Scaling Local Self-Attention for Parameter Efficient Visual Backbones

Ashish Vaswani
Google Research

Prajit Ramachandran
Google Research

Aravind Srinivas
UC Berkeley

Niki Parmar
Google Research

Blake Hechtman
Google Research

Jonathon Shlens
Google Research

Abstract

Self-attention has the promise of improving computer vision systems due to parameter-independent scaling of receptive fields and content-dependent interactions, in contrast to parameter-dependent scaling and content-independent interactions of convolutions. Self-attention models have recently been shown to have encouraging improvements on accuracy-parameter trade-offs compared to baseline convolutional models such as ResNet-50. In this work, we aim to develop self-attention models that can outperform not just the canonical baseline models, but even the high-performing convolutional models. We propose two extensions to self-attention that, in conjunction with a more efficient implementation of self-attention, improve the speed, memory usage, and accuracy of these models. We leverage these improvements to develop a new self-attention model family, HaloNets, *which reach state-of-the-art accuracies on the parameter-limited setting of the ImageNet classification benchmark. In preliminary transfer learning experiments, we find that HaloNet models outperform much larger models and have better inference performance. On harder tasks such as object detection and instance segmentation, our simple local self-attention and convolutional hybrids show improvements over very strong baselines. These results mark another step in demonstrating the efficacy of self-attention models on settings traditionally dominated by convolutional models.*

1. Introduction

Vision and natural language processing (NLP) systems divide the landscape of computational primitives. While self-attention is the primary workhorse in NLP, convolutions are ubiquitous in nearly all vision models. Convolutions embody the principle of *local* processing, to learn local spatial features such as edges and texture that are abundant in images. On the other hand, the Transformer [57] showed that self-attention is an effective and computationally efficient mechanism for capturing *global* interactions between words in a sentence. The success of self-attention in NLP motivates research in understanding how self-attention can improve vision. Self-attention has several properties that make it a good fit for vision: (a) content-based interactions as opposed to content-independent interactions of convolution; (b) parameter-independent scaling of receptive field size as opposed to parameter-dependent scaling of convolution; (c) empirical ability to capture long-range dependencies for use in larger images; (d) flexibility to handle and integrate multiple types of data that appear in vision, such as pixels [59, 3, 44, 66], point clouds [63], sequence conditioning information [62], and graphs [32]. Self-attention may also be regarded as an adaptive nonlinearity paralleling a long history of nonlinear processing techniques in computer vision, such as bilateral filtering [39] and non-local means [4].

Several recent papers [3, 43, 12, 66, 50] have attempted using self-attention primitives to improve image classification accuracy over the strong and commonly used ResNet backbones [17, 18]. Among them, the Stand-Alone Self-Attention (SASA) [43] is a fully self-attentive model that replaces every spatial convolution with *local* self-attention, which improves the performance of ResNet backbones while having fewer parameters and floating point operations. While conceptually promising, these models lag behind state-of-the-art convolutional models in image classification. State-of-the-art convolutional models [55, 67, 42] use a variety of scaling techniques to achieve strong performance across a range of computation and parameter regimes. In this work, we aim to develop and understand techniques for scaling *local* self-attention models to outperform some of the best convolutional models. Scaling self-attention models presents a unique set of challenges. For example, convolutions have been very efficiently mapped to matrix accelerators such as TPUs and GPUs that drive most deep learning workloads, but fast implementations of local 2D self-attention do not currently exist. To bridge this gap, we introduce a *non-centered* version of local attention that efficiently maps to existing hardware with *haloing*. While our formulation breaks *translational equivariance*, it improves both throughput and accuracies over the centered local self-attention used in SASA. We also introduce a strided self-attentive downsampling operation for multi-scale feature extraction.

We leverage these techniques to develop a new local self-attention model family, *HaloNet*, which achieves state-of-the-art performance across different parameter regimes. The largest HaloNet achieves 84.9% top-1 accuracy on the ImageNet [47] classification benchmark (Section 4.1). We perform a detailed study to uncover how self-attention and convolutional models scale differently. Our self-attention layers also show promising results on harder tasks such as object detection and instance segmentation (Section 4.6) using the Mask R-CNN framework on the COCO benchmark. Finally, we end with a discussion of current limitations and ideas for future work in applying self-attention to vision.

2. Models and Methods

Although our models use self-attention instead of convolutions for capturing spatial interactions between pixels, they adopt some important architectural features of modern convolutional neural networks (CNNs). Like CNNs, we compute *multi-scale feature hierarchies* [34] which enable detecting objects at multiple sizes in tasks such as localization and instance segmentation. For this, we develop a strided self-attention layer, a natural extension of strided convolutions (Section 2.2). To deal with the computational cost in larger resolutions where global attention is infeasible, we follow the fairly general principle of *local processing*, which is at the heart of convolutions and natural perceptual systems [25, 26], and use spatially restricted forms of self-attention. However, unlike the model of [43], that also use local self-attention, we abstain from enforcing translation equivariance in lieu of better hardware utilization, which improves the speed-accuracy tradeoff (Section 2.2). Also note that while we use local attention, our receptive fields per pixel are quite large (up to 18×18) and we show in Section 4.2.2 that larger receptive fields help with larger images. In the remainder of this section, we will motivate self-attention for vision tasks and describe how we relax translational equivariance to efficiently map local self-attention to hardware.

2.1. Self-attention can generate spatially varying convolutional filters

Recent work [8] has shown that self-attention with sufficient number of heads and the right geometric biases can simulate convolutions, suggesting a deeper relationship between self-attention and convolutions. Self-attention has been viewed as a method to directly capture relationships between distant pixels [43, 22, 58]. It has also been interpreted as a specific instantiation of the classic technique of non-local means [4, 59]. The perspective that we discuss in this section is one that views self-attention as generating spatially varying filters, in contrast to the reuse of the same filter across every spatial location in standard convolutions [14]. To observe this, we write self-attention and convolution as specific instances of a general spatial pooling function. Given an input $x \in \mathcal{R}^{H \times W \times c_{in}}$, where H is the height, W is the width, and c_{in} is the number of input channels, we define a local 2D pooling function that computes an output at location (i, j), $y_{ij} \in \mathcal{R}^{c_{out}}$ as

$$y_{ij} = \sum_{a,b \in \mathcal{N}(i,j)} f(i, j, a, b) x_{ab},$$

where $f(i, j, a, b)$ is a function that returns a weight matrix $W \in \mathcal{R}^{c_{in} \times c_{out}}$ at every location in a 2D window $\mathcal{N}(i, j)$ of size $k \times k$ centered at (i, j). Note that later in this section, we introduce non-centered windows for self-attention, but we use centering here for ease of explanation. This computation is repeated for every pixel (i, j). For a convolution, $f(i, j, a, b)$ returns a *different linear transformation* for each relative distance in neighborhood, and these weights are shared across all (i, j). Weight sharing significantly reduces parameters and encourages learning features that repeat spatially. In dot-product relative self-attention [48, 43, 3] (eqs. (2) and (3)), every pixel in the neighborhood shares the *same linear transformation* which is multiplied by a scalar probability that is a function of both content-content and content-geometry interactions resulting in weights that can vary spatially. As an example, for a ball and an orange at two different locations in an image, pixels inside the ball and the orange are likely to generate different $p^{ij}_{a-i,b-j}$ because of the different content around them, such as color or texture.

