Bottleneck Transformers for Visual Recognition

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

Abstract

*We present BoTNet, a conceptually simple yet powerful backbone architecture that incorporates self-attention for multiple computer vision tasks including image classification, object detection and instance segmentation. By just replacing the spatial convolutions with global self-attention in the final three bottleneck blocks of a ResNet and no other changes, our approach improves upon the baselines significantly on instance segmentation and object detection while also reducing the parameters, with minimal overhead in latency. Through the design of BoTNet, we also point out how ResNet bottleneck blocks with self-attention can be viewed as Transformer blocks. Without any bells and whistles, BoTNet achieves **44.4**% Mask AP and **49.7**% Box AP on the COCO Instance Segmentation benchmark using the Mask R-CNN framework; surpassing the previous best published single model and single scale results of ResNeSt [67] evaluated on the COCO validation set. Finally, we present a simple adaptation of the BoTNet design for image classification, resulting in models that achieve a strong performance of **84.7**% top-1 accuracy on the ImageNet benchmark while being up to **1.64x** faster in "compute"[1] time than the popular EfficientNet models on TPU-v3 hardware. We hope our simple and effective approach will serve as a strong baseline for future research in self-attention models for vision.*[2]

1. Introduction

Deep convolutional backbone architectures [37, 54, 28, 66, 56] have enabled significant progress in image classification [52], object detection [17, 40, 21, 20, 50], instance segmentation [25, 13, 27]. Most landmark backbone architectures [37, 54, 28] use multiple layers of 3×3 convolutions.

While the convolution operation can effectively capture local information, vision tasks such as object detection, instance segmentation, keypoint detection require modeling long range dependencies. For example, in instance segmentation, being able to collect and associate scene information from a large neighborhood can be useful in learning relationships across objects [32]. In order to globally aggregate the locally captured filter responses, convolution based architectures require stacking multiple layers [54, 28]. Although stacking more layers indeed improves the performance of these backbones [67], an explicit mechanism to model global (non-local) dependencies could be a more powerful and scalable solution without requiring as many layers.

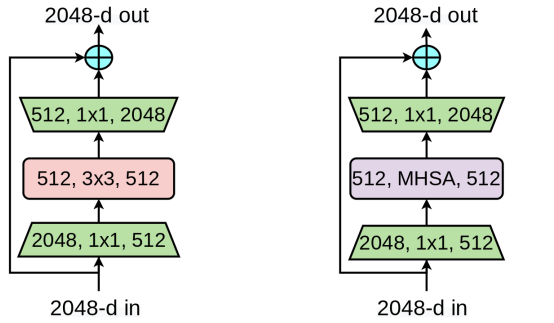

Figure 1: **Left:** A ResNet Bottleneck Block, **Right:** A Bottleneck Transformer (BoT) block. The only difference is the replacement of the spatial 3×3 convolution layer with Multi-Head Self-Attention (MHSA). The structure of the self-attention layer is described in Figure 4.

Modeling long-range dependencies is critical to natural language processing (NLP) tasks as well. Self-attention is a computational primitive [61] that implements pairwise entity interactions with a content-based addressing mechanism, thereby learning a rich hierarchy of associative features across long sequences. This has now become a standard tool in the form of Transformer [61] blocks in NLP with prominent examples being GPT [46, 5] and BERT [14, 42] models.

A simple approach to using self-attention in vision is to replace spatial convolutional layers with the multi-head self-attention (MHSA) layer proposed in the Transformer [61] (Figure 1). This approach has seen progress on two seemingly different approaches in the recent past. On the one hand, we have models such as SASA [49], AACN [4],

[1]Forward and backward propagation for batch size 32
[2]Please refer to https://arxiv.org/abs/2101.11605 for a longer version.

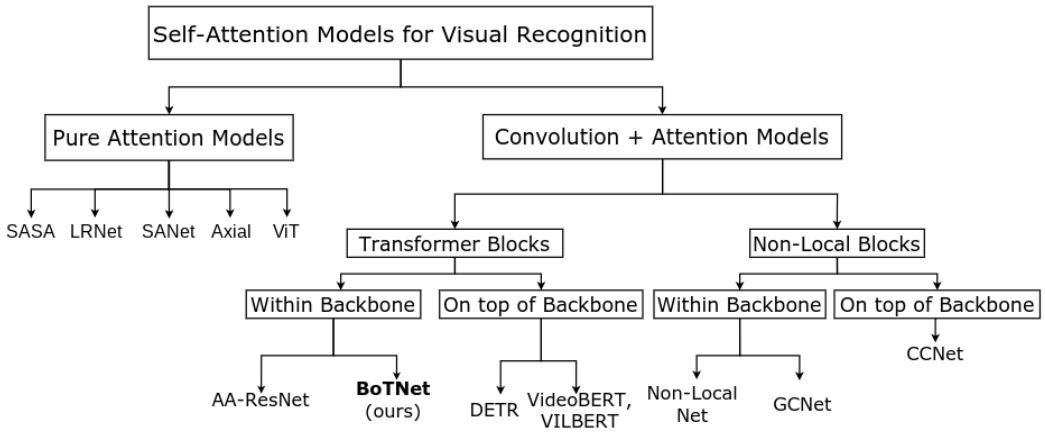

Figure 2: A taxonomy of deep learning architectures using self-attention for visual recognition. Our proposed architecture BoTNet is a hybrid model that uses both convolutions and self-attention. The specific implementation of self-attention could either resemble a Transformer block [61] or a Non-Local block [63] (difference highlighted in Figure 4). BoTNet is different from architectures such as DETR [10], VideoBERT [55], VILBERT [44], CCNet [34], etc by employing self-attention within the backbone architecture, in contrast to using them outside the backbone architecture. Being a hybrid model, BoTNet differs from pure attention models such as SASA [49], LRNet [33], SANet [68], Axial-SASA [31, 62] and ViT [15]. AA-ResNet [4] also attempted to replace a fraction of spatial convolution channels with self-attention.

