Novel Approach for Object Recognition using Self Attention Networks: ORSAN

Jaswinder singh#, B. K. Sharma*

w.s.jaswinder@gmail.com, Dr. A.P.J. Abdul Kalam Technical University, India
*drbksharma@nitratextile.org, NITRA Technical Campus, Ghaziabad, India

Article Info
Page Number: 664 - 676
Publication Issue:
Vol 70 No. 2 (2021)

Article History
Article Received*: 05 September 2021*
Revised*: 09 October 2021*
Accepted*: 22 November 2021*
Publication*: 26 December 2021*

Abstract
We propose BoTrNe, a theoretically simple but strong backbone architecture for various computer vision tasks such as image classification, object recognition, and instance segmentation that includes self-attention. Our method substantially improves on the baselines, on instance segmentation, and object recognition while simultaneously lowering the parameters, with little latency overhead, by simply substituting spatial convolutions with global self-attention in the last three bottleneck blocks of a ResNet. We also show how ResNet bottleneck blocked with self-attention may be regarded as Transformer blocks via the architecture of BoTrNe. BoTrNe obtains 46.2 % Mask AP and 51.8 % Box AP utilizing the Mask R-CNN framework on the COCO Instance Segmentation benchmark, exceeding the previous best reported single model and weighted linear results of ResNet tested on the COCO validation set.

1. Introduction

Image classification[1], object identification[2]–[4][5], and instance segmentation[6] have all benefited from deep convolutional backbone architectures[7]–[9]. The majority of well-known backbone designs use several layers of 3 x 3 convolutions.

While the convolution process is capable of capturing local information, vision applications such as object recognition, instance segmentation, and key point detection require the modeling of long-range relationships. E.g. Being able to gather and associate scene details from a large neighborhood, can be effective in understanding relationships across objects[10] in instance segmentation. Convolution-based architectures require the stacking of multiple layers[9], [11] in order to aggregate the regionally captured filter responses globally. Although adding more layers to these backbones[12] improves performance, an implied mechanism to simulate global dependencies may be a more reliable and flexible solution that does not require as many layers.

Natural language processing (NLP) activities need the modeling of long-range relationships as well. Self-attention[13] is a mathematical primitive that learns a complex hierarchy of associative characteristics over lengthy sequences by combining paired entity connections with a feature addressing method. In NLP, this is now becoming a common technique in the format of Transformer blocks[13], with GPT[14] and BERT[15], [16] models serving as notable examples.

Substitute spatial convolutional level with the multi-head self-attention (MHSA) level described in the Transformer[13] for a basic method to utilizing self-attention in vision (Figure 1). In the recent

past, this method has seen success on two apparently disparate approaches. On the one hand, models like SASA [17], AACN [18], SANet [19], Axial-SASA [20], and others suggest that spatial convolutions in ResNet bottleneck blocks [11] be replaced with various kinds of self-attention (global, local, axial, vector, etc). On the other hand, the Vision Transformer (VT) [21] suggests stacking Transformer blocks [13] that operate on non-overlapping patch linear projections. These methods seem to offer two distinct architectural classes. This isn't the case, as we show out. ResNet bottleneck modules with the MHSA layer, on the other hand, may be thought of as Transformer modules with a bottleneck architecture, with minor variations such as residual connections, normalization layer selection, and so on (Figure 3). ResNet bottleneck modules with the MHSA layer are referred to as Bottleneck Transformer (BoTrNe) modules as a result of this equivalency.