$$f(i, j, a, b)^{conv} = W_{a-i,b-j} \tag{1}$$

$$f(i, j, a, b)^{self-att} = \texttt{softmax}_{ab}\Big((W_Q x_{ij})^\top W_K x_{ab} + (W_Q x_{ij})^\top r_{a-i,b-j}\Big) W_V \tag{2}$$

$$= p^{ij}_{a-i,b-j} W_v \tag{3}$$

For self-attention, W_Q, W_K, and W_V are learned linear transformations that are shared across all spatial locations, and respectively produce *queries*, *keys*, and *values* when used to transform x. Spatial geometry is captured by $r_{a-i,b-j}$, which is a learned relative position based embedding. The $(W_Q x_{ij})^\top W_K x_{ab}$ component captures the content-content interaction between the query pixel and a key pixel in the window. The $(W_Q x_{ij})^\top r_{a-i,b-j}$ component is the content-geometry interaction that captures the relationship between

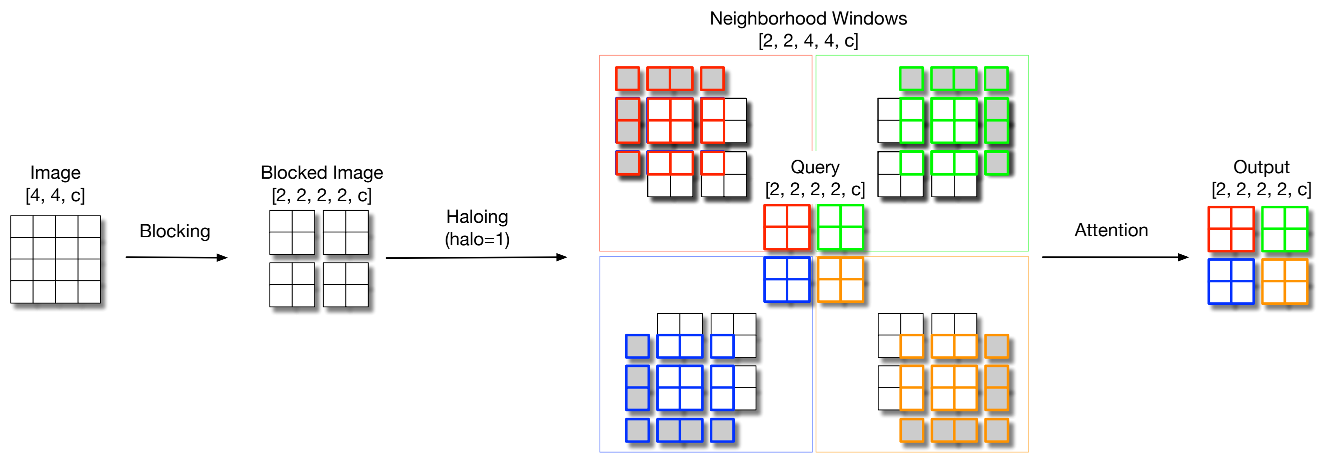

Figure 1. **HaloNet local self-attention architecture: The different stages of blocked local attention for a $[4, 4, c]$ image, block size $b = 2$, and halo $h = 1$.** The image is first blocked into non-overlapping $[2, 2, c]$ images from which the queries are computed. The subsequent haloing step then extracts a $[4, 4, c]$ memory around each of the blocks which linearly transform to keys and values. The spatial dimensions after attention are the same as the queries.

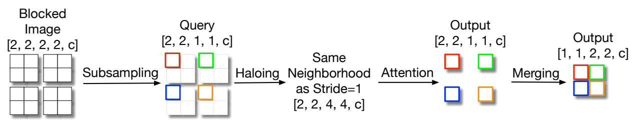

Figure 2. The attention downsampling layer subsamples the queries but keeps the neighborhood the same as the the stride=1 case.

the query and the relative position of the key pixel [48]. Note that this formulation preserves *translational equivariance*. If an object translates in an image, for any pixel within the object, the content around it stays the same, generating the same $p^{ij}_{a-i,b-j}$, thereby producing the same output after self-attention. To increase expressivity, multi-headed attention [57] is used, which repeats this computation multiple times in parallel with different parameters, analogous to group convolutions [30, 61].

In the *SASA* model of [43], the local window $\mathcal{N}(i, j)$ is a $k \times k$ window centered around (i, j), just like a convolution. The size of this local window k is an important setting to leverage in self-attention. Unlike dense convolutions, k can grow without significantly increasing the number of parameters. Since the projection parameters (W_Q, W_K, W_V) are independent of k, the only parameters that increase with k is $r_{a-i,b-j}$. However, $r_{a-i,b-j}$ constitutes a trivial fraction of the parameters compared to the projection parameters [1] , so increasing k does not not impact the number of parameters of the layer significantly. In contrast, the number of parameters in a convolution layer scale quadratically with k (*e.g.*, a 5×5 convolution has $\frac{25}{9}$ times the parameters of a 3×3 convolution). On the other hand, the computational cost of self-attention grows quadratically with k, preventing the use of very large values for k.

[1]For a window size as large as 63, and 16 dimensions per attention head, $r_{a-i,b-j}$ would add only $63 * 16 = 1008$ parameters per layer because $r_{a-i,b-j}$ are shared among heads. In contrast, if the dimensions of the attention layer were 512, W_Q, W_K, W_V would contribute 786432 parameters. We show details in the appendix.

2.2. Improving the speed-memory tradeoff by relaxing translational equivariance

Global self-attention, in which all locations attend to each other, is too expensive for most image scales due to the quadratic computation cost with respect to k. Thus, multi-scale visual backbones need to use local attention to limit the size of k. We follow the intuitive form of local attention developed in [43], which tries to mimic the square neighborhoods used by convolutions. This form of local attention requires extracting local 2D grids around each pixel. Unfortunately, while deep learning libraries automatically handle neighborhood gathering for convolutions, no such neighborhood gathering function exists for local self-attention (or any general local function). Thus, implementing local self-attention requires explicitly gathering the local neighborhoods before the actual self-attention operation can be performed. While the implementation of this local neighborhood gathering function might initially appear to be a relatively minor implementation detail, in practice, it must actually be carefully designed to reduce memory usage while avoiding unnecessary extra computation. An unoptimized implementation can prevent self-attention models from scaling up due to either out-of-memory errors or excessive slowness. The following discussion frames the design considerations of this neighborhood gathering function.

A straightforward approach would gather $k \times k$ sized windows *separately* around each pixel. As summarized in Table 1 (Row 1), this method blows up the memory used by

Method	Neighborhood Memory	Receptive Field	FLOPs Per Pixel
Global	HWc	$HW \times HW$	$4(HW)^2c$
Per pixel windows	HWk^2c	$k \times k$	$4k^2c$
SASA [43]	$\frac{HW}{b^2}(b+2h)^2c$	$k \times k$, where $h = \lfloor \frac{k}{2} \rfloor$	$4(b+2h)^2c$
Blocked local (ours)	$\frac{HW}{b^2}(b+2h)^2c$	$(b+2h) \times (b+2h)$	$4(b+2h)^2c$

Table 1. **Scaling behavior of self-attention mechanisms**. f is the number of heads, b is the size of the block, c is the total number of channels, and h is the size of the halo

a factor of k^2 due to replicating the pixel contents for each of the k^2 neighborhoods it participates in. This solution quickly leads to out-of-memory errors. *Global* attention (Row 4) is at the other end of the spectrum, where all pixels *share* the same neighborhood, lowering memory at the expense of considerably more FLOPs [2]. This solution slows down models significantly, while also imposing memory problems due the massize size of the attention matrix. A solution that lies in-between these two extremes should trade-off memory and compute appropriately, with the recognition that a small amount of waste is required.