SANet [68], Axial-SASA [62], etc that propose to replace spatial convolutions in ResNet botleneck blocks [28] with different forms of self-attention (local, global, vector, axial, etc). On the other hand, we have the Vision Transformer (ViT) [15], that proposes to stack Transformer blocks [61] operating on linear projections of non-overlapping patches. It may appear that these approaches present two different classes of architectures. We point out that it is *not the case*. Rather, ResNet botteneck blocks with the MHSA layer can be viewed as Transformer blocks with a bottleneck structure, modulo minor differences such as the residual connections, choice of normalization layers, etc. (Figure 3). Given this equivalence, we call ResNet bottleneck blocks with the MHSA layer as *Bottleneck Transformer* (BoT) blocks.

Here are a few challenges when using self-attention in vision: (1) Image sizes are much larger (1024×1024) in object detection and instance segmentation compared to image classification (224×224). (2) The memory and computation for self-attention scale quadratically with spatial dimensions [58], causing overheads for training and inference.

To overcome these challenges, we consider the following design: (1) Use convolutions to *efficiently* learn *abstract* and *low resolution* featuremaps from large images; (2) Use global (*all2all*) self-attention to process and aggregate the information contained in the featuremaps captured by convolutions. Such a hybrid design [4] (1) uses existing and well optimized primitives for both convolutions and all2all self-attention; (2) can deal with large images efficiently by having convolutions do the spatial downsampling and letting attention work on smaller resolutions. Here is a simple practical instantiation of this hybrid design: Replace *only* the final three bottleneck blocks of a ResNet with BoT blocks *without any other changes*. Or in other words, take a ResNet and only replace the final three 3×3 convolutions with MHSA layers (Fig 1, Table 1). This simple change improves the mask AP by 1.2% on the COCO instance segmentation benchmark [40] over our canonical baseline that uses ResNet-50 in the Mask R-CNN framework [27] with *no hyperparameter differences* and minimal overheads for training and inference. Moving forward, we call this simple instantiation as BoTNet given its connections to the Transformer through the BoT blocks. While we note that there is no novelty in its construction, we believe the simplicity and performance make it a useful reference backbone architecture that is worth studying.

Using BoTNet, we demonstrate significantly improved results on instance segmentation *without any bells and whistles* such as Cascade R-CNN [7], FPN changes [41, 19, 43, 57], hyperparameter changes [56], etc. A few key results from BoTNet are: (1) Performance gains across various training configurations (Section 4.1), data augmentations (Section 4.2) and ResNet family backbones (Section 4.4); (2) Significant boost from BoTNet on small objects (+2.4 Mask AP and +2.6 Box AP) (Appendix); (3) Performance gains over Non-Local layers (Section 4.6); (4) Gains that scale well with larger images resulting in **44.4%** mask AP, competitive with state-of-the-art performance among entries that only study backbone architectures with modest training schedules (up to 72 epochs) and no extra data or augmentations.[3].

[3]SoTA is based on https://paperswithcode.com/sota/instance-segmentation-on-coco-minival.

Lastly, we scale BoTNets, taking inspiration from the training and scaling strategies in [56, 49, 38, 51, 48, 67, 3], after noting that BoTNets do not provide substantial gains in a smaller scale training regime. We design a family of BoTNet models that achieve up to **84.7%** top-1 accuracy on the ImageNet validation set, while being upto **1.64x** faster than the popular EfficientNet models in terms of *compute* time on TPU-v3 hardware. By providing *strong results* through BoTNet, we hope that self-attention becomes a widely used primitive in future vision architectures.

2. Related Work

A taxonomy of deep learning architectures that employ self-attention for vision is presented in Figure 2. In this section, we focus on: (1) Transformer vs BoTNet; (2) DETR vs BoTNet; (3) Non-Local vs BoTNet.