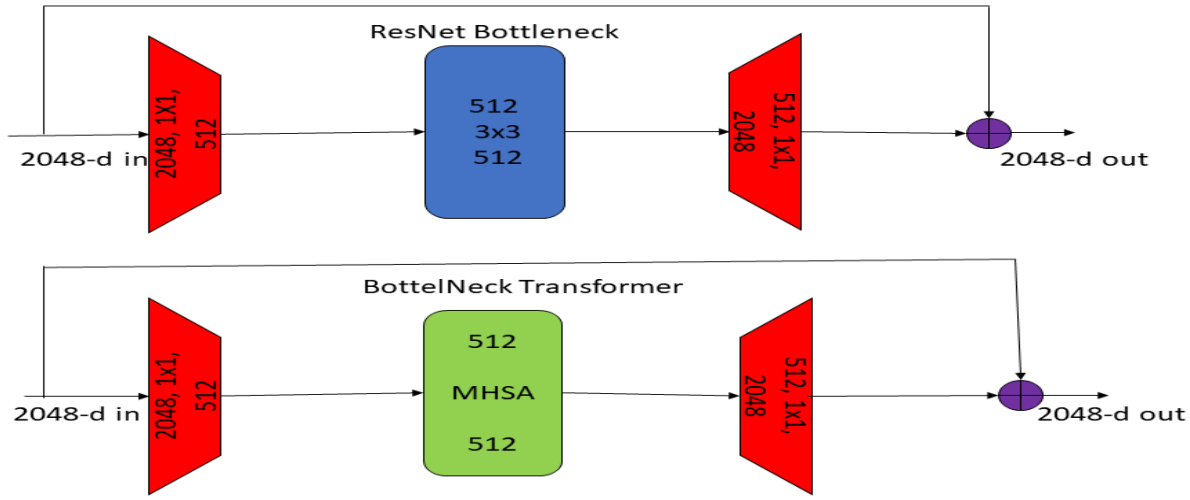

Figure 1: ResNet and Bottleneck Transformer (BoTr) module

When it comes to utilizing self-attention in vision, there are a few issues to consider: (1) In object identification and instance segmentation, picture sizes are considerably bigger (1024 x 1024) than in image classification (224 x 224). (2) Self-attention memory and processing scale exponentially with spatial dimensions [22], resulting in training and inference overheads.

To address these issues, we propose the following design: (1) Use convolutions to effectively learn abstract and low-resolution feature maps from big pictures; (2) process and assemble the data contained in the feature maps acquired by convolutions using global (all2all) self-attention. A hybrid design like this [18] (1) can deal with big pictures efficiently by allowing convolutions perform the spatial down sampling and allowing attention work on lower resolutions; (2) can work with large images successfully by making convolutions do the geometric down sampling and allowing attention work on reduced resolutions. Here's a quick example of how to use this hybrid design in practice: Without making any additional modifications, just the last 3 bottleneck modules of a ResNet should be replaced with BoTr blocks. To put it another way, take a ResNet and only use MHSA layers for the last three 3 3 convolutions (Fig 1, Table 1). With no hyperparameter changes and low overheads for training and inference, this simple modification improves the mask AP by 1.2 percent on the COCO instance segmentation benchmark [23] over our conventional baseline that utilizes ResNet-50 in the Mask R-CNN framework [24]. Given its links to the Transformer via the BoTr blocks, we'll refer to this basic instantiation as BoTrNe from here

on. Despite the fact that its design is not new, we think its simplicity and efficiency make it a valuable reference backbone architecture worth examining.

Without any bells and whistles like Cascade R-CNN [25], FPN modifications [26]–[28], hyperparameter modifications [28], etc., we show substantially better results on instance segmentation using BoTrNe. (1) Implementation gains across different training configurations (Section 4.1), data augmentations (Section 4.2), and ResNet family backbones (Section 4.4); (2) Considerable boost from BoTrNe on small objects (+2.4 Mask AP and +2.6 Box AP); (3) Implementation gains over non-Local layers (Section 4.6); (4) Gains that scale well with larger images (Section 4.6); (5) Gains that scale well with larger images resulting.

Finally, after observing that BoTrNe do not produce significant improvements in a smaller size training regime, we scale BoTrNe, drawing inspiration from the teaching and scaling methods in [12], [17], [28]–[31]. On TPU-v3 hardware, we develop a family of BoTrNe models that reach top-1 accuracy of up to 84.7 percent on the ImageNet validation set while being up to 1.64x quicker than the renowned EfficientNet methods in terms of computation time. We believe that by demonstrating good results with BoTrNe, self-attention will become a frequently utilized primitive in forthcoming vision systems.

1. Background study

Figure 2 shows a categorization of deep learning systems that use self-attention for vision. We'll look at (1) Transformer vs. BoTrNe; (2) DETR vs. BoTrNe; and (3) Non-Local vs. BoTrNe in this part.

Transformer connection: One important lesson in this work is that ResNet restriction modules with Multi-Head Self-Attention (MHSA) levels may be regarded as Transformer modules with a restriction structure, as the article's title implies. Figure 3 depicts this, and we refer to this module as the Bottleneck Transformer (BoTr). We would like to point out that the architecture of the BoTr module is not our work. Rather, we draw attention to the connection amongst MHSA ResNet bottleneck module and the Transformer in the hopes of bettering our knowledge of architectural design spaces [30] for computer vision self-attention. Apart from the ones previously apparent in the diagram (residual connections and module borders), there are a few other differences: (1) Normalization: Transformers utilize Layer Normalization [32], whereas BoTr modules utilize Batch Normalization [33], which is common in ResNet bottleneck modules [11]. (2) Non-Linearities: Transformers just use non-linearity in the FFN modules, whereas the ResNet structure allows for three non-linearities in the BoTr modules. (3) Output projections: In a Transformer, the MHSA module has an output projection, whereas the MHSA layer (Fig 4) in a BoTr module (Fig 1) does not; (4) We have used the SGD with momentum optimizer, which is commonly used in computer vision [11], [24], [34], whereas Transformers are typically trained with the Adam optimizer [21], [35].