A compromise solution can be achieved by leveraging the idea that *neighboring pixels share most of their neighborhood*. For example, two pixels that are right next to each other share $k \times (k-1)$ pixels of their neighborhoods. Thus a local neighborhood for a *block* of pixels can be extracted once together, instead of extracting separate neighborhoods per pixel. The FLOPs can be controlled by varying the number of pixels that form a block. We name this strategy *blocked local self-attention*. The two extremes discussed above are a special case of blocked local self-attention. Global attention corresponds to setting the block size to be the entire spatial extent, while the per-pixel extraction corresponds to setting the block size to be 1.

Figure 1 depicts the different steps involved in executing blocked local self-attention for an image with height $H = 4$, width $W = 4$, and c channels with stride 1. Blocking chops up the image into a $\frac{H}{b}$, $\frac{W}{b}$ tensor of *non-overlapping* (b, b) blocks. Each block behaves as a group of query pixels and a *haloing* operation combines a band of h pixels around them (with padding at boundaries) to obtain the corresponding *shared neighborhood* block of shape $(\frac{H}{b}, \frac{W}{b}, b+2h, b+2h, c)$ from which the keys and values are computed. $\frac{H}{b} \times \frac{W}{b}$ attention operations then run in parallel for each of the query blocks and their corresponding neighborhoods, illustrated with different colors in Figure 1. SASA [43] used the same blocking strategy[3], setting $h = \lfloor \frac{k}{2} \rfloor$ and uses attention masks to emulate pixel-centered neighborhood windows of size $k \times k$. Our approach For example, to achieve a 7×7 pixel centered window, [43] set $h = 3$. The use of attention masks gives the operation translational equivariance, since each pixel only looks at a square window around it.

However, the downside of using attention masks is that it wastes computation that must happen regardless due to the implementation of this algorithm. If attention masks are not used, the receptive field increases without any additional computation, as shown in Table 1 (Rows 2 and 3). However, pixel-level translational equivariance is lost because the non-square receptive fields means that the output of a pixel is dependent on which block it falls into. Take for example a pixel at the left edge of its block, which sees additional pixels that are to the right of its square receptive field. If the entire image is shifted one pixel to the right, the pixel now falls into right edge of a neighboring block, and now sees additional pixels that are to the left of its square receptive field. Thus the output of the pixel is dependent on its position in a block, which can change if the image shifts. Another perspective is that blocked local self-attention is only translational equivariant to shifts of size b. While pixel-level translational equivariance is considered important for achieving good performance[65], we find that empirically, using a non-masked block local self-attention actually improves the accuracy of the model (see Section 4.3). We suspect that the image shifting and cropping perturbations in common data augmentation strategies reduce the reliance on such inductive biases. Thus we adopt unmasked blocked local self-attention because it improves accuracy without sacrificing performance.

Another difference with SASA is our implementation of downsampling. We replace attention followed by post-attention strided average pooling by a single strided attention layer that subsamples queries similar to strided convolutions, as shown in Figure 2. Note that we use the same neighborhood as is extracted in the stride 1 case (Figure 1). This change does not impact accuracy while also reducing the FLOPs 4× in the downsampling layers. We also implement some important algorithmic optimizations that improve our throughput primarily by avoiding reshapes and data formatting operations. In interest of space, we list them in the Appendix D. Taken together, the speedups produced by these improvements are significant as seen in Figure 3, with up to 2× improvements in step time. These improvements can be leveraged to train large self-attention models that were previously too expensive. We leave additional optimizations, such as fused operations and better pipelining of memory accesses with computation, to future work.

To conclude this section, it's important to note that in the deeper layers of multiscale architectures, smaller spatial dimensions and larger channels would shift the compute calculus in favor of global attention. The models we introduce in Section 4, also take advantage of this, typically using local

[2]To illustrate this, on a 128×128 resolution with 64 channels, global self-attention would incur about 28 times more FLOPs than a 3×3 convolution with 64 input and output channels

[3]Code for both SASA and HaloNet will be made available, along with the checkpoints for HaloNet

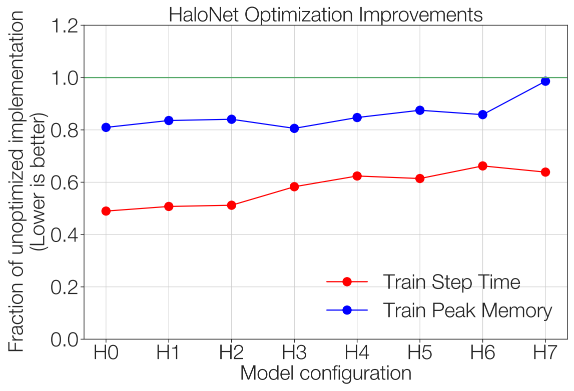

Figure 3. **Optimizations improve performance.** The improvements here are a result of reducing FLOPs with our attention downsampling and improved local self-attention algorithms that avoid reshapes and data formatting. In some cases, we halve the training step time computed on TPU v3.

attention in the higher resolutions and global attention when the image resolutions are the smallest.

Output Resolution	Layers
$\frac{s}{4} \times \frac{s}{4}$	7×7 conv stride 2, 64 3×3 max pool stride 2
$\frac{s}{4} \times \frac{s}{4}$	$\left\{ \begin{array}{c} 1 \times 1, 64 \\ \text{attention}(b, h), 64 \cdot r_v \\ 1 \times 1, 64 \cdot r_b \end{array} \right\} \times 3$
$\frac{s}{8} \times \frac{s}{8}$	$\left\{ \begin{array}{c} 1 \times 1, 128 \\ \text{attention}(b, h), 128 \cdot r_v \\ 1 \times 1, 128 \cdot r_b \end{array} \right\} \times 3$
$\frac{s}{16} \times \frac{s}{16}$	$\left\{ \begin{array}{c} 1 \times 1, 256 \\ \text{attention}(b, h), 256 \cdot r_v \\ 1 \times 1, 256 \cdot r_b \end{array} \right\} \times l_3$
$\frac{s}{32} \times \frac{s}{32}$	$\left\{ \begin{array}{c} 1 \times 1, 512 \\ \text{attention}(b, h), 512 \cdot r_v \\ 1 \times 1, 512 \cdot r_b \end{array} \right\} \times 3$
$\frac{s}{32} \times \frac{s}{32}$	$1 \times 1, d_f$
1×1	global average pooling fc, 1000

Table 2. **HaloNet model family specification.**

2.3. HaloNet

Using the implementation of local 2D self-attention with haloing detailed above, we propose a new model, *HaloNet* that matches state-of-the-art convolutional models on the parameter-accuracy trade-off curve. We leverage the structure of ResNets [17] that stack multiple residual bottleneck blocks together (see Table 2). HaloNet uses a few minor modifications from ResNets: (a) adding a final 1×1 convolution before the global average pooling for larger models, following EfficientNet [55], (b) modifying the bottleneck block width factor, which is traditionally fixed at 4, (c) modifying the output width multiplier of the spatial operation, which is traditionally fixed at 1, (d) changing the number of blocks in the third stage from 4 to 3 for computational reasons because attention is more expensive in the higher resolution layers. We also fix the number of heads for each of the four stages to $(4, 8, 8, 8)$ because heads are more expensive at higher resolutions. To summarize, the scaling dimensions in HaloNet are: image size s, query block size b, halo size h, attention output width multiplier r_v, bottleneck output width multiplier r_b, number of bottleneck blocks in the third group l_3, and final 1×1 conv width d_f. Our attention neighborhoods range from 14×14 ($b = 8, h = 3$) to 18×18 ($b = 14, h = 2$).