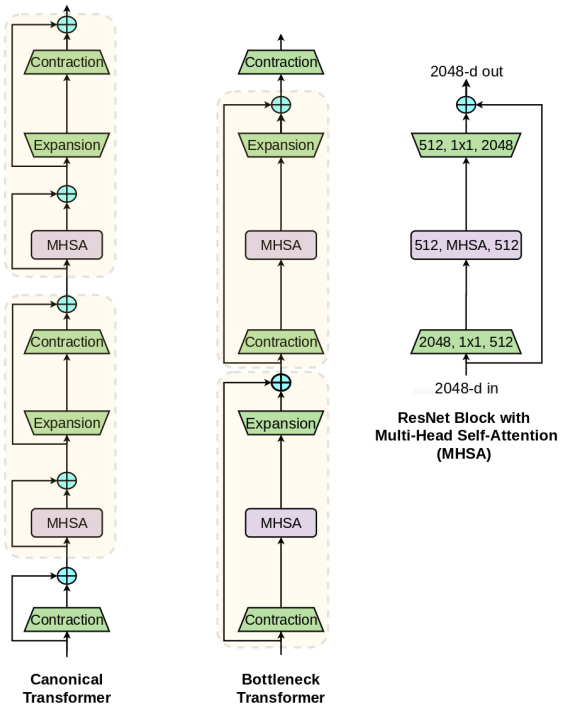

Figure 3: **Left:** Canonical view of the Transformer with the boundaries depicting the definition of a Transformer block as described in Vaswani et. al [61]. **Middle:** Bottleneck view of the Transformer with boundaries depicting what we define as the Bottleneck Transformer (BoT) block in this work. The architectural structure that already exists in the Transformer can be interpreted a ResNet bottleneck block [28] with Multi-Head Self-Attention (MHSA) [61] with a different notion of block boundary as illustrated. **Right:** An instantiation of the Bottleneck Transformer as a ResNet bottleneck block [28] with the difference from a canonical ResNet block being the replacement of 3×3 convolution with MHSA.

Connection to the Transformer: As the title of the paper suggests, one key message in this paper is that ResNet bottleneck blocks with Multi-Head Self-Attention (MHSA) layers can be viewed as Transformer blocks with a bottleneck structure. This is visually explained in Figure 3 and we name this block as Bottleneck Transformer (BoT). We note that the architectural design of the BoT block is not our contribution. Rather, we point out the relationship between MHSA ResNet bottleneck blocks and the Transformer with the hope that it improves our understanding of architecture design spaces [47, 48] for self-attention in computer vision. There are still a few differences aside from the ones already visible in the figure (residual connections and block boundaries): (1) Normalization: Transformers use Layer Normalization [1] while BoT blocks use Batch Normalization [35] as is typical in ResNet bottleneck blocks [28]; (2) Non-Linearities: Transformers use one non-linearity in the FFN block, while the ResNet structure allows BoT block to use three non-linearities; (3) Output projections: The MHSA block in a Transformer contains an output projection while the MHSA layer (Fig 4) in a BoT block (Fig 1) does not; (4) We use the SGD with momentum optimizer typically used in computer vision [28, 27, 22] while Transformers are generally trained with the Adam optimizer [36, 61, 10, 15].

Connection to DETR: Detection Transformer (DETR) is a detection framework that uses a Transformer to implicitly perform region proposals and localization of objects instead of using an R-CNN [21, 20, 50, 27]. Both DETR and BoTNet attempt to use self-attention to improve the performance on object detection and instance (or panoptic) segmentation. The difference lies in the fact that DETR uses Transformer blocks outside the backbone architecture with the motivation to get rid of region proposals and non-maximal suppression for simplicity. On the other hand, the goal in BoTNet is to provide a backbone architecture that uses Transformer-like blocks for detection and instance segmentation. We are agnostic to the detection framework (be it DETR or R-CNN). We perform our experiments with the Mask [27] and Faster R-CNN [50] systems and leave it for future work to integrate BoTNet as the backbone in the DETR framework. With visibly good gains on small objects in BoTNet, we believe there maybe an opportunity to address the lack of gain on small objects found in DETR, in future (refer to Appendix).

Connection to Non-Local Neural Nets:[4] Non-Local (NL) Nets [63] make a connection between the Transformer and the Non-Local-Means algorithm [6]. They insert NL blocks into the final one (or) two blockgroups (`c4,c5`) in a ResNet and improve the performance on video recognition and instance segmentation. Like NL-Nets [63, 8], BoTNet is a hybrid design using convolutions and global self-attention.

[4]The replacement vs insertion contrast has previously been pointed out in AA-ResNet (Bello et. al) [4]. The difference in our work is the complete replacement as opposed to fractional replacement in Bello et al.

(1) Three differences between a NL layer and a MHSA layer (illustrated in Figure 4): use of multiple heads, value projection and position encodings in MHSA; (2) NL blocks use a bottleneck with channel factor reduction of 2 (instead of 4 in BoT blocks which adopt the ResNet structure); (3) NL blocks are *inserted* as *additional* blocks into a ResNet backbone as opposed to *replacing* existing convolutional blocks as done by BoTNet. Section 4.6 offers a comparison between BoTNet, NLNet as well as a NL-like version of BoTNet where we *insert* BoT blocks in the same manner as NL blocks instead of replacing.

3. Method

stage	output	ResNet-50	**BoTNet-50**
c1	512 × 512	7×7, 64, stride 2	7×7, 64, stride 2
c2	256 × 256	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times3$
c3	128 × 128	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times4$
c4	64 × 64	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times6$
c5	32 × 32	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times3$	$\begin{bmatrix} 1\times1, 512 \\ \texttt{MHSA}, 512 \\ 1\times1, 2048 \end{bmatrix} \times3$
# params.		**25.5**×10^6	**20.8**×10^6
M.Adds		**85.4**×10^9	**102.98**×10^9
TPU steptime		**786.5** ms	**1032.66** ms

Table 1: Architecture of BoTNet-50 (BoT50): The only difference in BoT50 from ResNet-50 (R50) is the use of MHSA layer (Figure 4) in c5. For an input resolution of 1024 × 1024, the MHSA layer in the first block of c5 operates on 64 × 64 while the remaining two operate on 32 × 32. We also report the parameters, multiply-adds (m. adds) and training time throughput (TPU-v3 steptime on a v3-8 Cloud-TPU). BoT50 has only 1.2x more m.adds. than R50. The overhead in training throughout is 1.3x. BoT50 also has 1.2x *fewer* parameters than R50. While it may appear that it is simply the aspect of performing slightly more computations that might help BoT50 over the baseline, we show that it is not the case in Section 4.4.

