Bottleneck Transformers for Visual Recognition - Supplementary

Aravind Srinivas[1] Tsung-Yi Lin[2] Niki Parmar[2] Jonathon Shlens[2] Pieter Abbeel[1] Ashish Vaswani[2]
[1]UC Berkeley [2]Google Research
{aravind}@cs.berkeley.edu

A. Appendix

A.1. Code for the BoT block

We provide the exact code used for implementing the Multi-Head Self-Attention (MHSA) in 2D with relative-position encodings as well as the implementation of BoT block in this `gist` link: https://gist.github.com/aravindsrinivas/56359b79f0ce4449bcb04ab4b56a57a2.

A.2. Implementation: COCO Instance Segmentation and Object Detection

Our implementation is based on the Cloud TPU Detection codebase - https://github.com/tensorflow/tpu/tree/master/models/official/detection. Our canonical setting uses the following hyperparameters which are updated in `configs/maskrcnn_config.py`:

- `output_size` of 1024×1024.
- `aug_scale_min=0.8,aug_scale_max=1.25`
- `mrcnn_head:num_convs=4, num_filters=256,mask_target_size=28`
- `frcnn_head:num_convs=4, num_filters=256,fc_dims=1024, num_fcs=1`
- `rpn_head:min_level=2, max_level=6, anchors_per_location=3, num_convs=2,num_filters=256`
- `fpn:min_level=2,max_level=6`
- `anchor: num_scales=1,anchor_size=8, min_level=2,max_level=6`

For all experiments, we use L2 weight decay of $4e-5$, sync-batch-norm with momentum 0.997 and epsilon 1e-4. We use batch norm in the backbone, FPN, RPN head, FRCNN head and MRCNN head. We initialize backbone weights with pre-trained ImageNet checkpoints and fine-tune all the weights (including the batch-norm parameters) as specified in MoCo [6].

Table 1 presents the hyperparameters for models that we train with batch size 64 on 8 TPU-v3 cores (equivalent to DGX-1) which applies to most of our models, in particular, all the models that we train with image size 1024×1024.

Type	Epochs	Train Steps	LR Schedule
1x	12	22.5k	[15k, 17.5k]
2x	24	45k	[37.5k, 40k]
3x	36	67.5k	[60k, 65k]
6x	72	135k	[120k, 130k]

Table 1: Learning Rate Schedules for the 1x, 2x, 3x and 6x settings for models trained with global batch size of 64, 8 TPU-v3 cores, 16 GB HBM per cores, learning rate 0.1 with schedule [0.01, 0.001]. The learning rate is initially warmed up for 500 warmup steps from 0.0067.

For models that do not fit with batch size of 8 per core (for example, the ones that train with 1280×1280), we train with a global batch size of 128 on 32 TPU-v3 cores (4 images per core). The 6x schedule for these models that train with 32 cores corresponds to training with 67.5k steps (since batch size 128 is double 64), with [60k, 65k] learning rate schedule. The learning rate is 0.15 with schedule [0.015, 0.0015] with 500 warmup steps started from 0.0067.

For the model that achieves best results (44.4% mask AP) with ResNet-200 backbone and BoTNet structure for blockgroup `c5`, we use 8 convolutions in the MRCC head.

For our object detection experiments, we just turned off the mask branch by setting the `include_mask` flag as `False` and `eval_type` as `box` instead of `box_and_mask`.

A.3. BoTNet improves over ResNet for Object Detection

Does BoTNet improve upon ResNet for the task of object detection as well? We verify this through experiments with the Faster R-CNN framework (Table 2). We observe that BoTNet indeed improves over ResNet significantly. BoT

R50 improves upon R50 by a significant margin of + 1.6% AP^{bb}_S (AP^{bb} for small objects). These results suggest that *self-attention has a big effect in detecting small objects* which is considered to be an important and hard problem for deploying object detection systems in the real world. This result is in contrast with DETR [2] which observes a big improvement on large objects but not on small objects. We believe that introducing self-attention in the backbone architecture might help fix the lack of gains on small objects in DETR. Finally, we study if larger images would further benefit BoTNet for object detection. Using a multi-scale jitter of $[0.1, 2.0]$ with 72 epoch training and image size of 1280×1280, we see that BoT R152 achieves a strong performance of **48.4%** AP^{bb}.

