Neural Processing Letters (2023) 55:6797–6813
https://doi.org/10.1007/s11063-023-11161-z

Performance-Efficiency Comparisons of Channel Attention Modules for ResNets

Sander R. Klomp[1,2] · **Rob G. J. Wijnhoven**[2] · **Peter H. N. de With**[1]

Accepted: 26 January 2023 / Published online: 11 February 2023

Abstract
Attention modules can be added to neural network architectures to improve performance. This work presents an extensive comparison between several efficient attention modules for image classification and object detection, in addition to proposing a novel Attention Bias module with lower computational overhead. All measured attention modules have been efficiently re-implemented, which allows an objective comparison and evaluation of the relationship between accuracy and inference time. Our measurements show that single-image inference time increases far more (5–50%) than the increase in FLOPs suggests (0.2–3%) for a limited gain in accuracy, making computation cost an important selection criterion. Despite this increase in inference time, adding an attention module can outperform a deeper baseline ResNet in both speed and accuracy. Finally, we investigate the potential of adding attention modules to pretrained networks and show that fine-tuning is possible and superior to training from scratch. The choice of the best attention module strongly depends on the specific ResNet architecture, input resolution, batch size and inference framework.

Keywords Attention modules · Resnet · CNN efficiency

1 Introduction

When developing the most powerful neural network architectures in computer vision, small changes to network architectures have historically resulted in significant gains in accuracy, such as the introduction of skip connections in ResNet [1], batch normalization [2], or the addition of self-attention mechanisms, like Squeeze-and-Excitation (SE) modules [3]. Due

✉ Sander R. Klomp
s.r.klomp@tue.nl

Rob G. J. Wijnhoven
rob.wijnhoven@vinotion.nl

Peter H. N. de With
p.h.n.de.with@tue.nl

[1] Electrical Engineering VCA Group, Eindhoven University of Technology, 5612AZ Eindhoven, The Netherlands

[2] ViNotion, Daalakkersweg 2-58, 5641JA Eindhoven, The Netherlands

to their simplicity, these changes come with relatively small computational overhead, making them excellent additions to networks used in real-world systems. Attention mechanisms have been gaining traction in recent years in the computer vision community, achieving large performance improvements in classification [3–5], object detection [6–8] and other visual domains, such as tracking [9]. Impressively, reported accuracy gains are significant over many different architectures and tasks [10, 11]. Many of these works focus on designing increasingly complex attention mechanisms to achieve the best possible accuracy, but this comes with additional computational cost. Instead, we investigate how much of the performance improvement of attention mechanisms can be retained, using a module that is as simple and computationally efficient as possible. These research objectives are similar to those of the recently published ECA-Net [11].

In this paper, we focus on attention modules that up- or down-weight entire feature maps based on global feature map information, often called "feature recalibration", similar to [3, 8, 10, 11]. Our objective is to either find or design the attention module with the best possible inference speed-to-accuracy trade-off. To this end, we propose a novel attention module called the Attention-Bias (AB) module, that recalibrates feature maps using only a single Hadamard product with learned weights and a single tensor addition per module, thereby making it the most low-cost attention module as far as the authors know. The architecture of the AB block is shown in Fig. 1 and can be used as a drop-in replacement of any residual block in a ResNet-like architecture, or added after convolutional blocks in other architecture types. Finally, we investigate under which conditions existing attention modules outperform ours, and when to apply attention modules in general.

The contributions of our research are as follows.

- Optimized re-implementations (code available[1]) of state-of-the-art attention modules are compared with the same data and under identical conditions, on both accuracy and computational complexity, based on computation time in PyTorch and TensorRT.
- Extensive experiments are performed to determine under which conditions attention modules provide the best speed-to-performance trade-off.
- A new attention module is designed that has a smaller computational and memory overhead but still performs similarly to more expensive attention modules in multiple situations.
- We investigate the potential of attention modules to improve the performance of pretrained networks, after only brief fine-tuning.

2 Related Work

2.1 Convolutional Neural Networks

Convolutional neural networks have been achieving state-of-the-art performance on computer vision tasks, ever since AlexNet [12] first achieved top performance on the ImageNet [13] classification dataset. The VGG network [14] showed that additional network depth can improve performance further, followed later by ResNet [1] which introduced residual connections to make even deeper networks trainable. Due to their simplicity and despite their age, ResNets are used to this day as a baseline network for comparisons, as a backbone network for tasks such as detection or segmentation, or as a starting point for modifications such as in ResNeXt [15].

[1] https://github.com/SanderKlomp/channel-attention.