Since the ResNet structure was initially designed for convolutions, we suspect that designing architectures specifically for attention may improve HaloNet. In our work we maintained homogeneity across all layers of model for hyperparameters such as the block (b) and halo (h) sizes. We also hope that using automated architecture search methods [55] to optimize these hyperparameters for specific accelerators will lead to better local attention architectures. In our work, we train with comparable image sizes as EfficientNet models to determine if attention models can scale to larger images.

3. Related Work

Attention has steadily risen in adoption in vision models in recent years. First introduced in various forms of sequence modeling [15, 1, 57], attention was used to attend to image features in the text generation module of image captioning models [62]. Attention is also closely related to non-local means [4], a pairwise-weighted global sum of pixels originally developed for image denoising. [59] applied non-local means on top of spatially downsampled convolutional features to improve video classification. However, since these methods scale quadratically with receptive field size, they cannot be used because the spatial size is too large. In order to apply self-attention, [40] applies local attention on images for the task of image generation. [3] spatially downsample the features for attention and concatenate the attention outputs to convolutional features. Instead, we directly build on top of the approach of [43], who compute attention on local regions in order to build a fully self-attentional vision model for classification and object detection. Different forms of attention for pure self-attention vision models have also been proposed [22, 66], which are orthogonal and complementary to the focus on scaling in this work. In addition to attention over the spatial extent that we focus on, components that perform attention over channels have also been used to augment convolutional models [23, 33]. In recent and concurrent work, Vision Transformer [12] show that apply-

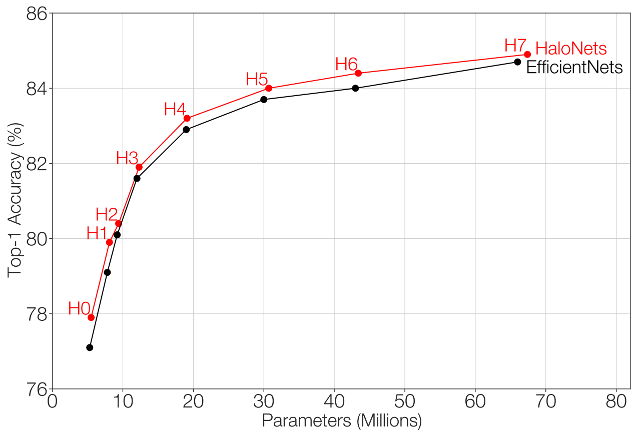

Figure 4. **HaloNets can match EfficientNets on the accuracy vs. parameter trade-off**. The accuracies for EfficientNets B5 and B7 were obtained using RandAugment.

ing transformers on projections of *non-overlapping* image patches can achieve accuracies comparable to SOTA when pre-trained on very large (JFT-300M [52]) and medium sized (ImageNet-21k [11]) classification datasets. However, their models do not adopt a multiscale architecture and our focus in this work is training on ImageNet [47] from scratch. In Section 4.5, we conduct transfer experiments and compare with ViT and BiT [29].

Generally, the performance of computational primitives tend to improve over time due to algorithmic changes to the primitive and better software implementations. In particular, convolution have improved over the last decade through changes in (a) the computation of the primitive [5, 28, 37, 56, 60, 31]; (b) the software implementation [6]; (c) the structure of the primitive itself, through for example, grouped convolution [61] and depthwise separable convolution [49]. Attention is in the beginning phases of this performance improvement trajectory, and given its importance in sequence modeling [57], it will likely see sustained effort to enhance performance. Local attention could also receive performance improvements if it is adopted more widely to combat the general problem of processing large inputs. Our work introduces blocked local attention to efficiently process immediate neighbors. Other forms of non-global pixel interaction can also be implemented efficiently [7, 21, 58, 2].

4. Experiments

Each HaloNet model (H0–H7) is designed by successively growing the values of the hyperparameters defined in Table 2. In interest of space, we leave the exact configurations of our models to the Appendix C.2. We also leave the training and evaluation of larger HaloNet models that compare with larger EfficientNet models for future work.

4.1. HaloNets are competitive with state-of-the-art convolutional models

We train our HaloNet models on ImageNet [47] (ILSVRC-2012) benchmark with a batch size of 4096 and learning rate of 1.6, which is linearly warmed up for 10 epochs and followed by cosine decay [36]. The models are trained for 350 epochs with Nesterov's Accelerated Gradient [38, 53], and regularized with dropout [51], weight decay, RandAugment [10] and stochastic depth [24].

We find that HaloNets perform at par or slightly better (Figure 4) than EfficientNet models for the same parameters, outperforming other model families. Our best model, H7, achieves **84.9**% top-1 ImageNet validation accuracy and **74.7**% top-1 accuracy on ImageNet V2 [45] (with a -0.5% gap to the linear fit in [45]). For each of our HaloNet models, we use image sizes comparable to the corresponding EfficientNet model, training on images sizes up to 600 $\times$ 600. (Table A2). For a comparison of our latencies with EfficientNet, the reader can refer to Section 5. To the best of our knowledge, these results are the first to show that self-attention based models for vision perform on par with the SOTA for image classification when trained on imagenet from scratch. Note that for all our experiments, we report accuracies at the end of training and we tune regularization hyperparameters such as augmentation hyperparameters for the baselines and HaloNet models.

4.2. Model study 1: comparing self-attention and convolutions

In the following sections, we will focus on model studies to distinguish the advantages of self-attention over convolutions for vision and and understand how to best design self-attention vision architectures. This knowledge is important since much of the progress in convolutional networks comes from improvements in architecture design while keeping the core convolution primitive the same [30, 54, 17]. We believe our study is the first to explicitly examine the design of optimal self-attention vision architectures.

For the remainder of the experimental section, we compare with ResNet-50 [18], the canonical vision model, because many of the components that we ablate have been well studied for ResNet-50, allowing us to use best practices for the baseline model. We tune our baseline ResNet-50 implementation to achieve a better accuracy, 77.6%, compared to commonly reported numbers in the literature. For example, [17] report 76.3%. We then create a new HaloNet architecture, HaloNet-50, that exactly matches the ResNet-50 architecture by replacing spatial convolutions with local self-attention. HaloNet-50 and ResNet-50 have about 18.0 million and 25.5 million parameters respectively. We train both for 150 epochs on 256 $\times$ 256 image size. We share other training details of the ablation set-up in the appendix

4.2.1 Transfer of convolutional components to self-attention

Utilizing regularizations and architectural modules beyond the core primitive is critical for achieving strong results [19]. In this section, we study the effects of these additional components on self-attention models. The components we study were all designed for use in convolutional models, as they were developed through experimentation (either human or automated search) on convolutional models. We examine whether these components can successfully transfer to the new model family of self-attention networks.

We focus on 4 different components based on the design of EfficientNet [55], 2 architecture modules and 2 regularizations: Squeeze-and-Excitation (SE) [23], a channel attention module used after the spatial convolution; SiLU/Swish-1 [44, 13, 20], an activation function with the form $x \cdot \text{sigmoid}(x)$; RandAugment (RA) [10], a data augmentation scheme that simplifies AutoAugment [9]; and Label Smoothing (LS) [54], a smoothing of the label distribution.

The results of adding these various components to the baseline is in Table 3. Suprisingly, regularizations of the same strength improve HaloNet accuracies significantly more than ResNet, despite HaloNet having around 30% fewer parameters than ResNet. When label smoothing and RandAugment are added, HaloNet improves by 1.3% while ResNet improves by 0.8%. This result suggests that self-attention models may require regularizations that are typical of larger convolutional models, perhaps due to the expressivity of self-attention.