BoTNet by design is simple: replace the final three spatial (3 × 3) convolutions in a ResNet with Multi-Head Self-Attention (MHSA) layers that implement global (*all2all*) self-attention over a 2D featuremap (Fig 4). A ResNet typically has 4 stages (or blockgroups) commonly referred to as [c2,c3,c4,c5] with strides [4,8,16,32] relative to the input image, respectively. Stacks [c2,c3,c4,c5] consist of multiple *bottleneck* blocks with residual connections (e.g, R50 has [3,4,6,3] bottleneck blocks).

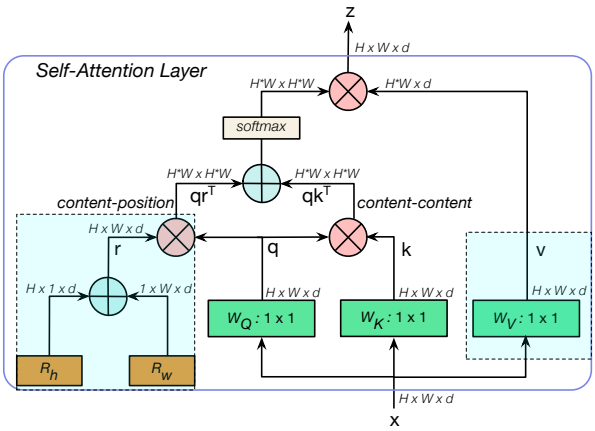

Figure 4: Multi-Head Self-Attention (MHSA) layer used in the BoT block. While we use 4 heads, we do not show them on the figure for simplicity. all2all attention is performed on a 2D featuremap with *split* relative position encodings R_h and R_w for height and width respectively. The attention logits are $qk^T + qr^T$ where q, k, r represent query, key and position encodings respectively (we use relative distance encodings [53, 4, 49]). $\bigoplus$ and $\bigotimes$ represent element wise sum and matrix multiplication respectively, while 1×1 represents a pointwise convolution. Along with the use of multiple heads, the highlighted blue boxes (position encodings and the value projection are the *only* three elements that are not present in the Non-Local Layer [63, 65].

Approaches that use self-attention throughout the backbone [49, 4, 68, 15] are feasible for input resolutions (224 × 224 (for classification) and 640 × 640 (for detection experiments in SASA [49])) considered in these papers. Our goal is to use attention in more realistic settings of high performance instance segmentation models, where typically images of larger resolution (1024 × 1024) are used. Considering that self-attention when performed globally across n entities requires $O(n^2d)$ memory and computation [61], we believe that the simplest setting that adheres to the above factors would be to incorporate self-attention at the lowest resolution featuremaps in the backbone, ie, the residual blocks in the c5 stack. The c5 stack in a ResNet backbone typically uses 3 blocks with one spatial 3 × 3 convolution in each. Replacing them with MHSA layers forms the basis of the BoTNet architecture. The first block in c5 uses a 3 × 3 convolution of stride 2 while the other two use a stride of 1. Since all2all attention is not a strided operation, we use a 2 × 2 average-pooling with a stride 2 for the first BoT block. The BoTNet architecture is described in Table 1 and the MHSA layer is presented in Figure 4. The strided

version of the BoT block is presented in the Appendix.

Relative Position Encodings: In order to make the attention operation *position aware*, Transformer based architectures typically make use of a position encoding [61]. It has been observed lately that *relative-distance-aware* position encodings [53] are better suited for vision tasks [4, 49, 68]. This can be attributed to attention not only taking into account the content information but also relative distances between features at different locations, thereby, being able to effectively associate information across objects with positional awareness. In BoTNet, we adopt the 2D relative position self-attention implementation from [49, 4].

4. Experiments

We study the benefits of BoTNet for instance segmentation and object detection. We perform a thorough ablation study of various design choices through experiments on the COCO dataset [40]. We report the standard COCO metrics including the AP^{bb} (averaged over IoU thresholds), AP^{bb}_{50}, AP^{bb}_{75}, AP^{mk}; AP^{mk}_{50}, AP^{mk}_{75} for box and mask respectively. As is common practice these days, we train using the COCO `train` set and report results on the COCO `val` (or `minival`) set as followed in Detectron [22][5]. Our experiments are based on the Google Cloud TPU detection codebase[6]. We run all the baselines and ablations with the same codebase. Unless explicitly specified, our training infrastructure uses `v3-8` Cloud-TPU which contains 8 cores with 16 GB memory per core. We train with the `bfloat16` precision and cross-replica batch normalization [35, 64, 27, 22, 45] using a batch size of 64.