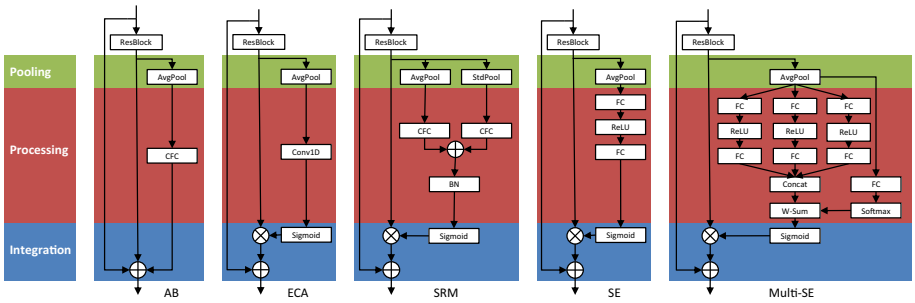


Fig. 1 Architecture of our Attention Bias (AB) module, compared with more expensive state-of-the-art attention modules. These include from left to right, Efficient Channel Attention (ECA) [11], Style-based Recalibration Module (SRM) [10], Squeeze-and-Excitation (SE) [3] and Multi-Squeeze-and-Excitation (Multi-SE) [8]. Abbreviations used: CFC (Hadamard product), FC (fully connected), one-dimensional convolution (Conv1D), Batch Norm (BN), weighted sum (W-Sum)

2.2 Global Image Information as Means and Variances

Networks trained on ImageNet are known to be highly biased towards style [16], reducing their performance on unseen data, especially in case of style differences. Using the assumption that style is primarily represented in the mean and variance of feature maps [17], it makes sense to optimize the classification accuracy by making the network invariant to changes in these values. Both batch normalization [2] and instance normalization [18] do this in some way. IBN-net [19] intelligently combines these two, by observing that the impact of style can effectively be reduced by applying instance normalization at only early layers of the network, removing the mean and variance. Instead of attempting to compensate for global information using a fixed normalization, Lee *et al.* design an attention module, called the Style-based Recalibration Module (SRM) that allows the network to learn style invariance, by learning multiplication weights based on feature means and variances [10].

2.3 Attention Mechanisms

Multiple types of attention exist, with channel attention and spatial attention being most common. Channel attention can be broadly defined as a mechanism by which a network can weight feature maps depending on context, to achieve better performance [3, 5, 10, 20]. In contrast, spatial attention focuses the network on specific spatial regions in the input or feature maps [21–23]. In this work we limit ourselves to channel attention, because their implementations are generally less computationally expensive. The simplest form of channel attention consists of a network block that performs global pooling of some features followed by processing layers and using the resulting values to weight the original features. Many attention module variants follow this scheme, such as SE [3], SRM [10], ResNeSt [5], GE [20], CBAM [24], AA [25], and ECA [11]. Alternatively, an entire separate network can be used to compute the feature attention weights [26]. In some recent attention block designs, such as GC-net [27] and NA-net [28] the ordering of pooling and processing is inverted, which appears to result in slightly better performance, but at a higher computation cost. ECA-Net [11] has been developed in parallel to our work and also aims at designing a module

with efficient speed-to-performance trade-off. Their "SE-Var2" module is especially similar in design to our proposal.

3 Attention Bias Module

Creating a computationally inexpensive attention module means removing the parts that are the most computationally expensive relative to their performance improvement. The components in the attention blocks of [3, 5, 10, 11, 20] vary, and while these studies contain ablation experiments, these ablation studies only look at classification performance and not inference speed. Therefore, our objective is to test various attention blocks and measure their relative improvement compared to their additional computational overhead. The three main components of channel attention modules, "pooling", "processing" and "integration", are illustrated in Fig. 1 for our own proposed module and four state-of-the-art modules. Each step is detailed in the sections below. As a starting point of our research, we commence with the design of the squeeze-and-excitation (SE) block [3].

3.1 Global Pooling

Global information can be extracted from a feature map in various ways (shown as the green color in Fig. 1), such as average pooling ("squeeze" in [3]), max pooling, or standard deviation pooling [10]. More complicated methods may be used, but these are more computationally expensive. Although max pooling is potentially the least computationally expensive, it was shown to perform poorly as a global pooling operation by Lee et al. [10], hence we restrict ourselves to modules that use the other two pooling options. The AB module employs only spatial average pooling, which is computed by

$$Z = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} X(i, j), \tag{1}$$

where $X \in \mathbb{R}^{H \times W \times C}$ is the input feature map tensor, that is reduced to a tensor $Z \in \mathbb{R}^{C}$.