Detection Transformer (DETR) Connection: Instead of utilizing an R-CNN, DETR is a detection framework which utilizes a Transformer to conduct implicit region suggestions and object localization [3], [4], [24]. Both DETR and BoTrNe utilize self-attention to enhance object recognition and occurrence segmentation performance. The distinction is that DETR employs Transformer modules outside of the backbone design in order to eliminate region suggestions and

non-maximal inhibition for the sake of simplicity. The aim of BoTrNe, on the other hand, is to offer a backbone framework that utilizes Transformer-like components for instance segmentation and detection. We are unconcerned with the detection architecture (be it DETR or R-CNN). We conduct our tests using the Mask [24] and Faster R-CNN [4] systems, and we leave the integration of BoTrNe as the DETR framework's backbone to future study. We think there may be an opportunity to solve the lack of gain on tiny items observed in DETR by using BoTrNe, which has shown excellent gains on small objects.

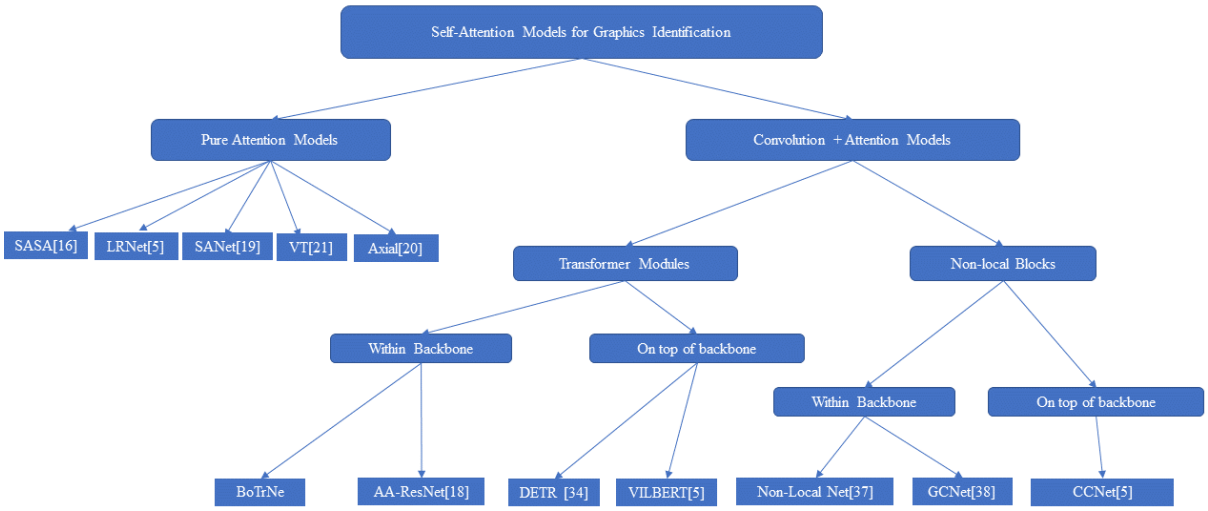

Figure 2: Deep learning architectures for object detection

Non-Local Neural Network Connection: The Transformer and the Non-Local-Means technique [36] are linked by Non-Local (NL) Nets [37]. They enhance video detection and instance segmentation by inserting NL blocks into the last one (or) two module groups (s4,s5) of a ResNet. BoTrNe is a hybrid architecture that uses convolutions and global self-attention, similar to NL-Nets [37], [38]. (1) Multiple heads, value projection, and location encodings in MHSA; (2) NL modules use a bottleneck with a channel factor mitigation of 2 (instead of 4 in BoTr blocks that adopt the ResNet framework); (3) NL module are embedded as additional blocks into a ResNet backbone rather than replacing existing convolution In Section 4.6, we compare BoTrNe, NL-Net, and an NL-like variant of BoTrNe, in which we inject BoTr modules in the same way as NL blocks rather than replacing them.

2. Methodology

Multi-Head Self-Attention (MHSA) levels that perform global (all2all) self-attention across a 2D featuremap replace the last three spatial (3x3) convolutions in a ResNet (Fig 4). A ResNet usually contains four stages [s2, s3, s4, s5] (or module groups) with strides of [4,8,16,32] relative to each other. For the input resolutions (224 x 224 (for classification) and 640 x 640 (for detection experiments in SASA [17]) examined in these studies, approaches that utilize self-attention across the backbone [18], [21] are viable.