However, the architecture modules that were developed for convolutional models only improve attention models by a small amount. When Squeeze-and-Excitation (SE) and SiLU/Swish-1 are added, ResNet improves by 1.3% while HaloNet only improves by 0.4%. We speculate that HaloNet models benefit from the gating and multiplicative interactions that comprise self-attention and do not need explicit gating such as SE. Further research must be conducted in order to discover architecture modules that can consistently improve a variety of self-attention models. Inspired by these findings, we decided to use label smoothing, SiLU/Swish-1, and RandAugment in our HaloNet $H0 - H7$ models. We also use stochastic depth for our larger models [24, 55].

4.2.2 Increasing image sizes improve accuracies

A beneficial property of self-attention is attention is that the receptive field size can scale along with image size without significantly impacting the number of parameters (see Section 2.1). As shown in Figure 6, HaloNet consistently improves when using larger images. Although we also see improvements with convolutional models, the accuracy gap between HaloNets and ResNets is maintained.

Components	HaloNet Accuracy	Baseline Δ	ResNet Accuracy	Baseline Δ
Baseline	78.6	0.0	77.6	0.0
+ LS	79.7	1.1	78.1	0.5
+ LS, RA	79.9	1.3	78.4	0.8
+ SE	78.6	0.0	78.6	1.0
+ SE, SiLU/Sw1	79.0	0.4	78.9	1.3
+ LS, SE	79.7	1.1	78.9	1.3
+ LS, SE, SiLU/Sw1	79.9	1.3	79.1	1.5
+ LS, SE, SiLU/Sw1, RA	80.5	1.9	79.5	1.9

Table 3. **HaloNet improves more than ResNet with regularizations, but does not improve significantly with architectural modules that strongly benefit ResNet.** Starting from a baseline model, adding label smoothing (LS), RandAugment (RA), Squeeze-and-Excitation (SE), and SiLU/Swish-1 (SiLU/Sw1).

4.3. Model study 2: HaloNet architecture study

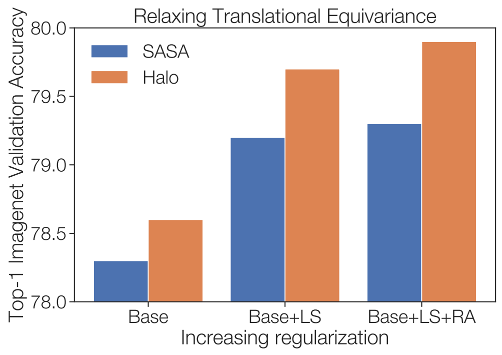

Figure 5. **Relaxing translational equivariance improves accuracies**

In this section, we will study the impact of relaxing translational equivariance and the relationship of neighborhood window and halo sizes. In the interest of space, a detailed study of scaling various components of our models such as r_v, r_{qk} etc can be found in the Appendix B.

Relaxing translational equivariance: In Figure 5, we see that HaloNet-50 with $b = 8$, and $h = 3$ achieves better accuracies using the same block and halo to achieve 7×7 neighborhoods with attention masks [43] and the gap widens with more regularizations. This suggests that larger receptive fields are more important than inductive biases such as translational equivariance.

Window and halo size: When using the blocked input format, there are two ways of changing the window size of attention: changing the query block size or the halo size. For the same window size w, smaller query blocks and larger halos require more memory than larger query blocks and smaller halos, as discussed in section 2.2.

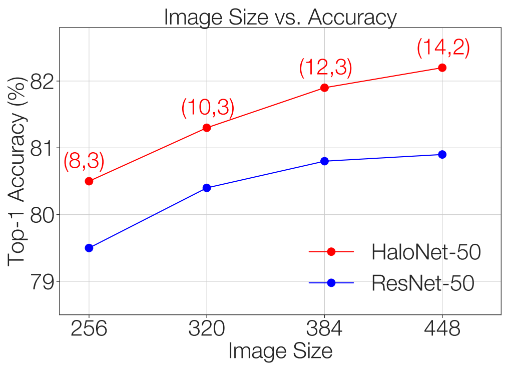

Figure 6. **The accuracy gap between HaloNet-50 and ResNet-50 is maintained with increasing image sizes.** The HaloNet experiments are annotated with block size (b), halo size (h).

We see in Figure 7 that accuracy consistently improves as the window size increases. In particular, doubling the window size from 6×6 to 12×12 produces a 1.3% accuracy gain. These results suggest that increasing window size can be successfully used to scale models without increasing the number of parameters, potentially beneficial for production environments. Furthermore, for a fixed window size, the choice of query block size does not impact results, enabling the usage of larger query block sizes to reduce memory. Figure 7 also shows that eschewing haloing for *non-overlapping* attention, can lower accuracy significantly unless the blocks are quite large. For example using a block size of 4 and a halo of 1 results in better accuracy than using a block size of 8 with 0 halo, despite a smaller neighborhood size.

4.4. Convolution-Attention hybrids improve the speed-accuracy tradeoff

In our final set of ablations, we replace self-attention with convolutions to understand where attention layers are currently most beneficial. In Table 4, we show results for replacing attention layers with convolutions with squeeze-and-excitation modules in each of the stages of our best performing model (HaloNet H7). Having convolutions in all stages except the last yields the fastest model albeit with a significant loss in top-1 accuracy (1%). Splitting the allocation between convolutions (in stages 1–2) and attention (in stages 3–4) minimally detriments predictive accuracy while significantly improving training and inference step times. We leave a detailed study of improved hybrid models for future work.

4.5. Transfer from ImageNet-21k

Our experiments thus far have focused on training from scratch on ImageNet-ILSVRC-2012 [47], where regularizations and longer training are critical for good accuracies. Papers such [12, 29] have shown that a short finetuning step after pretraining models on larger labelled datasets such

Conv Stages	Attention Stages	Top-1 Acc (%)	Norm. Train Time
-	1, 2, 3, 4	84.9	1.9
1	2, 3, 4	84.6	1.4
1, 2	3, 4	84.7	1.0
1, 2, 3	4	83.8	0.5

Table 4. **Replacing attention layers with convolutions in stages 1 and 2 exhibit the best speed vs. accuracy tradeoff.** All the models had about 67 million parameters and the train and inference times are normalized to the corresponding times for EfficientNet B7. Please see Figure 8 for a comparison of step time with other HaloNet models.

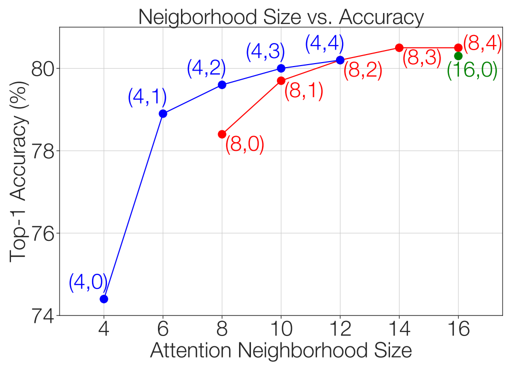

Figure 7. **Increasing window sizes improves accuracy up to a point.** The experiments in the graph have been annotated with their block size (b), halo size (h), $h = 0$ implies attention with *non-overlapping* blocks

as ImageNet-21k [11] or JFT-300M [52] can achieve better accuracies without the need for regularization. To understand the transfer properties of HaloNet models, we scale up HaloNet-H4 by increasing the base width to 128 and evaluate the transfer protocol from [29], pretraining on the public ImageNet-21k dataset, and finetuning on ImageNet. Following our observation in Table 4, we train a hybrid version of this model with convolutions in the first two stages. Note that our hybrids can be seen as using a series of convolution layers to downsample the image. Since we For a fair comparison with [29], we do not use squeeze-and-excitation [23] in the stages with convolutions. We also investigate the effect of replacing the convolutional stem with linear projections of 4×4 non-overlapping patches, similar to the vision Transformer. The details of the models can be found in the Appendix E. ImageNet-21k contains 14.2 million annotated images, and 21k labels, both an order of magnitude larger than ImageNet. Following [29], we pretrain for 90 epochs with a batch size of 4096, and a base learning rate of 0.16, which is linearly warmed up for 2 epochs followed by cosine decay [36]. We also use a weight decay of 0.00008, and

