

We also run preliminary experiments with ResNet83 (bottleneck block $9n + 2$, with $n = 9$), which are presented in Figure 7. These show a similar trend to the ResNet29 models, but they also indicate that attention, either as SE or ASC might be more rewarding in bigger models.

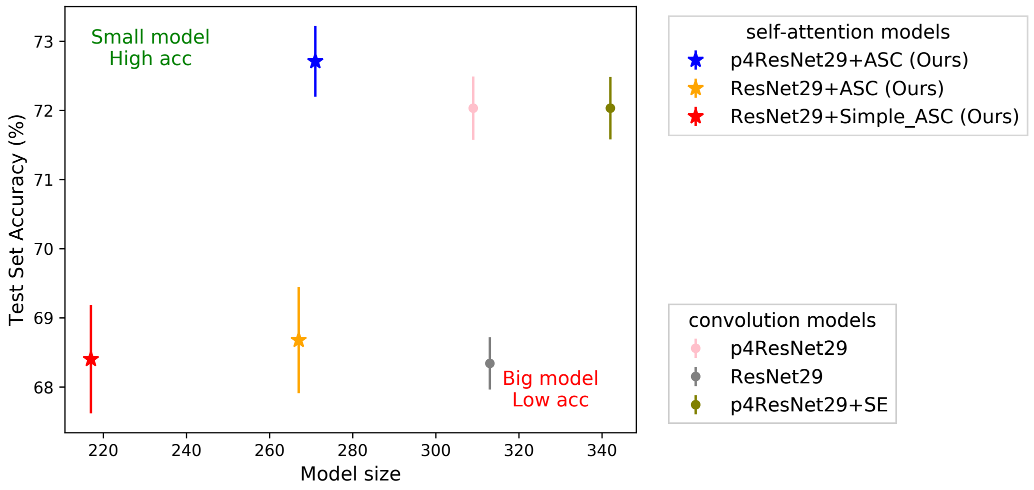

Figure 6: Comparing the model size (in thousands of parameters) to accuracy of several variants of ResNet29 on CIFAR100. Error bars represent 1 standard deviation from 8 runs. Precise numbers are in Table 1.

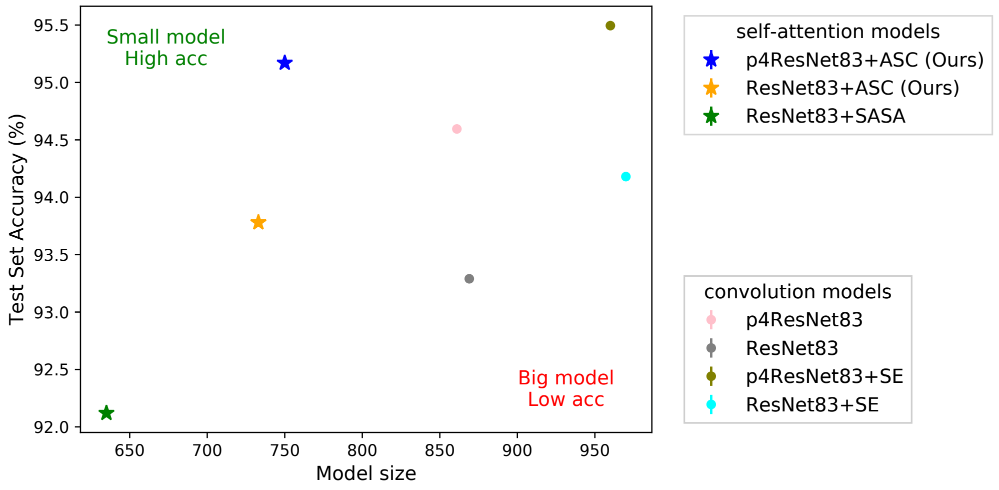

Figure 7: Comparing the parameter efficiency (model size) to accuracy of several variants of ResNet83 on CIFAR10. Model size is divided by 1000. These results are from only 1 run of each model.