4.1. BoTNet improves over ResNet on COCO Instance Segmentation with Mask R-CNN

We consider the simplest and most widely used setting: ResNet-50[7] backbone with FPN[8]. We use images of resolution 1024×1024 with a multi-scale jitter of $[0.8, 1.25]$ (scaling the image dimension between 820 and 1280, in order to be consistent with the Detectron setting of using 800×1300). In this setting, we benchmark both the ResNet-50 (R50) and BoT ResNet-50 (BoT50) as the backbone architectures for multiple training schedules: **1x:** 12 epochs, **2x:** 24 epochs, **3x**: 36 epochs, **6x**: 72 epochs[9], all using the same hyper-parameters for both the backbones across all the training schedules (Table 2). We clearly see that BoT50 is a significant improvement on top of R50 barring the 1x schedule (12 epochs). This suggests that BoT50 warrants longer training in order to show significant improvement over R50. We also see that the improvement from BoT50 in the 6x schedule (72 epochs) is worse than its improvement in the 3x schedule (32 epochs). This suggests that training much longer with the default scale jitter hurts. We address this by using a more aggressive scale jitter (Section 4.2).

Backbone	epochs	AP^{bb}	AP^{mk}
R50	12	39.0	35.0
BoT50	12	39.4 (+ **0.4**)	35.3 (+ **0.3**)
R50	24	41.2	36.9
BoT50	24	42.8 (+ **1.6**)	38.0 (+ **1.1**)
R50	36	42.1	37.7
BoT50	36	43.6 (+ **1.5**)	38.9 (+ **1.2**)
R50	72	42.8	37.9
BoT50	72	43.7 (+ **0.9**)	38.7 (+ **0.8**)

Table 2: Comparing R50 and BoT50 under the 1x (12 epochs), 3x (36 epochs) and 6x (72 epochs) settings, trained with image resolution 1024×1024 and multi-scale jitter of $[0.8, 1.25]$.

4.2. Scale Jitter helps BoTNet more than ResNet

Backbone	`jitter`	AP^{bb}	AP^{mk}
R50	$[0.8, 1.25]$	42.8	37.9
BoT50	$[0.8, 1.25]$	43.7 (+ **0.9**)	38.7 (+ **0.8**)
R50	$[0.5, 2.0]$	43.7	39.1
BoT50	$[0.5, 2.0]$	45.3 (+ **1.8**)	40.5 (+ **1.4**)
R50	$[0.1, 2.0]$	43.8	39.2
BoT50	$[0.1, 2.0]$	45.9 (+ **2.1**)	40.7 (+ **1.5**)

Table 3: Comparing R50 and BoT50 under three settings of multi-scale jitter, all trained with image resolution 1024×1024 for 72 epochs (6x training schedule).

In Section 4.1, we saw that training much longer (72 epochs) reduced the gains for BoT50. One way to address this is to increase the amount of multi-scale jitter which has been known to improve the performance of detection and segmentation systems [16, 18]. Table 3 shows that BoT50 is significantly better than R50 (+ 2.1% on AP^{bb} and + 1.7% on AP^{mk}) for multi-scale jitter of $[0.5, 2.0]$, while also showing significant gains (+ 2.2% on AP^{bb} and + 1.6% on AP^{mk}) for scale jitter of $[0.1, 2.0]$, suggesting that BoTNet (self-attention) benefits more from extra augmentations such as multi-scale jitter compared to ResNet (pure convolutions).

[5] `train` - 118K images, `val` - 5K images

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

[7] We use the ResNet backbones pre-trained on ImageNet classification as is common practice. For BoTNet, the replacement layers are **not** pre-trained but randomly initialized for simplicity; the remaining layers are initialized from a pre-trained ResNet.

[8] FPN refers to Feature Pyramid Network [39]. We use it in every experiment we report results on, and our FPN levels from 2 to 6 (`p2` to `p6`) similar to Detectron [22].

[9] 1x, 2x, 3x and 6x convention is adopted from MoCo [26].

4.3. Relative Position Encodings Boost Performance

BoTNet uses relative position encodings [53]. We present an ablation for the use of relative position encodings by benchmarking the individual gains from content-content interaction (qk^T) and content-position interaction (qr^T) where q, k, r represent the query, key and relative position encodings respectively. The ablations (Table 4) are performed with the canonical setting[10]. We see that the gains from qr^T and qk^T are complementary with qr^T more important, ie, qk^T standalone contributes to 0.6% AP^{bb} and 0.6% AP^{mk} improvement over the R50 baseline, while qr^T standalone contributes to 1.0% AP^{bb} and 0.7 % AP^{mk} improvement. When combined together ($qk^T + qr^T$), the gains on both AP^{bb} and AP^{mk} are additive (1.5% and 1.2% respectively). We also see that using absolute position encodings (qr^T_{abs}) does not provide as much gain as relative. This suggests that introducing relative position encodings into architectures like DETR [10] is an interesting direction for future work.

Backbone	Att. Type	AP^{bb}	AP^{mk}
R50	-	42.1	37.7
BoT50	qk^T	42.7 (+ **0.6**)	38.3 (+ **0.6**)
BoT50	qr^T_{relative}	43.1 (+ **1.0**)	38.4 (+ **0.7**)
BoT50	$qk^T + qr^T_{\text{relative}}$	43.6 (+ **1.5**)	38.9 (+ **1.2**)
BoT50	$qk^T + qr^T_{\text{abs}}$	42.5 (+ **0.4**)	38.1 (+ **0.4**)

Table 4: Ablation for Relative Position Encoding: Gains from the two types of interactions in the MHSA layers, content-content (qk^T) and content-position (qr^T).