3.2 Processing

The weight computation of the feature map of the AB block consists of a single Hadamard product of the tensor Z with learnable weights $W \in \mathbb{R}^{C}$, one for each channel, specified by

$$Z^{'} = Z \circ W. \tag{2}$$

This is a notable simplification compared to the two ReLU-separated fully connected (FC) layers in the SE block. While these FC layers account for only a few FLOPs due to the prior pooling operation and reduction factor, inefficiencies of GPUs for computing small inner products give rise to a larger than expected performance overhead for the SE block. Furthermore, FC layers reach relatively large numbers of parameters for deeper layers with many channels. Lee et al. [10] perform the same simplification for SRM, by first realizing that a single FC layer performs similarly to two layers, and then simplifying it further by removing cross-channel connections. This simplification also results in a single Hadamard product. Removing these cross-channel connections means that attention is now only based on a single channel of the feature tensor at a time. In contrast, Multi-SE [8] expands the

cross-channel power by learning multiple sets of FC weights. This allows the network to train on multiple, very different datasets simultaneously (e.g. ImageNet and sets of medical images), but this is computationally expensive and may not add value for a dataset with less image variety. Hence, we stay with our proposed approach of using a single Hadamard product for tensor weighting.

3.3 Weight Integration

All commonly used attention blocks combine their computed feature-importance weights with the original features through multiplication [3, 5, 10, 20], after applying a sigmoid to the weights to ensure a unity interval for each weight value. This essentially means that features can be only reweighted *relative* to other features, but never in the absolute sense. Without the sigmoid, training can become unstable. However, applying a sigmoid introduces some computational overhead. Instead, we remove the sigmoid computation and integrate the computed weights with the original features through a summation, which also remains stable during training. This essentially allows the network to learn an input-dependent bias for each feature map instead of a weight. An advantage of summation is that, when the weights of the attention module are initialized to zero, it can be added to any pretrained network without adverse effects on the performance. The weights may then be learned during a very brief fine-tuning phase.

4 Experiments

In this section we perform an extensive evaluation of the chosen attention modules, both in accuracy and computational cost. We start by investigating classification networks and extend our experiments to object detection. Computational cost is evaluated both in the number of floating-point operations (FLOPs) and inference time in two different implementation frameworks. We re-implement all attention modules with more efficient PyTorch operations, the same across all modules for fair comparison. All reported results are from our own training iterations and all timing results are evaluated on a single RTX 2080 Ti, with unity batch size. We limit ourselves to widely used ResNet architectures [1], since attention modules have been shown to improve performance for many different other architectures already [3, 5, 10, 20].

By default, ResNet blocks are implemented differently depending on the network depth. When defining the ResNet, one can use either "basic blocks" or "bottleneck blocks" (see Fig. 5 in [1]). Bottleneck blocks contain four times as many feature channels in between blocks, compared to basic blocks, and compress and decompress around 3×3 convolutions using 1×1 convolutions for computational efficiency. Basic blocks use two sets of 3×3 convolutions without any channel compression. Basic blocks are default for ResNet-18 and ResNet-34 and bottlenecks are default for ResNet-50 and deeper. Note that the number of channels, which differs between basic blocks and bottleneck blocks, dictates the amount of information available inside attention modules after global pooling. In all experiments, we use default settings for the attention modules, i.e. reduction ratio between FC layers $r = 16$ for SE, 1D-convolution kernel size $k = 3$ for ECA and 3 branches for Multi-SE. Attention modules are applied at each residual block and "baseline" refers to the network without attention modules. The total number of residual blocks, and thus attention modules, depends on the ResNet depth.

Table 1 Average CIFAR-10 and CIFAR-100 accuracy (%) over 3 runs, using a ResNet-56 and various attention modules

Dataset	Model	Basic		Bottleneck	
		Top-1	Top-5	Top-1	Top-5
CIFAR-10	Baseline	93.77	99.88	93.05	99.87
	AB	93.71	**99.89**	93.81	99.88
	ECA	94.02	99.87	94.03	99.88
	SE	94.01	99.88	94.25	99.91
	SRM	93.79	99.83	**94.65**	**99.92**
	Multi-SE	**94.03**	99.88	94.23	**99.92**
CIFAR-100	Baseline	71.16	92.41	72.67	93.40
	AB	72.17	**93.03**	74.84	94.14
	ECA	71.57	92.71	74.43	94.02
	SE	72.18	**93.00**	**75.43**	**94.56**
	SRM	71.90	92.44	74.59	93.94
	Multi-SE	**72.20**	92.98	74.76	94.27

Best scores are made bold and those within one standard deviation of the best are underlined. There is no clear best attention module for all cases

4.1 Classification Accuracy

We perform a general comparison of the classification accuracy of state-of-the-art attention modules. Three different datasets are evaluated: CIFAR-10, CIFAR-100 and ImageNet. All networks are trained from scratch.