C Hyperparameters, initialization and training schedule

We normalize the images and use the standard data augmentation technique of random horizontal flips and random crops of 32×32 from the zero padded 36×36 images.

We initialize convolutional layers using He initialization (He et al., 2015) (for the $p4$ variant, the number of channels is multiplied by 4 in the He initialization) and we initialize batchnorms scaling coefficient γ to 1 and shifting coefficient β to 0. The reduction ratio in SE is 16, while in $p4$ models, the reduction ratio in SE is 4.

In the self-attention models we replace the spatial convolutions in the bottleneck layers with self-attention layers. Where the baseline ResNet uses a stride of 2, the self-attention models applies self-attention then an average pooling layer with kernel size 2×2 and stride 2. The self-attention layers use the multi head mechanism with 8 heads and a kernel size of 5. We set $d_k = d_{\text{out}}$. In all examples we initialize $\psi, \beta \sim \mathcal{N}(0, 1)$. We use the dot product score normalized by $\frac{1}{d_k}$. These models are more unstable in the first couple of epochs. Therefore, we initialize the scaling coefficient γ of the last batchnorm in each residual block to 0 as done in Goyal et al. (2017). Moreover, we warmup (per epoch) the learning rate for the first 10 epochs, up to the learning rate of 0.1. The models are trained using Nesterov accelerated gradient with momentum 0.9 and weight decay of 0.0001. The ResNet29 models and its variants are trained for 100 epochs, where the learning rate was divided by 10 at epochs 50 and 75 and each model was trained for 100 epochs. We also include examples of ResNet83. These bigger models we trained for 200 epochs and divided the learning rate by 10 at epochs 100 and 150. We do this for all the models, for a fair comparison. Throughout our experiments we used Pytorch (Paszke et al., 2017). We have also taken inspiration from the GitHub repositories: Smery; Idelbayev.

When we use roto-translation layers instead of standard layers, we divide the number of channels in each stage of the ResNet by $\sqrt{|p4|} = 2$. This preserves a similar number of parameters between the roto-translation models and the standard models.

D Roto-translation ASC

In practice, the input is discrete and we turn to functions on $p4, G := p4, H := C_4, \mathbb{Z}^2 = G/H$. For the general roto-translation ASC all quantities are defined analogously to the ASC on $\mathbb{Z}^2$ in Equation 11, using the score function in Equation 13 and replacing the domain $\mathbb{Z}^2$ with G. Therefore, we define roto-translation ASC with:

1) three linear mappings of the input f (defined analogously for K and V):

$$Q_c(P,x) = [f \star_G W_c^Q](P,x) \tag{17}$$

2) three affine maps $\mathcal{A}^Q, \mathcal{A}^K, \mathcal{A}^V$ for the Q, K, V terms (defined analogously for K and V):

$$\mathcal{A}^Q_{(P,x)}[Q]_c(R,y) = Q_c(R,y)\mathcal{L}_{(P,x)}\psi_c^Q(R,y) + \mathcal{L}_{(P,x)}\beta_c^Q(y) \tag{18}$$

3) a score function between center (P,x) and neighbor y, which is then normalized with softmax:

$$\alpha((P,x),y) = \text{softmax}_y(\text{score}\left(\left(\sum_{R\in H}\mathcal{A}^Q_{(P,x)}[Q](R,x)\right), \left(\sum_{R\in H}\mathcal{A}^K_{(P,x)}[K](R,y)\right)\right)) \tag{19}$$

4) an aggregation of the V embeddings based on the score function: The final aggregation step completely describes the roto-translation ASC operation:

$$\square_G[V,\mathcal{A}^V]_c(P,x) = \sum_{y\in\mathbb{Z}^2}\alpha((P,x),y)\sum_{R\in H}\mathcal{A}^V_{(P,x)}[V]_c(R,y) \tag{20}$$

Similarly to ASC, for roto-translation ASC we learn ψ^Q, β^Q for only one spatial index 0.