Model	$\mathrm{AP^{bb}}$	$\mathrm{AP^{bb}_s}$	$\mathrm{AP^{bb}_m}$	$\mathrm{AP^{bb}_l}$	$\mathrm{AP^{mk}}$	$\mathrm{AP^{mk}_s}$	$\mathrm{AP^{mk}_m}$	$\mathrm{AP^{mk}_l}$	Speed (ms)	Train time (hrs)
R50 baseline in lit	42.1	22.5	44.8	59.1	37.7	18.3	40.5	54.9	409	14.6
R50 + SE (our baseline)	44.5 (+2.4)	25.5	47.7	61.2	39.6 (+1.9)	20.4	42.6	57.6	446	15.2
R50 + SE + Local Att ($b = 8$)	45.2 (++0.7)	25.4	48.1	63.3	40.3 (++0.7)	20.5	43.1	59.0	540	15.8
R50 + SE + Local Att ($b = 32$)	45.4 (++0.9)	25.9	48.2	63.0	40.5 (++0.9)	21.2	43.5	58.8	613	16.5
R101 + SE (our baseline)	45.9 (+3.8)	25.8	49.5	62.9	40.6 (+2.9)	20.9	43.7	58.7	740	17.9
R101 + SE + Local Att ($b = 8$)	46.8 (++0.9)	26.3	50.0	64.5	41.2 (++0.6)	21.4	44.3	59.8	799	18.4

Table 5. **Accuracies on object detection and instance segmentation.** We experiment with two settings for self-attention in the last stage: A block size of (b) of 8 and a halo size (h) of 3 and also with ($b = 32, h = 3$) for ResNet-50. *bb* (bounding box) refers to detection, and mk (mask) refers to segmentation. The identifiers s, m, and l refer to small, medium, and large objects respectively. Speed is measured as the milliseconds taken by only the backbone (and not the FPN) for a batch size of 32 on 2 TPUv3 cores. The train time the total training time calculated from the peak images/sec of the Mask-RCNN training run on 8 TPUv3 cores with a batch size of 64.

Model	Parameters (Millions)	Pretraining Image Size (Pixels)	Pretraining Step Time (32 per core)	Finetuning Image Size	Finetuning Top-1 Accuracy (%)	Inference Speed img/sec/core
H4 (base 128)	85	256	377 ms	384/512	85.6/85.8	121.3/48.6
H4 (base 128, 4×4 patch)	85	256	366 ms	384/512	85.4/85.4	125.7/56.5
H4 (base 128, Conv-12)	87	256	213 ms	384/512	85.5/85.8	257.6/120.2
ViT-L/16	300	224	445 ms	384/512	85.2/85.3	74.6/27.4
BiT-M	928	224	1021 ms	384	85.4	54.2

Table 6. **HaloNet models pretrained on ImagetNet-21k perform well when finetuned on ImageNet**. For HaloNet and ViT, we finetuned on 384×384 and 512×512 size images. The pretraining step time reports the TPUv3 compute time for a batch size of 32 per core. The inference speed is also computed on a single TPUv3 core.

train with Nesterov's Accelerated Gradient [38, 53] during pretraining and finetuning. We pretrain on 256×256 size images and finetune on different image sizes, as shown in Table 6. Our wider H4 and hybrid-H4 models achieves better accuracy than the Vision Transformer and a $4\times$ wide ResNet-152 from [29] and are also faster at inference on larger images. We finetune for 8 epochs on ImageNet, initializing with the parameters learned from pretraining except for the label embedding matrix, which is initialized to zeros. We train with a batch size of 512, a learning rate of 0.016 and cosine decay after linearly warming it up for 0.5 epochs. We benefit from finetuning with a label smoothing of 0.1 during finetuning despite pretrainig on a larger dataset. We do not use Polyak averaging [41], and other regulariations during finetuning.

Our preliminary results on transfer are promising since we achieve better parameter-accuracy and speed-accuracy tradeoffs than other models on this dataset. We leave the study of transfer with larger HaloNet and HaloNet hybrids for future work. The speed advantages of our models on larger images make them desirable for challenging structured prediction tasks on large images such as object detection and instance segmentation, which we briefly explore in the next section.

4.6. Detection and instance segmentation

To understand if our primitives will generalize to structured prediction tasks on larger images, we conduct initial investigations with the simple attention-convolutional hybrids on detection and instance segmentation, using the Mask R-CNN [16] framework. These hybrids are also faster and consume less memory than pure attention models, enabling faster experimental cycles. We only replace the last 3 convolutional layers in the ResNet-50 and ResNet-101 backbones with two halo layers with block size, $b = 8$ and halo size $h = 3$ (Rows 3 and 6 in Table 5). For ResNet-50, we also examine using $b = 32$ and halo size $h = 3$ to understand benefits from larger receptive fields. We also use squeeze-and-excitation with convolutions and pre-train them on 512×512 images with the regularizations mentioned in Section 4.2.1: label smoothing, RandAugment, and stochastic depth. We train our models on the COCO dataset [35] with 1024×1024 size images for 32 epochs, using the Cloud TPU Detection Codebase [4]. We provide more training details in the Appendix C.4.

Our ResNet-50 baseline in row 2 of Table 5, is significantly better than what is usually reported in the literature

[4]https://github.com/tensorflow/tpu/tree/master/models/official/detection

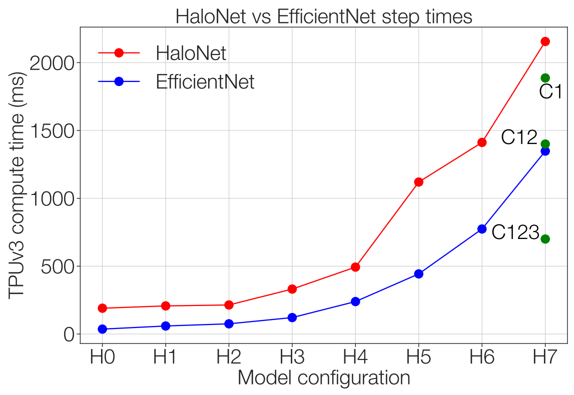

Figure 8. **Pure attention based HaloNet models are currently slower to train than efficient net models.** The times are the TPUv3 compute time needed to process a batch size of 32 per core. The points in green with annotations C1, C12, and C123 correspond to the hybrid models with convolutions in stages 1, 1–2 and 1–3 respectively. (see Table 4).

(row 1). Our attention variants achieve at least 0.7 mAP gains on bounding box detection and at least 0.6 mAP gains on instance segmentation on top of our stronger baselines (denoted by **++** in rows 3, 4 and 6 in Table 5). The gain from local attention with block size $b = 8$ closes half of the mAP gap between the R50 and R101 baselines in detection and 70% of the gap in instance segmentation despite being less than a third of the gap in terms of wall-clock time. Local attention with $b = 8$ and $h = 3$ also improves on top of the deep R101 backbone. Interestingly, localization of large objects (AP^*_l) shows the largest improvement when attention is used. Larger block sizes ($b = 32$ in row 4) achieve very close performance to $b = 8$ while being slower. However, we see that $b = 32$ does much better than $b = 8$ on small objects (AP^*_s). Future work can combine the best of these two settings. Note that with $b = 32$, the last two attention layers do global attention since the image is downsampled to $\frac{1024}{32} = 32$ pixels in each spatial dimension. Concurrent work, BoTNet [50], uses global self-attention in ResNet-Attention hybrids for structured prediction tasks and classification. See [50] for additional details on the efficacy of global attention for localization tasks

These models have only three layers of self-attention, and more layers could alter these results. We leave the study of detection and instance segmentation with pure attention models to future work.