Our aim is to utilize attention in more realistic circumstances of high-performance instance segmentation techniques, where larger-resolution pictures (1024 x 1024) are usually used. Given that self-attention across n entities needs $O(n^2d)$ space and computation [13], we think that incorporating self-attention at the lowest pixel density featuremaps in the backbone, i.e. the residual

modules in the s5 stack, is the simplest configuration that conforms to the aforementioned criteria. In a ResNet backbone, the s5 stack usually consists of three blocks, each with one spatial 3 x3 convolution. The BoTrNe architecture is based on substituting them with MHSA layers. In s5, the first block utilizes a 3 x 3 convolution of stride 2, whereas the other two use stride 1. We employ a 2 x 2 mean with a stride 2 for the first BoTr block as all2all attention is not really a strided operation. Table 1 details the BoTrNe design, whereas Figure 4 depicts the MHSA layer.

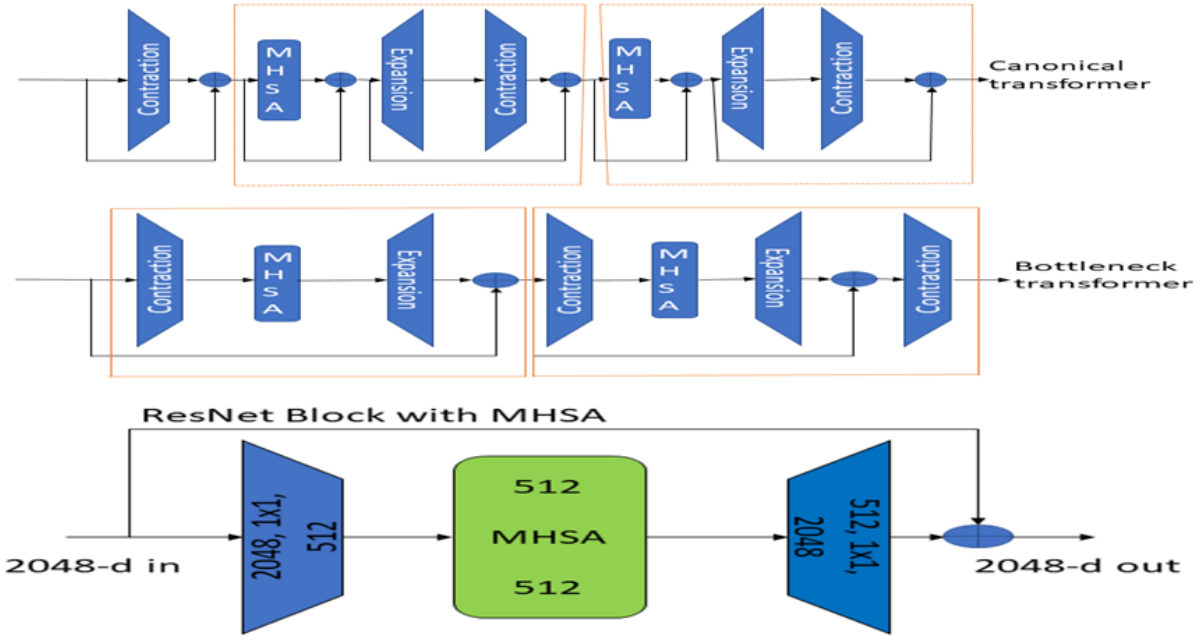

Figure 3: Architecture of canonical and bottleneck transformer

Relative Position Encoding: Transformer-based designs usually employ position encoding to make the attention operation position aware [13]. Relative-width-aware position encodings have recently been shown to be more suitable for vision tasks [17], [18]. This is due to attention taking into consideration not just content information but also relative widths between features at various places, allowing for efficient information association between objects while maintaining positional awareness. The 2D relative location self-attention technique from [17], [18] is used in BoTrNe.

Stage	**S1**	**S2**		**S3**	**S4**	**S5**
Output	512 x 512	256 x 256		128 x 128	64 x 64	32 x 32
ResNet-50	7 x 7, 64, stride 2	3 x 3 max pool, stride 2	1 x1, 64 3 x 3, 64 1 x 1, 256	1 x 1, 128 3 x 3, 128 1x 1, 512	1 x 1, 256 3 x 3, 256 1 x 1, 1024	1 x1, 512 3 x 3, 512 1 x1, 2048
BoTrNe-50	7 x 7, stride 2	3 x 3 max pool, stride 2	1 x1, 64 3 x 3, 64 1 x 1, 256	1 x 1, 128 3 x 3, 128	1 x 1, 256 3 x 3, 256 1 x 1, 1024	1 x1, 512 3 x 3, 512

				1x 1, 512		1 x1, 2048

Table 1: Architecture comparision of BoTrNe-50 with ResNet-50

3. Experiment

The advantage of BoTrNe for object identification and graphics segmentation are investigated. On the COCO dataset [23], we conduct a comprehensive ablation analysis of different design options. The usual COCO metrics are reported, including APbb (averaged across IoU thresholds), APbb50, APbb75, and APmk; APmk50, APmk75 for box and mask, respectively. We train with the COCO train set and publish results with the COCO val (or minival) set, as described in Detectron [34]. The Google Cloud GPU detection codebase was used in our tests. We use the same codebase for all of the baselines and ablations. Our training infrastructure utilizes the Google Colab GPU, which has 4 cores and 8 GB of memory per core unless otherwise stated. We use a batch size of 32 to train with the bfloat16 accuracy and cross-replica batch normalization [24], [33], [34], [39].