Backbone	AP^{bb}	AP^{bb}_S	AP^{bb}_L
R50	41.5	24.9	54.3
BoT50	43.1 (+ **1.6**)	27.6 (+ **2.7**)	55.7 (+ **1.3**)
R101	42.4	24.5	55.0
BoT101	44.3 (+ **1.9**)	26.5 (+ **2.0**)	57.5 (+ **2.5**)
R152	45.1	30.1	56.6
BoT152	**48.4** (+ **3.3**)	**32.7** (+ **2.6**)	**60.6** (+ **4.0**)

Table 2: BoTNet for Faster R-CNN: The first four rows are trained for 36 epochs, jitter $[0.8, 1.25]$, image size 1024×1024. Final two rows are trained for 72 epochs with 1280×1280 image size, jitter $[0.1, 2.0]$ using `v3-32` Cloud TPU.

A.4. Why replace all three `c5` spatial convolutions?

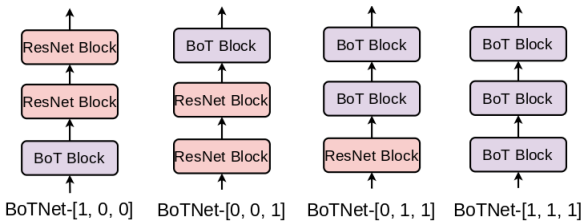

Figure 1: Replacement configs for BoTNet in the `c5` block-group of a ResNet

Is replacing all the 3 spatial convolutions in `c5` the *minimum effective* change or could it be simplified even further? We perform an ablation study on the replacement design in order to answer this question (Please refer to Table 3, Figure 1). The baseline R50 and BoT50 go by the notation `[0,0,0]` and `[1,1,1]` since the former does not replace anything while the latter replaces the spatial convolution in all three `c5` blocks. As mentioned already, the first replacement in `c5` operates on 64×64 feature map while the remaining two operate on 32×32. We ablate for the configs: `[0,0,1]`, `[0,1,1]` and `[1,0,0]`. The first two configs test how useful is the replacement when performed only on the smaller 32×32 featuremap(s) once and twice respectively, while the last tests how useful is the replacement when performed only on the larger 64×64 featuremap.

First, we see that in terms of aggregate measures such as AP^{bb} and AP^{mk}, each of the configs for BoT50 is a strict improvement over R50, with similar performance. Config. `[1,0,0]` is closer to the performance of BoT50 (`[1,1,1]`) compared to the other configurations, however a difference of 0.2 AP^{bb} is within the noise typically observed in COCO experiments. It is indeed surprising that just a single self-attention replacement layer right at the end (`[0,0,1]`) provides a visible gain of 1.3 AP^{bb}. When contrasted with the performance of R101 (43.2 AP^{bb} and 38.4 AP^{mk}), the config. `[0,0,1]` is very much competitive with 43.4 AP^{bb} and 38.6 AP^{mk}, with more efficient compute steptime on the TPU (1.2x faster). Nevertheless, the gains on large objects for the `[0,0,1]` config (+ 0.6 AP^{bb}_L) are not as significant as those in R101 (+ 1.6 AP^{bb}_L).

Among the different configs for BoT50, we see that `[1,0,0]` and `[0,1,1]` are the best in terms of good performance on both small and large objects. Surprisingly, the actual BoTNet config (`[1,1,1]`) shows significant boost on small objects (2.6 AP^{bb}_S), but does not show substantial gain on large objects, even relative to other ablation configs. We suspect this could be due to poor optimization and leave it for future work to carefully understand how self-attention affects the performance on small and large objects.

Based on these ablations, consider the question: is it better to replace convolutions with self-attention in `c5` (BoT50) vs stacking more convolution layers (R101)? An argument in favor of R101 is that the gains are clear on both small and large objects unlike BoT50 where the gains are much more on small objects. However, there does exist a BoT50 config that can strictly improve upon R101, ie the `[0,1,1]` config. It has similar properties to R101 (gain on both small and large objects), similar performance on aggregate measures like AP^{bb} and AP^{mk}, with a more efficient compute steptime. Hence, we can affirmatively say that *self-attention replacement is more efficient than stacking convolutions*.