4.4. BoTNet improves backbones in ResNet Family

How well does the replacement setup of BoTNet work for other backbones in the ResNet family? Table 5 presents the results for BoTNet with R50, R101, and R152. All these experiments use the canonical training setting (refer to footnote in 4.3). These results demonstrate that BoTNet is applicable as a drop-in replacement for any ResNet backbone. Note that BoT50 is better than R101 (+ 0.3% AP^{bb}, **+ 0.5%** AP^{mk}) while it is competitive with R152 on AP^{mk}. Replacing 3 spatial convolutions with `all2all` attention gives more improvement in the metrics compared to stacking 50 more layers of convolutions (R101), and is competitive with stacking 100 more layers (R152), supporting our initial hypothesis that long-range dependencies are better captured through attention than stacking convolution layers.[11]

[10] `res:1024x1024, 36 epochs (3x schedule), multi-scale jitter:[0.8, 1.25]`

[11] Note that while one may argue that the improvements of BoT50 over R50 could be attributed to having 1.2x more M. Adds, BoT50 (121×10^9 M.Adds) is also better than R101 (162.99×10^9 B M. Adds and is competitive with R152 (240.56×10^9 M. Adds) despite performing significantly less computation.

Backbone	AP^{bb}	AP^{mk}
R50	42.1	37.7
BoT50	43.6 (+ **1.5**)	38.9 (+ **1.2**)
R101	43.3	38.4
BoT101	45.5 (+ **2.2**)	40.4 (+ **2.0**)
R152	44.2	39.1
BoT152	46.0 (+ **1.8**)	40.6 (+ **1.5**)

Table 5: Comparing R50, R101, R152, BoT50, BoT101 and BoT152; all 6 setups using the canonical training schedule of 36 epochs, 1024×1024 images, multi-scale jitter $[0.8, 1.25]$.

4.5. BoTNet scales well with larger images

We benchmark BoTNet as well as baseline ResNet when trained on 1280×1280 images in comparison to 1024×1024 using the best config: multi-scale jitter of $[0.1, 2.0]$ and training for 72 epochs. Results are presented in Tables 6 and 8. Results in Table 6 suggest that BoTNet benefits from training on larger images for all of R50, R101 and R152. BoTNet trained on 1024×1024 (leave alone 1280×1280) is significantly better than baseline ResNet trained on 1280×1280. Further, BoT200 trained with 1280×1280 achieves a AP^{bb} of **49.7%** and AP^{mk} of **44.4%**. We believe this result highlights the power of self-attention, in particular, because it has been achieved without any bells and whistles such as modified FPN [41, 19, 16, 57], cascade RCNN [7], etc. This result surpasses the previous best published single model single scale instance segmentation result from ResNeSt [67] evaluated on the COCO `minival` (44.2% AP^{mk}).

Backbone	`res`	AP^{bb}	AP^{mk}
R50	1280	44.0	39.5
BoT50	1024	45.9 (+ **1.9**)	40.7 (+ **1.2**)
BoT50	1280	46.1 (+ **2.1**)	41.2 (+ **1.8**)
R101	1280	46.4	41.2
BoT101	1024	47.4 (+ **1.0**)	42.0 (+ **0.8**)
BoT101	1280	47.9 (+ **1.5**)	42.4 (+ **1.2**)

Table 6: All the models are trained for 72 epochs with a multi-scale jitter of $[0.1, 2.0]$.

4.6. Comparison with Non-Local Neural Networks

How does BoTNet compare to Non-Local Neural Networks? NL ops are *inserted* into the `c4` stack of a ResNet backbone between the pre-final and final bottleneck blocks. This *adds* more parameters to the model, whereas BoTNet ends up reducing the model parameters (Table 5). In the NL mould, we add ablations where we introduce BoT block in the exact same manner as the NL block. We also run an

Backbone	Change in backbone	AP^{bb}	AP^{mk}
R50	-	42.1	37.7
R50 + NL [63]	+ 1 NL block in `c4`	43.1	38.4
R50 + BoT (c4)	+ 1 BoT block in `c4`	43.7	38.9
R50 + BoT (c4, c5)	+ 2 BoT blocks in `c4,c5`	44.9	39.7
BoT50	Replacement in `c5`	43.6	38.9

Table 7: Comparison between BoTNet and Non-Local (NL) Nets: All models trained for 36 epochs with image size 1024×1024, jitter $[0.8, 1.25]$.

Backbone	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
BoT152	49.5	71.0	54.2	43.7	68.2	47.4
BoT200	**49.7**	**71.3**	**54.6**	**44.4**	**68.9**	**48.2**

Table 8: BoT152 and BoT200 trained for 72 epochs with a multi-scale jitter of $[0.1, 2.0]$.

ablation with the insertion of two BoT blocks, one each in the `c4,c5` stacks. Results are presented in Table 7. Adding a NL improves AP^{bb} by 1.0 and AP^{bb} by 0.7, while adding a BoT block gives +1.6 AP^{bb} and +1.2 AP^{mk} showing that BoT block design is better than NL. Further, BoT-R50 (which replaces instead of adding new blocks) provides +1.5 AP^{bb} and + 1.2 AP^{mk}, as good as adding another BoT block and better than adding one additional NL block.