Backbone	**Rs50**	**BoTr50**	**Rs50**	**BoTr 50**	**Rs50**	**BoTr50**	**Rs50**	**BoTr50**
Iterations	12	12	24	24	36	36	72	72
AP^{bb}	39	39.4	41.2	42.8	42.1	43.6	42.8	43.7
AP^{mk}	35	35.3	36.9	38	37.7	38.9	37.9	38.7

Table 2: Comparison of Rs50 with BoTr50 under 1x(12 iterations), 3x(36 iterations), 6x(72 iterations).

3.1 With Mask RCNN, BoTrNe outperforms ResNet on COCO Instance Segmentation:

We'll look at the most basic and frequently used configuration: a ResNet-50 backbone with a Feature Pyramid Network. We utilize pictures with a resolution of 1024 x 1024 and a multi-scale delay of [0.8, 1.25] (adjusting the image size between 820 and 1280 to match the Detectron default of 800 x 1300). We compare the ResNet-50 (Rs50) and BoTr ResNet-50 (BoTRs50) as backbone designs for various training regimens in this setting: 1x: 12 iterations, 2x: 24 iterations, 3x: 36 iterations, 6x: 72 iterations, all with the same hyperparameter for both backbones across all training schedules (Table 2). With the exception of the 1x schedule, we can plainly see that BoTRs50 is a considerable improvement over Rs50 (12 epochs). This indicates that in order to demonstrate substantial progress over Rs50, BoTRs50 need more training time. We can also observe that the BoTRs50 improvement in the 6x schedule (72 iterations) is less than in the 3x schedule (32 iterations). This indicates that training with the standard scale jitter for a long time is harmful. This is addressed by using a greater aggressive scale jitter (Section 4.2).

Backbone	**Rs50**	**BoTr50**	**Rs50**	**BoTr50**	**Rs50**	**BoTr50**
Delay	0.8 ,1.25	0.8,1.25	0.5, 2	0.5, 2	0.1, 2.0	0.1, 2.0
AP^{bb}	42.8	43.7	43.7	45.3	43.8	45.9
AP^{mk}	37.9	38.7	39.1	40.5	39.2	40.7

Table 3: Comparing Rs50 and BoTr50 on multi-scale delay

3.2 BoTrNe benefits more from Scale latency than ResNet:

We observed in Section 4.1 that training for a longer period of time (72 iterations) decreased the gains for BoTRs50. Increasing the amount of multi-scale latency, which has been shown to enhance the performance of identification and segmentation systems [40], is one approach to solve this. Table 3 shows that BoTRs50 performs significantly better than Rs50 for multi-scale latency of [0.5, 2.0], while also revealing significant gains (+ 2.2 percent on APbb and + 1.6 percent on APmk) for scale latency of [0.1, 2.0], implying that BoTrNe (self-attention) benefits more from additional augmentations such as multi-scale latency than ResNet (pure convolutions).