A.5. BoT block with stride

The first block in the final block group `c5` of a ResNet runs the spatial 3×3 convolution with a stride 2 on a resolution that is 2x the height and width of the other two blocks in the group. Unlike spatial convolutional layers, the MHSA layers in BoT blocks do not implement striding. In fact, implementing strided self-attention without strided convolutions (both local and global) is an engineering challenge which we leave for future work. Our goal is to only use existing primitives. So, we adopt a very simple fix of using a local 2×2 Avg. Pooling with stride of 2 for implementing the spatial downsampling in the first BoT block. As noted in our ablation on placement of the attention (Section A.4), we see that the self-attention in the first block is very useful for

Backbone	c5-attn.	TC Time	AP^{bb}	AP^{bb}_S	AP^{bb}_L	AP^{mk}	AP^{mk}_S	AP^{mk}_L
R50	[0,0,0]	786.5	42.1	22.5	59.1	37.7	18.3	54.9
BoT 50	[0,0,1]	813.7	43.4 (+ **1.3**)	23.7 (+ **1.2**)	59.7 (+ **0.6**)	38.6 (+ **0.9**)	19.0 (+ **0.7**)	55.6 (+ **0.7**)
BoT 50	[0,1,1]	843.87	43.4 (+ **1.3**)	24.0 (+ **1.5**)	60.2 (+ **1.1**)	38.6 (+ **0.9**)	19.4 (+ **1.1**)	55.9 (+ **1.0**)
BoT R50	[1,0,0]	983.2	43.7 (+ **1.6**)	23.9 (+ **1.4**)	60.6 (+ **1.5**)	38.9 (+ **1.2**)	19.3 (+ **1.0**)	55.9 (+ **1.0**)
BoT 50	[1,1,1]	1032.7	43.6 (+ **1.5**)	25.1 (+ **2.6**)	59.4 (+ 0.3)	38.9 (+ **1.2**)	20.7 (+ **2.4**)	55.5 (+ **0.6**)
R101	[0,0,0]	928.7	43.3 (+ **1.2**)	24.2 (+ **1.7**)	60.7 (+ **1.6**)	38.4 (+ **0.7**)	19.6 (+ **1.3**)	56.8 (+ **1.9**)
BoT101	[1,1,1]	1174.9	45.5 (+ **2.2**)	26.0 (+ **1.8**)	62.3 (+ **1.6**)	40.4 (+ **2.0**)	21.1 (+ **1.5**)	58.0 (+ **1.2**)

Table 3: Ablation study on the replacement design in BoTNet: All models are trained with the canonical config with image size 1024×1024, jitter $[0.8, 1.25]$, 36 epochs. TC Time refers to TPU-v3 Compute Step Time (in milliseconds) during training.

gain on small objects. At the same time, it involves running self-attention on a resolution of 64×64, equivalent to 4096 tokens and 4096×4096 query-key matrix. We believe a well optimized strided attention implementation will drastically make this more efficient in future. The block is explained in Figure 2.

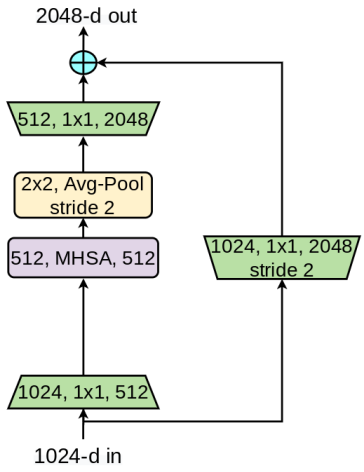

Figure 2: BoT block with stride=2, implemented in the first block of the c5 blockgroup in a BoT-ResNet. Since the incoming tensor has 1024 channels and has a stride of 16 with respect to input resolution, there is a projection shortcut with a strided 1×1 convolution. The self-attention operation in the MHSA is global and maintains the resolution. Therefore, we use local 2x2 Avg. Pooling with a stride 2 on top of it.

A.6. BoTNet-S1

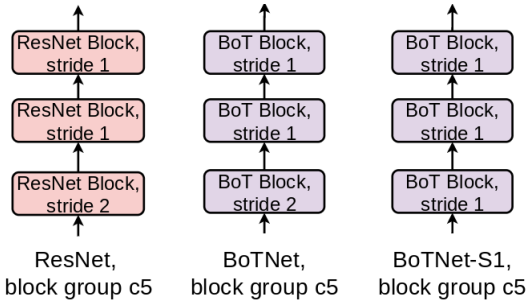

Figure 3: The c5 (final) block groups for ResNet (left), BoTNet (middle) and BoTNet-S1 (right).