4.7. Image Classification on ImageNet

4.7.1 BoTNet-S1 architecture

While we motivated the design of BoTNet for detection and segmentation, it is a natural question to ask whether the BoTNet architecture design also helps improve the image classification performance on the ImageNet [52] benchmark. Prior work [65] has shown that *adding* Non-Local blocks to ResNets and training them using canonical settings does *not* provide substantial gains. We observe a similar finding for BoTNet-50 when contrasted with ResNet-50, with both models trained with the canonical hyperparameters for ImageNet [48]: 100 epochs, batch size 1024, weight decay 1e-4, standard ResNet data augmentation, cosine learning rate schedule (Table 9). BoT50 does *not* provide significant gains over R50 on ImageNet though it does provide the benefit of reducing the parameters while maintaining comparable computation (M.Adds).

A simple method to fix this lack of gain is to take advantage of the image sizes typically used for image classification. In image classification, we often deal with much smaller image sizes (224×224) compared to those used in object detection and segmentation (1024×1024). The featuremaps on which the BoT blocks operate are hence much smaller (e.g 14×14, 7×7) compared to those in instance segmentation and detection (e.g 64×64, 32×32). With the same number of parameters, and, without a significant increase in computation, the BoTNet design in the `c5` blockgroup can be changed to uniformly use a stride of 1 in all the final MHSA layers. We call this design as BoTNet-S1 (S1 to depict stride 1 in the final blockgroup). We note that this architecture is similar in design to the hybrid models explored in Vision Transformer (ViT) [15] that use a ResNet up to stage `c4` prior to stacking Transformer blocks. The main difference between BoTNet-S1 and the hybrid ViT models lies in the use of BoT blocks as opposed to regular Transformer blocks (other differences being normalization layer, optimizer, etc as mentioned in the contrast to Transformer in Related Work (Sec. 2). The architectural distinction amongst ResNet, BoTNet and BoTNet-S1, in the final blockgroup, is visually explained in the Appendix). The strided BoT block is visually explained in the Appendix.

4.7.2 Evaluation in the standard training setting

We first evaluate this design for the 100 epoch setting along with R50 and BoT50. We see that BoT-S1-50 improves on top of R50 by 0.9% in the regular setting (Table 9). This improvement does however come at the cost of more computation (m.adds). Nevertheless, the improvement is a promising signal for us to design models that scale well with larger images and improved training conditions that have become more commonly used since EfficientNets [56].

Backbone	M.Adds	Params	top-1 acc.
R50	3.86G	25.5M	76.8
BoT50	3.79G	20.8M	77.0 (+0.2)
BoT-S1-50	4.27G	20.8M	77.7 (+ **0.9**)

Table 9: ImageNet results in regular training setting: 100 epochs, batch size 1024, weight decay 1e-4, standard ResNet augmentation, for all three models.

4.7.3 Effect of data augmentation and longer training

We saw from our instance segmentation experiments that BoTNet and self-attention benefit more from regularization such as data augmentation (in the case of segmentation, increased multi-scale jitter) and longer training. It is natural to expect that the gains from BoT and BoT-S1 could improve when training under an improved setting: 200 epochs, batch size 4096, weight decay 8e-5, RandAugment (2 layers, magnitude 10), and label smoothing of 0.1. In line with our intuition, the gains are much more significant in this setting for both BoT50 (+ 0.6%) and BoT-S1-50 (+ 1.4%) compared to the baseline R50 (Table 10).

Backbone	top-1 acc.	top-5 acc.
R50	77.7	93.9
BoT50	78.3 (+ **0.6**)	94.2 (+ **0.3**)
BoT-S1-50	79.1 (+ **1.4**)	94.4 (+ **0.5**)

Table 10: ImageNet results in an improved training setting: 200 epochs, batch size 4096, weight decay 8e-5, RandAugment (2 layers, magnitude 10), and label smoothing of 0.1

4.7.4 Scaling BoTNets

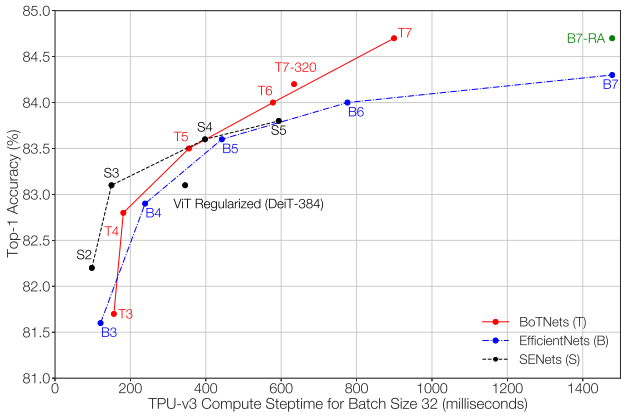

Figure 5: All backbones along with ViT and DeiT summarized in the form of scatter-plot and Pareto curves. SENets and BoTNets were trained while the accuracy of other models have been reported from corresponding papers.

The previous ablations show the BoNets performance with a ResNet-50 backbone and 224×224 image resolution. Here we study BoTNets when scaling up the model capacity and image resolution. There have been several works improving the performance of ConvNets on ImageNet [67, 56, 3]. Bello *et al*. [3] recently propose scaling strategies that mainly increase model depths and increase the image resolutions much slower compared to the compound scaling rule proposed in EfficientNets [56]. We use similar scaling rules and design a family of BoTNets. The details of model depth and image resolutions are in the Appendix. We compare to the SENets baseline to understand the impact of the BoT blocks. The BoTNets and SENets experiments are performed under the same training settings (*e.g*., regularization and data augmentation). We additionally show EfficientNet and DeiT [60] (regularized version of ViT [15])[12] to understand the performance of BoTNets compared with popular ConvNets and Transformer models. EfficientNets and DeiT are trained under strong data augmentation, model regularization, and long training schedules, similar to the training settings of BoTNets in the experiments.