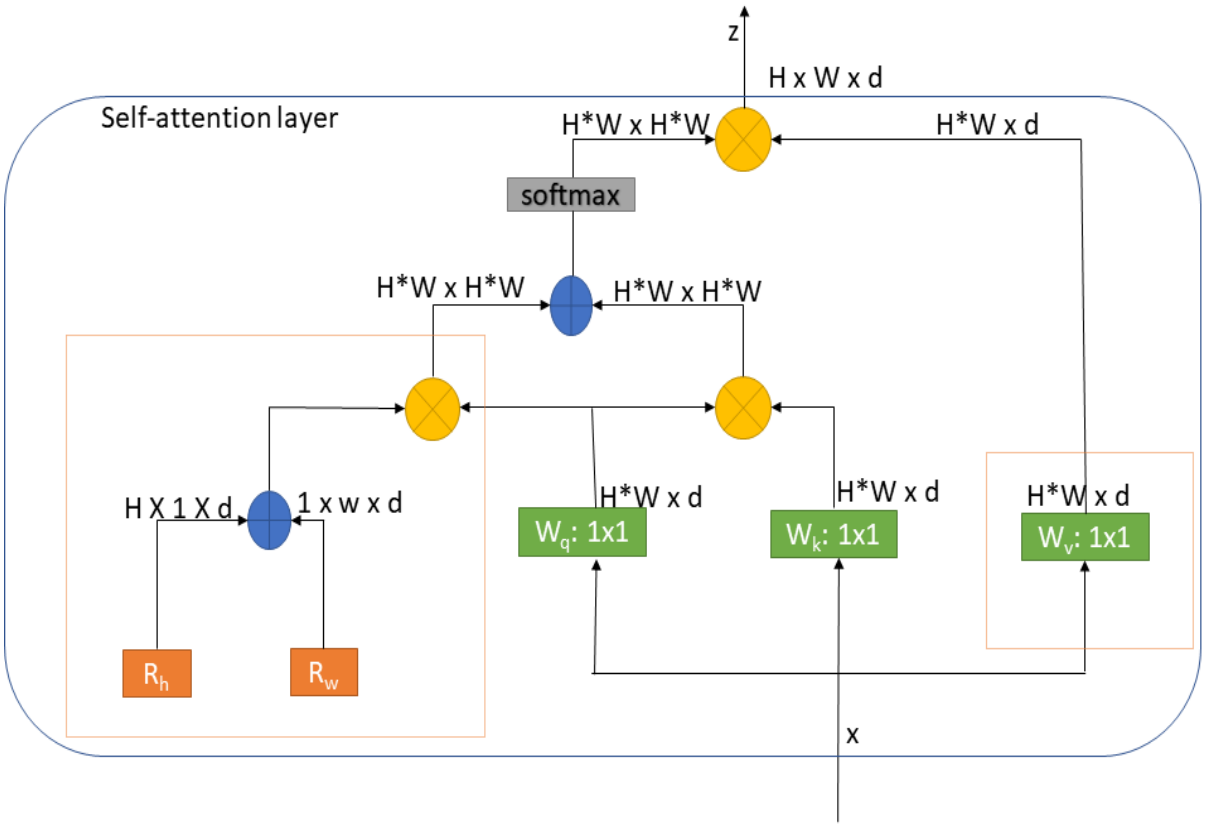

Figure 4: Detail architecture of BoTr module with MHSA layer

3.3 Performance is Boosted by Relative Position Encodings

Relative position encodings [41] are used by BoTrNe . We provide an argument for the use of relative position encodings by comparing individual benefits from content-content interaction (qkT) and content-position interaction (qrT), where q, k, and r denote query, key, and relative position encodings, respectively. The canonical setting is used for the ablations (Table 4). We could see that the gains from qrT and qkT are complementary, with qrT being the more important. For example, qkT alone contributes to 0.6 percent APbb and 0.6 percent APmk improvement over the Rs50 baseline, whereas qrT alone contributes to 1.0 percent APbb and 0.7 percent APmk improvement over the R50 baseline. The improvements on both APbb and APmk are cumulative when combined (qkT + qrT). (1.5 percent and 1.2 percent respectively). We can also observe that absolute position encodings (qrT abs) do not offer as much benefit as relative position encodings. This indicates that incorporating relative position encodings into systems like DETR [10] may be a promising future research path.

3.4 Comparison with Non-Local Neural Networks:

What is the difference between BoTrNe and Non-Local Neural Networks? Between the pre-final and final bottleneck modules of a ResNet backbone, NL operations are introduced into the s4 stack. This increases the number of parameters in the model, while BoTrNe reduces the number of parameters (Table 5). We add ablations to the NL mold, where we present BoTr module in the same way as the NL module. We also do an ablation using two BoTr module, one in each of the s4,s5 stacks. Table 7 summarizes the findings. Adding an NL increases APbb by 1.0 and APbb by 0.7, while inserting a BoTr module increases APbb by +1.6 and APmk by +1.2, demonstrating that the BoTr module design is superior than the NL. Furthermore, BoT-R50 (which replaces rather than adds new blocks) yields +1.5 APbb and + 1.2 APmk, which is comparable to adding another BoTr module and superior to adding one more NL module.

3.5 Examining BoTrNe model on ImageNet:

While the design of BoTrNe was driven by the need for segmentation and identification, it is reasonable to wonder whether the architectural design of BoTrNe aids in improving picture recognition accuracy on the ImageNet [42] benchmark. Incorporating Non-Local modules to ResNets and training them with canonical parameters does not offer significant improvements, according to previous research [5], [43]. When comparing BoTrRs-50 to ResNet-50, we get a comparable result when both models are trained using the canonical ImageNet hyperparameters [30]: 100 iterations, batch size 1024, weight decay 1e-4, conventional ResNet data augmentation, and cosine learning rate schedule (Table 9). On ImageNet, BoT50 does not offer substantial improvements over Rs50, although it does have the advantage of reducing parameters while maintaining similar computation (M.Adds).