A.7. Non-Local Comparison

The main paper offered a comparison between Non-Local Nets and BoTNets as well as an implementation of BoT blocks in the design of Non-Local Nets. In order to make it more clear visually, we provide an illustration of all designs in Figure 4. The figure clearly highlights the differences between inserting blocks vs replacing blocks. We note that the NL style insertion requires careful placement as prescribed in the original paper. On the other hand, simply replacing the final three blocks is a convenient design choice. Nevertheless, BoT blocks are still an improvement over vanilla NL blocks even in the insertion design, likely due to using multiple heads, relative position encodings, and value projection. As already explained in the main paper, even the *replacement* design of BoTNet performs better than the *insertion* design of vanilla NL, likely due to performing more self-attention (three vs one). Adding value projections to the NL block only improved the performance by 0.2 AP^{bb}, while the gains from using 4 heads in the MHSA layer in BoT block only helps by $\tilde{0}.2AP^{mk}$. Therefore, BoT blocks can be viewed as an improved implementation of NL blocks with relative position encodings being the main driver of the performance difference.

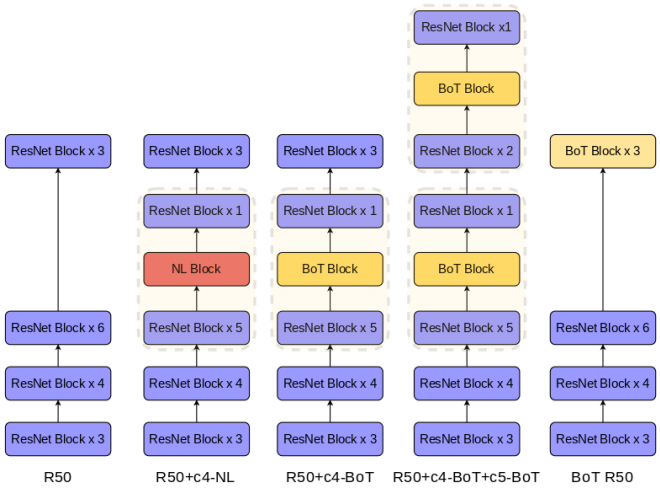

Figure 4: **First:** Regular ResNet 50 with `[3,4,6,3]` blockgroup structure. **Second:** Non-Local (NL) block *inserted* in the `c4` blockgroup of a ResNet, between the pre-final and final ResNet block, as specificed by Wang et. al [19]. **Third:** BoT block *inserted* in the same manner as a NL block with the differences between a BoT and NL block highlighted in in the main paper. **Fourth:** 2 BoT blocks, one each in `c4,c5` blockgroups *inserted* in the same manner (between pre-final and final block) as prescribed by Wang et al. [19]. **Fifth:** BoT50, where the final three ResNet blocks are *replaced* by BoT blocks.

A.8. Comparison to Squeeze-Excite

One may argue that squeeze-excitation blocks and self-attention blocks are very much related in the sense that both of them perform global aggregation and provide that as a context to convolutional models. Therefore, it is natural to ask for a comparison between the two blocks especially given that Squeeze-Excite (SE) blocks are computationally cheap and easy to use compared to BoT blocks. Table 4 presents this comparison. For fair comparison to BoTNet, we only place SE blocks in `c5` blockgroup and call this setup as R50+ c5-SE. We do not see any difference between R50 and R50 + c5-SE. However, we note that it is possible to get visible gains on top of R50 when placing SE blocks in all bottleneck blocks throughout the ResNet and not just in `c5`. Such a change (using SE in `c2,c3,c4`) is orthogonal to BoTNet and can be combined with BoTNet by using BoT blocks in `c5` and SE blocks in `c2,c3,c4`. We leave exploration of such architectures for future work.

A.9. ImageNet Test-Set Accuracy

As has been the convention since 2017, our results in the paper only report and compare the validation set (50K images) accuracy on the ImageNet benchmark. However, the EfficientNet paper [14] presented updated results on the ImageNet test-set (100K images submitted on `http://image-net.org`). We also provide the results on the ImageNet test set for the ablation setup in Table 10, to verify that there are not any surprising differences between the validation and test set (Table 5 presents similar numbers to Table 10).

