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 [53], 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 [39], 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, and expanded results.

C.1. ImageNet classification

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 [51] model. The image sizes can be found in Table A2.

C.2. 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 [33] 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 [60]. 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 RandAug [8] we grow our RangAug 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 RandAug magnitudes.

C.3. Transfer from ImageNet-21k

Our experiments thus far have focused on training from scratch on ImageNet-ILSVRC-2012 [43], where regularizations and longer training are critical for good accuracies. Papers such [10, 26] have shown that a short finetuning step after pretraining models on larger labelled datasets such as ImageNet-21k [9] or JFT-300M [48] 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 [26], pretraining on the public ImageNet-21k dataset, and finetuning on ImageNet. Following our observation in Table 4, we also train a hybrid version of this model with convolutions in the first two stages and one that uses linear projections of non-overlapping 4×4 patches (row 2 in Table A3, similar to ViT. Note that using linear projections with non-overlapping patches is equivalent to using a convolution where the spatial dimensions of the stride and the kernel are the same, 4×4 in this case. 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

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

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 A3. **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.

larger number of labels. We also add another layer in the second stage, increasing it from 3 to 4 For the hybrid model, we use convolutions in the first two stages. For a fair comparison with [26], we do not use squeeze-and-excitation [20] in the stages with convolutions.

ImageNet-21k contains 14.2 million annotated images, and 21k labels, both an order of magnitude larger than ImageNet. Following [26], 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 [33]. We also use a weight decay of 0.00008, and train with Nesterov's Accelerated Gradient [35, 49] during pretraining and finetuning. We pretrain on 256×256 size images and finetune on different image sizes, as shown in Table A3. Our wider H4, patch based, and hybrid-H4 models achieves better accuracy than the Vision Transformer and a $4\times$ wide ResNet-152 from [26] 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 [37], 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, as shown in Section 4.5.

C.4. Detection and instance segmentation hyperparameters

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 A4). 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 [32] with 1024×1024 size images for 32 epochs, using the Cloud TPU Detection

Model	$\mathbf{AP^{bb}}$	$\mathbf{AP^{bb}_s}$	$\mathbf{AP^{bb}_m}$	$\mathbf{AP^{bb}_l}$	$\mathbf{AP^{mk}}$	$\mathbf{AP^{mk}_s}$	$\mathbf{AP^{mk}_m}$	$\mathbf{AP^{mk}_l}$	**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 A4. **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.

Codebase [5]. We use Mask-RCNN [13] for all detection and instance segmentation experiments. We pretrain the backbone on ImageNet, mostly reusing the same hyperparameters as in Section C.2. Backbones are pretrained for 350 epochs using an image size of 512, which was chosen to be closer to the 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. 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 $[\frac{4}{5}, \frac{5}{4}]$.

In Table A4, we see that 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 [46], uses global self-attention in ResNet-Attention hybrids for structured prediction tasks and classification. See [46] for additional details on the efficacy of global attention for localization tasks

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

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 [24] 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.

E. Discussion on training speed

Figure A1 shows that pure self-attention[6] 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 [26] and ViT [10], 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 [62, 42], 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

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

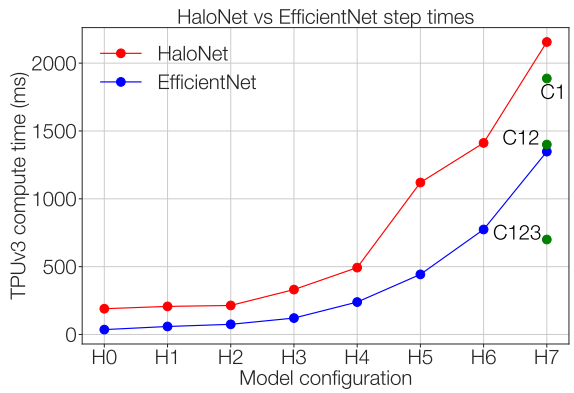

Figure A1. **Pure attention based HaloNet models are currently slower to train than efficient net models.** The times are the TPUv3 compute time needed to process a batch size of 32 per core. The points in green with annotations C1, C12, and C123 correspond to the hybrid models with convolutions in stages 1, 1–2 and 1–3 respectively. (see Table 4).

lead to better speed-accuracy tradeoffs. We leave this study for future work.

F. 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.