Taking use of the picture sizes often employed for image categorization is a straightforward way to address this lack of benefit. We typically deal with considerably lower picture sizes (224 x 224) in image classification than we do in object recognition and segmentation (1024 x 1024). As a result, the featuremaps used by the BoTr modules are considerably smaller (e.g. 14 x 14, 7 x 7) than those used in instance segmentation and detection (e.g. 64 x 64, 32 x 32). The BoTrNe architecture in the c5 block group may be modified to universally utilize a stride of 1 in the last MHSA layers with the same number of parameters and without a substantial increase in computation. This design is known as BoTrNe-F1 (F1 denotes the first stride in the last module group). This architecture is comparable in concept to the Vision Transformer (VT) [21] hybrid models, which utilize a ResNet up to stage c4 before stacking Transformer blocks. The major difference between hybrid VT models and the BoTrNe-F1 models is that BoTr modules are used instead of standard Transformer modules (other changes include the normalization layer, optimizer, and so on, as stated in the Background study comparison to Transformer) (Sec. 2).

Backbone	Att. Type	Ap^{bb}	Ap^{mk}
Rs50		42.1	37.7
BoTr50	qk^T	42.7	38.3
BoTr50	$qr^T_{relative}$	43.1	38.4

BoTr50	$qk^T + qr^T_{relative}$	43.6	38.9
BoTr50	$qk^T + qr^T_{abs}$	42.5	38.1

Table 4: Relative position encoding analysis

Backbone	AP^{bb}	AP^{mk}
Rs50	42.1	37.7
BoTr50	43.6	38.9
Rs101	43.3	38.4
BoTr101	45.5	40.4
Rs152	44.2	39.1
BoTr152	46.0	40.6

Table 5: Comparing different ResNet architectures with BoTr architectures

3.6 Analysis under standard environment:

This design is initially evaluated for the 100-iteration setting, as well as Rs50 and BoT50. In the normal configuration, we find that BoT-F1-50 improves by 0.9 percent over Rs50 (Table 9). However, this gain comes at the expense of additional computation (m.adds). Nonetheless, the boost is a hopeful indication that we can build models that scale well with bigger pictures and better training circumstances, which have been increasingly popular since EfficientNets [7].

Backbone	Res	AP^{bb}	AP^{mk}
Rs50	1280	44.0	39.5
BoTr50	1024	45.9	40.7
BoTr50	1280	46.1	41.2
Rs101	1280	46.4	41.2
BoTr101	1024	47.4	42.0
BoTr101	1280	47.9	42.4

Table 6: Comparing different ResNet architectures with BoTr architectures while training for 72 iterations

Backbone	Change in background	AP^{bb}	AP^{mk}
Rs50		42.1	37.7
Rs50 + NL	+1 NL module in s4	43.1	38.4
Rs50 + BoTr [s4]	+1 BoTr module in s4	43.7	38.9
Rs50 + BoTr [s4,s5]	+2 BoTr module in s4,s5	44.9	39.7
BoTr50	Substitution in s5	43.6	38.9

Table 7: Table 6: Comparing different ResNet architectures with BoTr architectures while training for 36 epochs

Backbone	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
BoTr152	49.5	71.0	54..2	43.7	68.2	47.4
BoTr200	49.7	71.3	54.6	44.4	68.9	48.2

Table 8: BoTr with 152 and 200 layers when trained for 72 epochs

Backbone	M.Adds	Parameters	Top 1% accuracy
Rs50	3.86 G	25.5M	76.8
BoTr50	3.79G	20.8M	77
BoTr-F1-50	4.27G	20.8M	77.7

Table 9: Analysis of Rs50 and BoTr50 and BoTr-F1-50 on ImageNet on 100 epochs

Backbone	Top 1% accuracy	Top 5% accuracy
Rs50	77.7	93.9
BoTr50	78.3	94.2
BoTr-F1-50	79.1	94.4

Table 10: Analysis of Rs50 and BoTr50 and BoTr-F1-50 on ImageNet on 200 epochs

3.7 Effect of longer training and larger data

Our instance segmentation studies revealed that regularization, such as increasing dataset (in the case of segmentation, greater multi-scale delay) and prolonged training, benefits BoTrNe and self-attention more. When training with a better setup, such as 200 iterations, batch size 4096, weight decay 8e-5, RandAugment (2 levels, magnitude 10), and label smoothing of 0.1, it's reasonable to anticipate that the benefits from BoTr and BoTr-F1 would increase. When compared to the baseline Rs50, the benefits for both BoTr50 (+ 0.6 percent) and BoTr-F1-50 (+ 1.4 percent) are considerably more significant in this scenario (Table 10).

4. Conclusion:

The use of self-attention in the design of visual backbone systems is an interesting subject. We believe that our work contributes to a better knowledge of architecture in this area. Self-attention frameworks for self-supervised learning in computer vision [44]; and expanding to much bigger datasets such as JFT, YFCC, and Instagram are all promising areas for future research. A key future goal is to compare and include alternatives to self-attention, such as lambda-layers [45].

Reference:

[1] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015, doi: 10.1007/s11263-015-0816-y.

[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (VOC) challenge,” *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010, doi: 10.1007/s11263-009-0275-4.