Backbone	AP^{bb}	AP^{mk}
R50	42.1	37.7
BoT50	43.6 (+ **1.5**)	38.9 (+ **1.2**)
R50 + c5-SE	42.1	37.6 (- 0.1)

Table 4: Comparing R50, BoT50 and R50 + c5-SE; all 3 setups using the canonical training schedule of 36 epochs, 1024×1024 images, multi-scale jitter $[0.8, 1.25]$.

Backbone	top-1 acc.
SE50	79.3
BoT-S1-50	80.3 (+ **1.0**)

Table 5: ImageNet *test set* results in a further improved training setting with SE blocks and SiLU non-linearity: 200 epochs, batch size 4096, weight decay 4e-5, RandAugment (2 layers, magnitude 10), and label smoothing of 0.1. R50 with SE blocks is referred to as SE50.

A.10. Resolution Dependency in BoTNet

Architectures such as BoTNet and ViT that use self-attention end up adopting position encodings [17]. This in turn creates a dependency at inference time to use the same resolution that the model was trained for. For example, taking T7 trained at 384×384, and running inference with a different resolution (say 512×512) would introduce additional positional encodings. Purely convolutional architectures such as ResNets and SENets do not face this problem. We leave it for future work to investigate various design choices in making Transformer based architectures for vision, resolution independent at inference time. Some potential ideas are multi-resolution training with bilinearly interpolated positional encodings, using a spatial convolution before every self-attention operation (which could be performed without any positional encoding), etc.

While BoTNet-like hybrid architectures benefit from the spatial dependencies implicitly learned by the convolutional stack prior to attention, the reason we still use position encodings is because of the improvements that arise from using them. We also observe similar performance gains on the ImageNet benchmark. Using no position encoding at all for the BoT blocks still provides a gain of + 0.7% top-1 accuracy, but lags behind the gains from using position encodings (+ 1.2%). Further, relative position encoding is not as important in ImageNet benchmark as it is in the COCO benchmark. Absolute position encodings provide similar gains to relative

posiition encodings. We believe this is likely due to the nature of the task being less contextual (and doesn't involve precise localization) for image classification compared to detection and segmentation. We retain relative position encodings for the ImageNet experiments for consistency with the architecture used for COCO. However, for practitioners, we recommend using absolute position encodings for image classification, especially when using convolutions prior to attention, since it is faster (for training and inference) and simpler to implement.

Backbone	Pos-Enc	top-1 acc.
SE50	N/A	79.2
BoT-S1-50	-	79.9 (+ **0.7**)
BoT-S1-50	Abs	80.2 (+ **1.0**)
BoT-S1-50	Rel	80.4 (+ **1.2**)

Table 6: ImageNet (val) results in a further improved training setting with SE blocks and SiLU non-linearity: 200 epochs, batch size 4096, weight decay 4e-5, RandAugment (2 layers, magnitude 10), and label smoothing of 0.1. R50 with SE blocks is referred to as SE50.

B. BoTNet and SENet Block Configs

Please refer to Table 7 for block configs. of the various BoTNets and SENets used for the ImageNet experiments.

Model	Block Groups
ResNet-50	`[3,4,6,3]`
ResNet-101	`[3,4,23,3]`
ResNet-152	`[3,8,36,3]`
SENet-50	`[3,4,6,3]`
SENet-101	`[3,4,23,3]`
SENet-152	`[3,8,36,3]`
SENet-350	`[4,40,60,12]`
BoTNet-S1-59	`[3,4,6,6]`
BoTNet-S1-77	`[3,4,6,12]`
BoTNet-S1-110	`[3,4,23,6]`
BoTNet-S1-128	`[3,4,23,12]`

Table 7: Backbones and their block group configurations. All of them use the standard convolutional stem of the ResNet [7]. `[3,4,6,6]`, refers to the use of 3, 4, 6 and 6 blocks respectively in stages `c2,c3,c4,c5`. For BoTNet-S1 design, the final blockgroup `c5` uses a stride of 1 for all blocks. In order to reflect the improved training setting used in EfficientNets, the four BoTNet models (above) make use of SE blocks for blocks in the groups `c2,c3,c4`.

B.1. Hyperparameters

Please refer to Table 9 for the hyperparameters used for all backbones presented for ImageNet classification results in Table 8.