5. Discussion

In this work, we built multiscale self-attention models that are competitive with the best convolutional models. To achieve this result, we developed two attention improvements: blocked local attention and attention downsampling. We also performed multiple ablations to understand how to improve the scaling of self-attention models.

Our results demonstrate that self-attention is competitive accuracy-wise when training on ImageNet from scratch. Figure 8 shows that pure self-attention[5] based HaloNets are currently slower to train than the corresponding EfficientNets and require further optimizations for large batch training. However, our hybrids have the same speed-accuracy tradeoff as EfficientNets. On transfer from ImageNet-21k, our models outperform very strong models such as BiT [29] and ViT [12], on both accuracy and speed. Model optimizations, such as using architecture search methods to find better speed-accuracy tradeoffs or different forms of more powerful and/or efficient attention forms [66, 46], are promising directions for machine learning researchers. Implementation optimizations, such as better memory management, can improve the practicality of these models. Also, scaling up our models to larger widths might cause our operations to transition from being memory bound to compute bound, and lead to better speed-accuracy tradeoffs. We leave this study for future work. Overall, our work shows that self-attention can be competitive in regimes traditionally dominated by computer vision. Future work can push these boundaries further, both in terms of scale and efficiency.

6. Acknowledgements

We would like to thank David Fleet for valuable discussions. We would also like to thank Irwan Bello, Barret Zoph, Mingxing Tan, and Lucas Beyer for valuable commentary on earlier drafts of the paper.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 5

[2] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. *arXiv preprint arXiv:2102.08602*, 2021. 6

[3] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3286–3295, 2019. 1, 2, 5

[4] 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, pages 60–65. IEEE, 2005. 1, 2, 5

[5] Kumar Chellapilla, Sidd Puri, and Patrice Simard. High performance convolutional neural networks for document processing. In *Tenth International Workshop on Frontiers in Handwriting Recognition*, 2006. 6

[6] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shel-

[5]By pure attention we mean models that use self-attention in all layers except the stem, which is convolutional.

hamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014. 6

[7] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 6

[8] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. *arXiv preprint arXiv:1911.03584*, 2019. 2

[9] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019. 7

[10] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical data augmentation with no separate search. *arXiv preprint arXiv:1909.13719*, 2019. 6, 7, 14

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009. 6, 8

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 5, 8, 10

[13] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. 7

[14] Gamaleldin F Elsayed, Prajit Ramachandran, Jonathon Shlens, and Simon Kornblith. Revisiting spatial invariance with low-rank local connectivity. *arXiv preprint arXiv:2002.02959*, 2020. 2

[15] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013. 5

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 9, 14

[17] 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*, pages 770–778, 2016. 1, 5, 6, 14

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. 1, 6

[19] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019. 7

[20] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 7

[21] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*, 2019. 6

[22] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3464–3473, 2019. 2, 5

[23] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 5, 7, 8

[24] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016. 6, 7

[25] David H Hubel and TN Wiesel. Shape and arrangement of columns in cat's striate cortex. *The Journal of physiology*, 165(3):559, 1963. 2

[26] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968. 2

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 15

[28] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678, 2014. 6

[29] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370*, 6(2):8, 2019. 6, 8, 9, 10

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 3, 6

[31] Andrew Lavin and Scott Gray. Fast algorithms for convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4013–4021, 2016. 6

[32] Linjie Li, Zhe Gan, Yu Cheng, and Jingjing Liu. Relation-aware graph attention network for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10313–10322, 2019. 1

[33] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 510–519, 2019. 5

[34] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 2

[35] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 9

[36] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 6, 8, 14

[37] Michael Mathieu, Mikael Henaff, and Yann LeCun. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851*, 2013. 6

[38] Y. E. NESTEROV. A method for solving the convex programming problem with convergence rate $o(1/k^2)$. *Dokl. Akad. Nauk SSSR*, 269:543–547, 1983. 6, 9

[39] Sylvain Paris, Pierre Kornprobst, Jack Tumblin, and Frédo Durand. *Bilateral filtering: Theory and applications*. Now Publishers Inc, 2009. 1

[40] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, 2018. 5

[41] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992. 9

[42] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. *arXiv preprint arXiv:2003.13678*, 2020. 1

[43] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 68–80. Curran Associates, Inc., 2019. 1, 2, 3, 4, 5, 7, 14, 15

[44] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. 1, 7

[45] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, pages 5389–5400. PMLR, 2019. 6

[46] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *arXiv preprint arXiv:2003.05997*, 2020. 10

[47] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 2, 6, 8

[48] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018. 2, 3

[49] Laurent Sifre. Rigid-motion scattering for image classification. *Ph. D. thesis*, 2014. 6

[50] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. *arXiv preprint arXiv:2101.11605*, 2021. 1, 10

[51] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 6

[52] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017. 6, 8

[53] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, 2013. 6, 9

[54] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 6, 7

[55] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 1, 5, 7, 14

[56] Nicolas Vasilache, Jeff Johnson, Michael Mathieu, Soumith Chintala, Serkan Piantino, and Yann LeCun. Fast convolutional nets with fbfft: A gpu performance evaluation. *arXiv preprint arXiv:1412.7580*, 2014. 6

[57] 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*, pages 5998–6008, 2017. 1, 3, 5, 6, 14

[58] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. *arXiv preprint arXiv:2003.07853*, 2020. 2, 6

[59] 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*, pages 7794–7803, 2018. 1, 2, 5

[60] Shmuel Winograd. *Arithmetic complexity of computations*, volume 33. Siam, 1980. 6

[61] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 3, 6

[62] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015. 1, 5

[63] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3323–3332, 2019. 1

[64] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. *arXiv preprint arXiv:2003.11142*, 2020. 14

[65] Richard Zhang. Making convolutional networks shift-invariant again. *arXiv preprint arXiv:1904.11486*, 2019. 4

[66] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. *arXiv preprint arXiv:2004.13621*, 2020. 1, 5, 10

[67] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image

recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018. 1

Appendix

A. Relative embeddings add very few parameters to the model

Our parameters grow very slowly with receptive field. In this section, we will show that the number of parameters in the relative embeddings, the only spatially dependent parameters, is quite small. As described in the paper, the output of local 2D self-attention at position (i, j) is computed as:

$$y_{ij} = \sum_{a,b \in \mathcal{N}(i,j)} \texttt{softmax}_{ab} \left(q_{ij}^\top k_{ab} + q_{ij}^\top r_{a-i,b-j} \right) v_{ab} \quad (4)$$

where the queries $q_{ij} = W_Q x_{ij}$, keys $k_{ab} = W_K x_{ab}$, and values $v_{ab} = W_V x_{ab}$ are linear transformations of the pixels, and $r_{a-i,b-j}$ is a learned relative position based embedding. Following the Transformer [57], we also use multihead attention, where we run multiple instances of the self-attention in parallel with different parameters. However, each head *shares* the parameters for the relative embeddings $r_{a-i,b-j}$. For an attention window of size k around each pixel, we factorize the relative embeddings along height and width following [43], and we allocate half the channels within a head to each of these. Keeping the dimension per head fixed at 16 as mentioned in the paper, this gives a constant $2(k-1) * 16$ parameters per attention layer layer for $r_{a-i,b-j}$. In contrast, if the channels in an attention layer are d, then each of the three linear transformations has d^2 parameters. Thus the ratio of parameters in the relative embeddings as compared with the linear projections is $\frac{2(k-1)*16}{3d^2}$, which is small for typical values of k and d.