E Proofs

ASC translation equivariance proof

Claim:

$$\square_{\mathbb{Z}^2}[\mathcal{L}_{(z)}[f],\mathcal{A}](x) = \mathcal{L}_{(z)}[\square_{\mathbb{Z}^2}[f,\mathcal{A}]](x) \tag{21}$$

Proof:

Expanding from the left hand side:

$$\square_{\mathbb{Z}^2}[\mathcal{L}_{(z)}[f],\mathcal{A}](x) = \sum_{y\in\mathbb{Z}^2}\text{softmax}_y(\text{score}(\mathcal{A}_{(x)}[\mathcal{L}_{(z)}[f]](x), \mathcal{A}_{(x)}[\mathcal{L}_{(z)}[f]](y))) \\ \mathcal{A}_{(x)}[\mathcal{L}_{(z)}[f]](y)$$

Using the substitution: $y \mapsto y+z$

Using: $y \mapsto y+z \Rightarrow \mathcal{A}_{(x)}[\mathcal{L}_{(z)}[f]](y) = \mathcal{A}_{(-z+x)}[f](y)$, which we prove in the Appendix 22.

Using: $\mathcal{A}_{(x)}[\mathcal{L}_{(z)}[f]](x) = \mathcal{A}_{(-z+x)}[f](-z+x)$, which we prove in the Appendix 23.

$$
\begin{aligned}
&= \sum_{y \in \mathbb{Z}^2} \text{softmax}_y(\text{score}(\mathcal{A}_{(-z+x)}[f](-z+x), \mathcal{A}_{(-z+x)}[f](y))) \\
&\quad \mathcal{A}_{(-z+x)}[f](y) \\
&= \square_{\mathbb{Z}^2}[f, \mathcal{A}](-z+x) \\
&= \mathcal{L}_{(z)}[\square_{\mathbb{Z}^2}[f, \mathcal{A}]](x)
\end{aligned}
$$

Proof

Claim:

$$
y \mapsto y + z \Rightarrow \mathcal{A}_{(x)}[\mathcal{L}_{(z)}[f]](y) = \mathcal{A}_{(-z+x)}[f](y) \tag{22}
$$

Proof:

$$
\begin{aligned}
\mathcal{A}_{(x)}[\mathcal{L}_{(z)}[f]](y) =& \mathcal{L}_{(z)}[f](y)\psi(y-x) + \beta(y-x) \\
=& f(y-z)\psi(y-x) + \beta(y-x)
\end{aligned}
$$

Using the substitution: $y \mapsto y + z$

$$
\begin{aligned}
=& f(y)\psi(y+z-x) + \beta(y+z-x) \\
=& \mathcal{A}_{(-z+x)}[f](y)
\end{aligned}
$$

Proof

Claim:

$$
\mathcal{A}_{(x)}[\mathcal{L}_{(z)}[f]](x) = \mathcal{A}_{(-z+x)}[f](x-z) \tag{23}
$$

Proof:

$$
\begin{aligned}
\mathcal{A}_{(x)}[\mathcal{L}_{(z)}[f]](x) =& \mathcal{L}_{(z)}[f](x)\psi(x-x) + \beta(x-x) \\
=& f(x-z)\psi((x-z)-(x-z)) + \beta((x-z)-(x-z)) \\
=& \mathcal{A}_{(x-z)}[f](x-z)
\end{aligned}
$$

Proof β is a function on G/H

Claim:

$$
\sum_{R \in H} \mathcal{A}_{(P,x)}[f](R,y) = \sum_{R \in H} f(R,y)\mathcal{L}_{(P,x)}[\psi](R,y) + \mathcal{L}_{(P,x)}[\beta](y) \tag{24}
$$