B.2. M.Adds and Params

In-built tools for computing parameters and FLOPs (2 * M.Adds) are often not accurate when used on architectures with `einsums`. We therefore explicitly calculated the M.Adds we report in this paper using this script https://gist.github.com/aravindsrinivas/e8a9e33425e10ed0c69c1bf726b81495.

B.2.1 Effect of SE blocks, SiLU and lower weight decay

Other aspects of improved training of backbone architectures for image classification has been the use of Squeeze-Excitation (SE) blocks [9], SiLU non-linearity [5, 12, 8] and further lowering the weight decay (for eg., EfficientNet uses 1e-5). When employing SE blocks in BoTNet and BoTNet-S1, we only do so for the ResNet blocks that employ 3×3 convolutions since self-attention is already designed for contextual global pooling. As expected, these changes lead to further improvements in the accuracy for all the models, with the gains from BoTNet remaining intact over the baseline SENet (ResNet with SE blocks) (Table 10).

The improvements from Table 10 suggest that BoTNets are a good replacement for ResNets and SENets, especially when trained with data augmentations and longer training schedules. We have also made sure to present strong baselines (eg. 79.2% top-1 acc. SENet-50).

B.3. Discussion

Figure 5 presents all the three model families (SENets, EfficientNets and BoTNets) together in one plot. As discussed in the previous sections, SENets are powerful models with strong performance that is better than EfficientNets and can be scaled all the way up to 83.8% top-1 accuracy without any bells and whistles such as ResNet-D, etc. BoTNets initially perform worse than SENets (*e.g.* T3) but begin to take over in terms of performance from T4 and strictly outperform EfficientNets, especially towards the end. EfficientNets do not scale well, particularly in the larger accuracy regime.

Recently, Transformer based models for visual recognition have become very popular since the Vision Transformer (ViT) model [4]. While our paper was developed concurrently to ViT, there has been a lot of follow-up to ViT such as DeiT [15] that have further improved the results of ViT on ImageNet-1K through regularization and distillation. As seen in the results and emphasized already, DeiT-384 is outperformed by both SENets and BoTNets currently when not considering distillation. This suggests that convolutional and hybrid (convolution and self-attention) models are still strong models to beat as far as the ImageNet benchmark goes. The message in ViT was that pure attention models without extra regularization struggle in the small data (ImageNet)

Model	Backbone	Resolution	Top-1 Acc.	Top-5 Acc.	Params	M.Adds	Steptime
B0	EfficientNet	224	77.1	93.3	5.3M	0.39B	35.6
S0	**SENet-50**	**160**	**77.0**	**93.5**	**28.02M**	**2.09B**	**32.3**
B1	**EfficientNet**	**240**	**79.1**	**94.4**	**7.8M**	**0.7B**	**58.8**
-	ResNet-50	224	78.3	94.3	25.5M	4.09B	50.7
-	ResNet-50	256	78.8	94.5	25.5M	5.34B	62.1
-	SENet-50	224	79.4	94.6	28.02M	4.09B	64.3
B2	EfficientNet	260	80.1	94.9	9.2M	1.0B	74.6
-	**ResNet-101**	**224**	**80.0**	**95.0**	**44.4M**	**7.8B**	**63.0**
B3	EfficientNet	300	81.6	95.7	12M	1.8B	120.8
T3	BoTNet-S1-59	224	81.7	95.8	33.5M	7.3B	156.2
-	ResNet-152	224	81.3	95.5	60.04M	11.5B	85.6
S1	**SENet-101**	**224**	**81.4**	**95.7**	**49.2M**	**7.8B**	**71.1**
S2	**SENet-152**	**224**	**82.2**	**95.9**	**66.6M**	**11.5B**	**97.7**
B4	EfficientNet	380	82.9	96.4	19M	4.2B	238.9
T4	BoTNet-S1-110	224	**82.8**	96.3	54.7M	10.9B	181.3
S3	**SENet-152**	**288**	**83.1**	**96.4**	**66.6M**	**19.04B**	**149.9**
B5	EfficientNet	456	83.6	96.7	30M	9.9B	442.6
S4	SENet-350	320	83.6	96.6	115.18M	52.9B	397.9
T5	**BoTNet-S1-128**	**256**	**83.5**	**96.5**	**75.1M**	**19.3B**	**355.2**
B6	EfficientNet	528	84.0	96.8	43M	19B	776.3
S5	SENet-350	384	83.8	96.6	115.18M	52.9B	397.9
T6	**BoTNet-S1-77**	**320**	**84.0**	**96.7**	**53.9M**	**23.3B**	**578.1**
B7	EfficientNet	600	84.3	97.0	66M	37B	1478.4
T7-320	**BoTNet-S1-128**	**320**	**84.2**	**96.9**	**75.1M**	**30.9B**	**634.3**
B7-RA	EfficientNet	600	84.7	97.0	66M	37B	1478.4
T7	**BoTNet-S1-128**	**384**	**84.7**	**97.0**	**75.1M**	**45.8B**	**804.5**