ResNets and SENets are strong baselines until 83% top-1 accuracy. ResNets and SENets achieve strong performance in the improved EfficientNet training setting. BoTNets T3 and T4 *do not* outperform SENets, while T5 does perform on par with S4. This suggests that pure convolutional models such as ResNets and SENets are still the best performing models until an accuracy regime of 83%. **BoTNets scale better beyond 83% top-1 accuracy.** While SENets are a powerful model class outperforms BoTNets (up to T4), we found gains to diminish beyond SE-350 (350 layer SENet described in Appendix) trained with image size 384. This model is referred to as S5 and achieves 83.8% top-1 accuracy. On the other hand, BoTNets scale well to larger image sizes (corroborating with our results in instance segmentation when the gains from self-attention were much more visible for larger images). In particular, T7 achieves 84.7% top-1 acc., matching the accuracy of B7-RA, with a **1.64x** speedup in efficiency. BoTNets perform better than ViT-regularized (DeiT-384), showing the power of hybrid models that make use of both convolutions and self-attention compared to pure attention models on ImageNet-1K.

[12] ViT refers to Vision Transformer [15], while DeiT refers to Data-Efficient Image Transformer [60]. DeiT can be viewed as a regularized version of ViT with augmentations, better training hyperparameters tuned for ImageNet, and knowledge distillation [30]. We do not compare to the distilled version of DeiT since it's an orthogonal axis of improvement applicable to all models.

5. Conclusion

The design of vision backbone architectures that use self-attention is an exciting topic. We hope that our work helps in improving the understanding of architecture design in this space. Incorporating self-attention for other computer vision tasks such as keypoint detection [9] and 3D shape prediction [23]; studying self-attention architectures for self-supervised learning in computer vision [29, 26, 59, 11, 24, 12]; and scaling to much larger datasets such as JFT, YFCC and Instagram, are ripe avenues for future research. Comparing to, and incorporating alternatives to self-attention such as lambda-layers [2] is an important future direction as well.

6. Acknowledgements

We thank Ilija Radosavovic for several useful discussions; Pengchong Jin and Xianzhi Du for help with the TF Detection codebase; Irwan Bello, Barret Zoph, Neil Houlsby, Alexey Dosovitskiy for feedback. We thank Zak Stone for extensive compute support throughout this project the through TFRC program providing Google Cloud TPUs (https://www.tensorflow.org/tfrc).

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*,

2016.

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

[3] Irwan Bello, William Fedus, Xianzhi Du, Ekin D. Cubuk, Aravind Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting ResNets: Improved Training and Scaling Strategies, 2021.

[4] 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.

[5] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[6] 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.

[7] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018.

[8] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[9] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017.

[10] 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.

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[12] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

[13] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] 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.

[16] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V Le, and Xiaodan Song. Spinenet: Learning scale-permuted backbone for recognition and localization. *arXiv preprint arXiv:1912.05027*, 2019.

[17] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[18] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D. Cubuk, Quoc V. Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation, 2020.

[19] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7036–7045, 2019.

[20] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[21] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[22] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron, 2018.

[23] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9785–9795, 2019.

[24] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[25] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014.

[26] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2019.

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

[28] 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.

[29] Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.

[30] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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

[32] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3588–3597, 2018.

[33] 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.

[34] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 603–612, 2019.

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

[36] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[38] Jungkyu Lee, Taeryun Won, Tae Kwan Lee, Hyemin Lee, Geonmo Gu, and Kiho Hong. Compounding the performance improvements of assembled techniques in a convolutional neural network. *arXiv preprint arXiv:2001.06268*, 2020.

[39] 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.

[40] 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.

[41] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, 2018.

[42] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[43] Yudong Liu, Yongtao Wang, Siwei Wang, TingTing Liang, Qijie Zhao, Zhi Tang, and Haibin Ling. Cbnet: A novel composite backbone network architecture for object detection. *arXiv preprint arXiv:1909.03625*, 2019.

[44] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, pages 13–23, 2019.

[45] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6181–6189, 2018.

[46] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[47] Ilija Radosavovic, Justin Johnson, Saining Xie, Wan-Yen Lo, and Piotr Dollár. On network design spaces for visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1882–1890, 2019.

[48] 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.

[49] 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.

[50] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[51] Tal Ridnik, Hussam Lawen, Asaf Noy, Emanuel Ben Baruch, Gilad Sharir, and Itamar Friedman. Tresnet: High performance gpu-dedicated architecture. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1400–1409, 2021.

[52] 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.

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

[54] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[55] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7464–7473, 2019.

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

[57] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. *arXiv preprint arXiv:1911.09070*, 2019.

[58] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*, 2020.

[59] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

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

[61] 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.

[62] 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.

[63] 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.

[64] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[65] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 501–509, 2019.

[66] 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.

[67] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Mueller, R. Manmatha, Mu Li, and Alexander Smola. Resnest: Split-attention networks, 2020.

[68] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition, 2020.