Proof:
For the ASC on groups, the map $\mathcal{A}$ and therefore, α and β, are always used inside $\sum_{R \in H}$. This leads to a more parameter efficient parametrization for β:

$$
\begin{aligned}
\sum_{R \in H} \mathcal{A}_{(P,x)}[f](R,y) &= \sum_{R \in H} f(R,y)\mathcal{L}_{(P,x)}[\psi](R,y) + \mathcal{L}_{(P,x)}[\beta](R,y) \\
&= \sum_{R \in H} f(R,y)\mathcal{L}_{(P,x)}[\psi](R,y) + \sum_{R \in H} \mathcal{L}_{(P,x)}[\beta](R,y) \\
\sum_{R \in H} \mathcal{L}_{(P,x)}[\beta](R,y) &= \sum_{R \in H} \beta(P^{-1}R, P^{-1}(y-x))
\end{aligned}
$$

Using the substitution: $R \mapsto PR$

$$
= \sum_{R \in H} \beta(1, P^{-1}(y-x))
$$

This is actually a function on G/H, not the whole group G. Therefore, we replace $\sum_{R\in H}\beta(R,x)$ with $\beta(x)$.

Roto-translation ASC equivariance proof

Claim:

$$\square_G[\mathcal{L}_{(S,z)}[f],\mathcal{A}](P,x)=\mathcal{L}_{(S,z)}[\square_G[f,\mathcal{A}]](P,x) \tag{25}$$

Proof:

Expanding on the left hand side:

$$\square_G[\mathcal{L}_{(S,z)}[f],\mathcal{A}](P,x)=\sum_{y\in G/H}\text{softmax}_y\left(\left(\sum_{R\in H}\mathcal{A}_{(P,x)}[\mathcal{L}_{(S,z)}[f]](R,x)\right)\left(\sum_{R\in H}\mathcal{A}_{(P,x)}[\mathcal{L}_{(S,z)}[f]](R,y)\right)\right)\left(\sum_{R\in H}\mathcal{A}_{(P,x)}[\mathcal{L}_{(S,z)}[f]](R,y)\right)$$

Using the substitution: $R\mapsto SR$ and $y\mapsto Sy+z$
Using: $R\mapsto SR\Rightarrow\mathcal{A}_{(P,x)}[\mathcal{L}_{(S,z)}[f]](R,x)=\mathcal{A}_{(S,z)^{-1}(P,x)}[f](R,S^{-1}(x-z))$, which we prove in the Appendix 26.
Using: $R\mapsto SR, y\mapsto Sy+z\Rightarrow\mathcal{A}_{(P,x)}[\mathcal{L}_{(S,z)}[f]](R,y)=\mathcal{A}_{(S,z)^{-1}(P,x)}[f](R,y)$, which we prove in the Appendix 27.

$$\begin{aligned}
&=\sum_{y\in G/H}\text{softmax}_y\left(\left(\sum_{R\in H}\mathcal{A}_{(S,z)^{-1}(P,x)}[f](R,S^{-1}(x-z))\right)\left(\sum_{R\in H}\mathcal{A}_{(S,z)^{-1}(P,x)}[f](R,y)\right)\right)\left(\sum_{R\in H}\mathcal{A}_{(S,z)^{-1}(P,x)}[f](R,y)\right)\\
&=\square_G[f,\mathcal{A}](S^{-1}P,S^{-1}(x-z))\\
&=\mathcal{L}_{(S,z)}[\square_G[f,\mathcal{A}]](P,x)
\end{aligned}$$

By arriving at the right hand side of equation 25, we concolude the proof that roto-translation ASC is equivariant to actions of the group $SE(2)$.

Proof

Claim:

$$R\mapsto SR\Rightarrow\mathcal{A}_{(P,x)}[\mathcal{L}_{(S,z)}[f]](R,x)=\mathcal{A}_{(S,z)^{-1}(P,x)}[f](R,S^{-1}(x-z)) \tag{26}$$

Proof:

$$\begin{aligned}
\mathcal{A}_{(P,x)}[\mathcal{L}_{(S,z)}[f]](R,x) =& \mathcal{L}_{(S,z)}[f](R,x)\mathcal{L}_{(P,x)}[\psi](R,x) + \mathcal{L}_{(P,x)}[\beta](x) \\
=& f(S^{-1}R, S^{-1}(x-z))\psi(P^{-1}R, P^{-1}(x-x)) + \beta(P^{-1}(x-x))
\end{aligned}$$