Table 8: Various backbone architectures, evaluated under the fair setting of using the improved training elements from EfficientNet [14]. Models are grouped by the accuracy of each EfficientNet model. For ResNets, SENets and BoTNets, the best weight decay from `[2e-5,4e-5,8e-5,1e-4]` the best dropconnect from `[0.2,None]`, and the best RandAugment magnitude from `[9,15,24]` are used to ensure fair comparisons to the already well tuned EfficientNets whose results we take from the latest update to the official codebase. The steptimes that we report, refer to the **compute** time on a TPU-v3 core, for a batch size of 32 for all the models. This is to ensure we do not use different batch sizes for different models and do not conflate TPU rematerialization for dense convolutions. Had we done so (i.e picked bigger batch sizes for ResNets and BoTNets as long as they fit on memory), the speedup gains would be even higher. Further, by **compute** time, we just mean the time spent for forward and backward passes, and **not** the data loading. This is again to ensure comparisons across models do not exploit inefficient (or non-cached) data loading. B7-RA refers to EfficientNet-B7 trained with RandAugment [3].

regime[1], but shine (achieve **85.1%** and **88.6**% fine-tuned top-1 accuracy) in the large data regime (ImageNet-21k and JFT dataset), where inductive biases such as data augmentation and regularization tricks used in the EfficientNet training setting are less important. Nevertheless, we think it is an interesting exercise to explore hybrid models such as BoTNet even in the large data regime simply because they seem to scale much better than SENets and EfficientNets (Figure 5) and achieve better performance than DeiT on ImageNet. We leave such a large scale effort for future work.

It is unclear what the right model class is, given that we have not yet explored the limits of hybrid models in the large data regimes, and that the pure attention models currently lag behind both convolutional and hybrid models in the small data regime. Nevertheless, with the hope that the ImageNet-1K benchmark has been representative of the best performing models in the vision community, BoTNets are likely to be a simple and compelling baseline to always consider. BoTNets are very much similar to the Hybrid-ViT models (R50 + ViT-B/16). The few minor differences have already been highlighted in the main paper, as to how BoTNets use Bottleneck Transformer (BoT) blocks, which are different from the regular Transformer blocks in ViT. These differences *may* lead to different optimizers and hyperparameters for training BoT and ViT models, e.g, SGD-momentum (BoT) *vs*. Adam (ViT); BatchNorm (BoT) *vs*. LayerNorm (ViT), number of non-linearities and residual connections, etc. One useful aspect of BoTNets is that they can exploit the hyperparame-

[1]ImageNet-1K may not be a small dataset by conventional standards, but is referred to as such here for the contrast with JFT.

Model	Backbone	Resolution	WD	RA	DC	DR
S0	SENet-50	160	8e-5	9	0.2	0.25
-	ResNet-50	224	8e-5	9	-	-
-	ResNet-50	256	8e-5	9	-	-
-	SENet-50	224	4e-5	15	0.2	0.25
-	ResNet-101	224	4e-5	15	0.2	0.25
T3	BoTNet-S1-59	224	4e-5	15	0.2	0.25
-	ResNet-152	224	4e-5	15	0.2	0.25
S1	SENet-101	224	4e-5	15	0.2	0.25
S2	SENet-152	224	4e-5	15	0.2	0.25
T4	BoTNet-S1-110	224	4e-5	15	0.2	0.25
S3	SENet-152	288	4e-5	15	0.2	0.25
S4	SENet-350	320	4e-5	15	0.2	0.25
T5	BoTNet-S1-128	256	4e-5	15	0.2	0.25
S5	SENet-350	384	4e-5	15	0.2	0.25
T6	BoTNet-S1-77	320	4e-5	15	0.2	0.25
T7-320	BoTNet-S1-128	320	2e-5	24	0.2	0.25
T7	BoTNet-S1-128	384	2e-5	24	0.2	0.25