[3] R. Girshick, “Fast R-CNN,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter, pp. 1440–1448, 2015, doi: 10.1109/ICCV.2015.169.

[4] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.

[5] A. Srinivas, T.-Y. Lin, N. Parmar, J. Shlens, P. Abbeel, and A. Vaswani, “Bottleneck transformers for visual recognition,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16519–16529.

[6] J. Dai, K. He, and J. Sun, “Instance-Aware Semantic Segmentation via Multi-task Network Cascades,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-

Decem, pp. 3150–3158, 2016, doi: 10.1109/CVPR.2016.343.

[7] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," *36th Int. Conf. Mach. Learn. ICML 2019*, vol. 2019-June, pp. 10691–10700, 2019.

[8] S. Xie, R. Girshick, and P. Doll, "Aggregated Residual Transformations for Deep Neural Networks http://arxiv.org/abs/1611.05431v2," *Cvpr*, pp. 1492–1500, 2017.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.

[10] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12346 LNCS, pp. 213–229, 2020, doi: 10.1007/978-3-030-58452-8_13.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[12] H. Zhang *et al.*, "ResNeSt: Split-Attention Networks," 2020, [Online]. Available: http://arxiv.org/abs/2004.08955.

[13] A. Vaswani *et al.*, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[14] T. B. Brown *et al.*, "Language models are few-shot learners," *Adv. Neural Inf. Process. Syst.*, vol. 2020-Decem, 2020.

[15] Y. Liu *et al.*, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," no. 1, 2019, [Online]. Available: http://arxiv.org/abs/1907.11692.

[16] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, vol. 1, no. Mlm, pp. 4171–4186, 2019.

[17] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, "Studying Stand-Alone Self-Attention in Vision Models," 2019.

[18] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V Le, "Attention augmented convolutional networks," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3286–3295.

[19] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10076–10085.

[20] H. Wang, Y. Zhu, B. Green, H. Adam, A. Yuille, and L.-C. Chen, "Axial-deeplab: Stand-alone axial-attention for panoptic segmentation," in *European Conference on Computer Vision*, 2020, pp. 108–126.

[21] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv Prepr. arXiv2010.11929*, 2020.

[22] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *arXiv Prepr. arXiv2009.06732*, 2020.

[23] T. Y. Lin *et al.*, "Microsoft COCO: Common objects in context," *Lect. Notes Comput. Sci.*

(including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 8693 LNCS, no. PART 5, pp. 740–755, 2014, doi: 10.1007/978-3-319-10602-1_48.

[24] P. Doll, R. Girshick, and F. Ai, "Mask R-CNN ar."

[25] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6154–6162.

[26] G. Ghiasi, T.-Y. Lin, and Q. V Le, "Nas-fpn: Learning scalable feature pyramid architecture for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7036–7045.

[27] Y. Liu *et al.*, "Cbnet: A novel composite backbone network architecture for object detection," in *Proceedings of the AAAI conference on artificial intelligence*, 2020, vol. 34, no. 07, pp. 11653–11660.

[28] M. Tan, R. Pang, and Q. V Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10781–10790.

[29] I. Bello *et al.*, "Revisiting resnets: Improved training and scaling strategies," *arXiv Prepr. arXiv2103.07579*, 2021.

[30] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10428–10436.

[31] T. Ridnik, H. Lawen, A. Noy, E. Ben Baruch, G. Sharir, and I. Friedman, "Tresnet: High performance gpu-dedicated architecture," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1400–1409.

[32] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv Prepr. arXiv1607.06450*, 2016.

[33] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, 2015, pp. 448–456.

[34] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron." 2018.

[35] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision*, 2020, pp. 213–229.

[36] A. Buades, B. 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)*, 2005, vol. 2, pp. 60–65.

[37] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7794–7803.

[38] Y. Cao, J. Xu, S. Lin, F. Wei, and H. Hu, "Gcnet: Non-local networks meet squeeze-excitation networks and beyond," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, p. 0.

[39] Y. Wu, "Group Normalization – Facebook Research," *Eccv*, no. Figure 1, 2018, [Online]. Available: https://research.fb.com/publications/group-normalization/.

[40] G. Ghiasi *et al.*, "Simple copy-paste is a strong data augmentation method for instance segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and*

Pattern Recognition, 2021, pp. 2918–2928.

[41] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," *arXiv Prepr. arXiv1803.02155*, 2018.

[42] K. Alex, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional networks."

[43] C. Xie, Y. Wu, L. van der Maaten, A. L. Yuille, and K. He, "Feature denoising for improving adversarial robustness," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 501–509.

[44] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*, 2020, pp. 1597–1607.

[45] I. Bello, "Lambdanetworks: Modeling long-range interactions without attention," *arXiv Prepr. arXiv2102.08602*, 2021.