Using the substitution: $R \mapsto SR$

$$\begin{aligned}
=& f(R, S^{-1}(x-z))\psi(P^{-1}SR, P^{-1}(x-x))+ \\
& + \beta(P^{-1}(x-x)) \\
=& f(R, S^{-1}(x-z))\psi(P^{-1}SR, P^{-1}S(S^{-1}(x-z) - S^{-1}(x-z)))+ \\
& + \beta(P^{-1}S(S^{-1}(x-z) - S^{-1}(x-z))) \\
=& f(R, S^{-1}(x-z))\mathcal{L}_{(S^{-1}P, S^{-1}(x-z))}[\psi](R, S^{-1}(x-z))+ \\
& + \mathcal{L}_{(S^{-1}P, S^{-1}(x-z))}[\beta](S^{-1}(x-z))) \\
=& \mathcal{A}_{(S^{-1}P, S^{-1}(x-z))}[f](R, S^{-1}(x-z)) \\
=& \mathcal{A}_{(S,z)^{-1}(P,x)}[f](R, S^{-1}(x-z))
\end{aligned}$$

Proof

Claim:

$$R \mapsto SR, y \mapsto Sy+z \Rightarrow \mathcal{A}_{(P,x)}[\mathcal{L}_{(S,z)}[f]](R,y) = \mathcal{A}_{(S,z)^{-1}(P,x)}[f](R,y) \tag{27}$$

Proof:

$$\begin{aligned}
\mathcal{A}_{(P,x)}[\mathcal{L}_{(S,z)}[f]](R,y) =& \mathcal{L}_{(S,z)}[f](R,y)\mathcal{L}_{(P,x)}[\psi](R,y) + \mathcal{L}_{(P,x)}[\beta](y) \\
=& f(S^{-1}R, S^{-1}(y-z))\psi(P^{-1}R, P^{-1}(y-x)) + \beta(P^{-1}(y-x))
\end{aligned}$$

Using the substitutions: $R \mapsto SR$ and $y \mapsto Sy+z$

$$\begin{aligned}
=& f(R,y)\psi(P^{-1}SR, P^{-1}(Sy+z-x))+ \\
& + \beta(P^{-1}(Sy+z-x)) \\
=& f(R,y)\psi(P^{-1}SR, P^{-1}S(y - S^{-1}(x-z)))+ \\
& + \beta(P^{-1}S(y - S^{-1}(x-z))) \\
=& f(R,y)\mathcal{L}_{(S^{-1}P, S^{-1}(x-z))}[\psi](R,y)+ \\
& + \mathcal{L}_{(S^{-1}P, S^{-1}(x-z))}[\beta](y)) \\
=& \mathcal{A}_{(S^{-1}P, S^{-1}(x-z))}[f](R,y) \\
=& \mathcal{A}_{(S,z)^{-1}(P,x)}[f](R,y)
\end{aligned}$$

Proof Group Squeeze is invariant to transformations of the group

Claim:

$$\text{squeeze}(\mathcal{L}_{(s)}[f]) = \text{squeeze}(f) \tag{28}$$

Proof:

$$\begin{aligned}\text{squeeze}(\mathcal{L}_{(s)}[f]) &= \sigma\left(W_1\left(\text{ReLU}\left(W_2\left(\frac{1}{|G|}\sum_{h\in G}\mathcal{L}_{(s)}[f](h)\right)\right)\right)\right)\\ &= \sigma\left(W_1\left(\text{ReLU}\left(W_2\left(\frac{1}{|G|}\sum_{h\in G}f(s^{-1}h)\right)\right)\right)\right)\end{aligned}$$

Using the substitution: $h \mapsto sh$

$$\begin{aligned}&= \sigma\left(W_1\left(\text{ReLU}\left(W_2\left(\frac{1}{|G|}\sum_{h\in G}f(h)\right)\right)\right)\right)\\ &= \text{squeeze}(f)\end{aligned}$$