Table 9: Hyperparameters for ResNets, SENets and BoTNets trained for ImageNet-1K classification benchmark reported in Table 8. Resolution, WD, RA, DC and DR refer to image size, weight-decay, randaugment magnitude (2 layers), dropconnect probability and dropout ratio respectively. Our implementation of drop-connect and the balance between RA and WD is similar to HaloNets [16]. Numbers reported for EfficientNet are just taken from the EfficientNet paper [14]. All models additionally use label smoothing of 0.1 and trained for 350 epochs. All models are trained with global batch size 4096, trained on TPU v3 hardware (v3-128), and *no* synchronized (cross-replica) batch-norm was used for training any of these models, unlike EfficientNet. We expect results to improve marginally with the use of sync. batchnorm. Training uses `bfloat16` precision and the throughput speeds reported also use `bfloat16` precision. The evaluation for all models were run on TPU v3-8 hardware with global batch size of 128 using `bfloat16` precision. All models use SGD-momentum optimizer (without Nesterov) with a momentum of 0.9. The base learning rate is 0.1 for a batch size of 256 as is standard, and is scaled to 1.6 for a batch size of 4096, with linear warmup for 5 epochs, and cosine decay for rest of the training.

Backbone	top-1 acc.	top-5 acc.
SE50	79.2	94.6
BoT50	79.6 (+ **0.4**)	94.6
BoT-S1-50	80.4 (+ **1.2**)	95.0 (+ **0.4**)

Table 10: ImageNet results in a further improved training setting with SE blocks and SiLU non-linearity: 200 epochs, batch size 4096, weight decay 4e-5, RandAugment (2 layers, magnitude 10), and label smoothing of 0.1. R50 with SE blocks is referred to as SE50.

ters that have been developed over many years for ResNets (SGD, BatchNorm, etc), resulting in a straightforward adoption in other vision tasks such as instance segmentation and object detection. These are benchmarks that have not yet seen widespread use of the Adam optimizer, LayerNorm and Transformer layers, yet. An appealing aspect of hybrid models in BoTNets over ViT-hybrids is their parameter and FLOPs efficiency. The ViT paper has not explicitly optimized for the design of hybrids. So, we believe strong conclusions cannot be drawn yet, and a careful empirical study of their differences is left for future work.

Our paper does not touch upon comparisons to backbones that employ group convolutions [20, 10], concurrent work on channel (or linear) attention in `c4` [1], axial attention [18, 13] and local attention [11], but we remark that changes prior to the `c5` blockgroup can benefit BoTNets as well. We also leave it to future research to carefully compare alternatives to self-attention such as lambda-layers [1]. We speculate that employing local attention in earlier blockgroups such as `c2,c3,c4` with the SASA [11] layers could lead to more efficient pure attention models in future. Our paper is a contribution in terms of identifying connections between ResNet MHSA blocks, Transformer blocks, and Non-Local blocks; and showing the power of simple hybrids, to achieve SoTA-competitive backbone architecture performance on both COCO and ImageNet-1K benchmarks.

References

[1] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. In *International Conference on Learning Representations*, 2021.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *arXiv preprint arXiv:2005.12872*, 2020.

Figure 5: Qualtiative Results from Mask R-CNN with BoTNet-50. Images are from the COCO dataset.

[3] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Work-*

Figure 6: Qualitative comparison between ResNet-50 and BoTNet-50 Mask R-CNN on images from the COCO dataset. While not very different, BoTNet is able to create more precise localizations as well as detect small objects better. A couple of examples are detecting the shadow of the kite, or the spoon next to the fork, etc.

shops, pages 702–703, 2020.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[5] 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.

[6] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[7] 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.

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

[9] 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.

[10] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020.

[11] 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.

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

[13] Zhuoran Shen, Irwan Bello, Raviteja Vemulapalli, Xuhui Jia, and Ching-Hui Chen. Global self-attention networks for

image recognition. *arXiv preprint arXiv:2010.03019*, 2020.

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

[15] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers and distillation through attention, 2021.

[16] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local attention for parameter efficient visual backbones. *TBD*, 2021.

[17] 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.

[18] 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.

[19] 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.

[20] 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.