Dimension	Values		Accuracy	Baseline
	Baseline	Scaled		Δ
Layers	50	98	81.4	0.9
r_v	1.0	3.0	81.0	0.5
r_w	1.0	1.25	80.9	0.4
r_b	4.0	6.5	80.6	0.1
r_{qk}	1.0	6.5	80.3	-0.2

Table A1. **Increasing the number of channels for the values and number of layers has the most impact on accuracy.**

B. Study of enlarging self-attention models

In Section 4.3, we presented some scaling properties of our models. In Table A1, we try to understand which other parts of our models most impact accuracy. For our study, we increase the size of HaloNet-50 by scaling different hyperparameters to reach a parameter budget of 30 million. We find that adding more computation in the attention by increasing r_v and adding more layers are most fruitful scaling dimensions for increasing accuracy.

C. Experimental details, hyperparameters

In this section, we list the experimental details and model configurations that were omitted from the main body in interest of space

C.1. Experimental details for model studies

In Sections 4.2 and 4.3, all the HaloNet-50 models use the same layer allocations and channels widths as the standard ResNet-50 [17] model. Both ResNet-50 and HaloNet-50 models were trained for 150 epochs on 256×256 size images with a learning rate of 0.1. For the experiments with RandAugment, we used a weight decay of 0.00004 for the settings that used RandAugment [10], and 0.00008 otherwise. Using a weight decay of 0.00008 with RandAugment seemed to have a negative impact on accuracies with ResNet-50. We used a RandAugment magnitude of 10 in these sections. For HaloNet-50, we used a block size $b = 8$, and halo $h = 3$. We fixed the number of channels per head to be 16. For the SASA models in section, we used a pixel *centered* window of size 7×7 following [43].

C.2. HaloNet Models

In Table A2, we describe the configurations of our HaloNet models, $H1 - H7$. The hyperparameters in the HaloNet family are: image size s, query block size b, halo size h, attention output width multiplier r_v, bottleneck output width multiplier r_b, number of bottleneck blocks in the third group l_3, and final 1×1 conv width d_f. Each of our HaloNet models is trained on a comparable image size to the corresponding EfficientNet [55] model, which can be found in Table A2.

C.3. Classification hyperparameters

In this section we complete the details of our training and regularization setup. We used a weight decay of $2e^{-5}$ and using a cosine annealing scheme [36] with learning rate 0.1. The largest models consistently overfit at the very end of training, which we attribute to the learning rate going to 0 at the end of training [64]. To combat this, we set the end of the cosine annealing to be $\frac{1.0}{128}$ of the original learning rate instead of 0. For RandAugment [10], we grow our RangAugment magnitudes for the smallest $H0$ to the the largest $H7$ models as $6, 8, 10, 14, 17, 19, 24$ and 31. Note that we have not extensively tuned the RandAugment magnitudes.

C.4. Detection and instance segmentation hyperparameters

We use Mask-RCNN [16] for all detection and instance segmentation experiments. We pretrain the backbone on ImageNet, mostly reusing the same hyperparameters as in Section C.3. Backbones are pretrained for 350 epochs using an image size of 512, which was chosen to be closer to the

HaloNet Model	b	h	r_v	r_b	**Total Layers**	l_3	s	d_f	**Params (M)**	**EfficientNet Params (M)**	**EfficientNet Image Size (M)**
H0	8	3	1.0	0.5	50	7	256	–	5.5	B0: 5.3	224
H1	8	3	1.0	1.0	59	10	256	–	8.1	B1: 7.8	240
H2	8	3	1.0	1.25	62	11	256	–	9.4	B2: 9.2	260
H3	10	3	1.0	1.5	65	12	320	1024	12.3	B3: 12	300
H4	12	2	1.0	3	65	12	384	1280	19.1	B4: 19	380
H5	14	2	2.5	2	98	23	448	1536	30.7	B5: 30	456
H6	8	4	3	2.75	101	24	512	1536	43.4	B6: 43	528
H7	10	3	4	3.5	107	26	600	2048	67	B7: 66	600

Table A2. **Configurations of HaloNet models, each of which matches a model from the EfficientNet family in terms of parameters**. The number of heads in the four stages are $(4, 8, 8, 8)$. The notations are: image size s, query block size b, halo size h, attention output width multiplier r_v, bottleneck output width multiplier r_b, number of bottleneck blocks in the third group l_3, and final 1×1 conv width d_f

1024 image size used in detection setting. The models were regularized with RandAug at a magnitude of 15 and stochastic depth with probability 0.1, and use Squeeze-Excitation with a reduction factor of $\frac{1}{8}$. The detection code and hyperparameters directly used the open-source TPU detection and segmentation framework [6]. During the detection / instance segmentation phase, the backbone is initialized with the pretrained weights, while the other parameters are initialized from scratch. The model is trained for 67500 steps with 0.1x learning rate decays at 60000 and 65000 steps, uses a learning rate of 0.1 in SGD with 0.9 momentum, a warmup of 500 steps with a fixed learning rate of $\frac{2}{300}$, a batch size of 64 spread across 32 TPUv3 cores, 1024×1024 image size, an L2 weight decay of $4e^{-5}$, and multi-scale jitter with magnitudes between $[\frac{4}{5}, \frac{5}{4}]$.

D. Optimizations

We endeavor to avoid data formatting operations whenever possible, which can slow down the model, resulting in the following two key optimizations

- **Persistent blocking:** Once the image is blocked, we flatten the (b, b) blocks to sequences of length b^2, and we do not reshape it back to 4D until the end of the network, implementing operations such as batch normalization [27] to handle the blocked format. The image is thus processed in 5D: (Batch, $\frac{H}{b}, \frac{W}{b}, b^2, c$) instead of (Batch, H, W, c).

- **Gathers with convolutions:** The haloing described in Section 2.2 is also carried out in 5D resulting in flattened neighborhoods. For speed, we implement haloing with 3D convolutions used as gathering operations instead of slices and concatenations.

[6]https://github.com/tensorflow/tpu/tree/master/models/official/detection

E. ImageNet-21k Models

For our ImageNet-21k transfer experiments Table 6), we make 4 changes to our HaloNet H4 model (See Table A2 for specification of the H4 model). To increase the number of parameters in the model body, We increase the base width r_w to 2.0 (Making the base width 128, twice the normal width), and we also change r_b from 3.0 to the default 4.0. We remove the final extra 1×1 convolution, so that the label embeddings have a large number of filters to account for the larger number of labels. Finally, we increase the number of layers in the second stage from 3 to 4. For the hybrid model, we use convolutions in the first two stages.

For pretraining on 256×256 images, we set $b = 8$ and $h = 2$. For finetuning on 384×384 images, we use $b = 12$, $h = 2$, and for finetuning on 512×512 size images, we use $b = 16$, $h = 1$. When transferring the pretrained model, we initialize all the parameters from the pretrained checkpoint at the final step of pretraining except for the label embeddings, which are initialized to zeros, and the relative embeddings, that are initialized by linearly interpolating from the ones learned at pretraining.

F. Impact of relative position encodings

[43] showed that using relative position was important for achieving good accuracies. We find the same outcome with HaloNet. Using absolute factorized abosolute position encodings, which are added to the activations before local self-attention in every layer, drops accuracy from to 78.6% (the first row in Table 3) to 77.5%