4.1.1 CIFAR

We use a typical architecture and training settings for CIFAR. We train a basic-block variant of ResNet-56 for 164 epochs using batches of 128 images, with learning rate 0.2 and reduce the learning rate by a factor 10 at the 81st and 122nd epochs. This architecture contains 27 basic ResNet blocks, thus also 27 attention modules. The accuracy scores for CIFAR-10 and CIFAR-100 are shown in Table 1. Note that there is no clear "best" attention module, despite reported results of cited work.

There is a difference in the impact of attention modules, depending on whether the ResNet is created with bottleneck or basic blocks, which is larger than the differences between the modules. On CIFAR-100, when using basic blocks, attention modules add only upwards of $\approx$1 percent point to the top-1 accuracy, while for a ResNet architecture using bottleneck blocks, it adds upwards of $\approx$3 percent points. The fact that bottleneck blocks result in far better performance when combined with attention modules is consistent over both CIFAR-10 and CIFAR-100 and occurs for all tested attention modules, although the gains are roughly twice as large for CIFAR-100 than for CIFAR-10. Having access to four times as many feature channels is thus highly beneficial for these modules.

4.1.2 ImageNet

We limit our experimental validation to ResNet-18, with basic blocks, and ResNet-50, with bottleneck blocks, and train them with multiple different attention modules. Again, each

Table 2 ImageNet classification performance for ResNet-18 and ResNet-50 with different attention modules (from scratch). Inference time is measured for a single image on an RTX 2080 Ti. Best result in bold, close second-best results are underlined. The AB module has the smallest computational overhead, especially in PyTorch, but also provides only a modest performance improvement

Module	ResNet-18					
	Accuracy		FLOPs (G)	Inference time (ms)		Params (M)
	Top-1	Top-5		PyTorch	TensorRt	
Baseline	70.05	89.19	1.82	1.90	0.87	11.69
AB	70.26	89.47	1.82	2.35	0.93	11.69
ECA	70.52	89.68	1.82	2.86	0.94	11.69
SE	70.58	**89.97**	1.82	3.26	0.99	11.70
SRM	70.39	89.60	1.82	3.50	1.09	11.78
Multi-SE	**70.78**	89.87	1.83	5.89	1.21	11.96
	ResNet-50					
Baseline	75.69	92.79	4.11	4.48	1.98	25.56
AB	76.15	92.98	4.12	5.40	2.19	25.57
ECA	77.01	**93.44**	4.12	6.52	2.21	25.56
SE	**76.98**	**93.44**	4.16	7.35	2.34	28.09
SRM	76.95	93.38	4.14	7.92	2.72	25.62
Multi-SE	76.83	93.34	4.26	12.01	2.88	33.20

Best scores are made bold and those within one standard deviation of the best are underlined

block receives a single attention module, which adds up to 8 attention modules for ResNet-18 and 16 for ResNet-50. The parameters and training code are available[2], based on the implementation of [10]. The results are shown in Table 2. The performance improvement of adding attention modules is significantly larger for ResNet-50 than for ResNet-18. This observation reinforces the idea that attention modules require sufficient feature channels to function properly. Due to its simplicity, our AB module does not outperform the other attention modules in accuracy, but has a significantly lower computational overhead when measured in execution time.

4.2 Fine-Tuning Pretrained Models

In practice, it is common to use a pretrained network and fine-tune it to a specific task, since training from scratch is costly in terms of both time and data. In this experiment, we investigate how close the new minimum of the network is to that of a pretrained network when adding initialized attention modules. First, we fine-tune an ImageNet-pretrained ResNet-18 network with added initialized attention modules using learning rate 0.001 (the final learning rate of the pretrained network's learning schedule) for up to 30 epochs. Second, under the assumption that the optimization process first needs to escape the current local minimum, we test whether train-from-scratch performance can be reached by fine-tuning for 40 epochs, first 20 at learning rate 0.01, then 20 at learning rate 0.001 to settle in a new local minimum. The results are shown in Table 3.

[2] https://github.com/SanderKlomp/channel-attention.

Table 3 Top-1 accuracy after fine-tuning an ImageNet pretrained ResNet-18, with initialized attention module. Baseline refers to the pretrained model without attention modules, after which attention modules are added. The AB module is the only one that does not deteriorate performance when added to an initialized model and maintains a lead over other modules during finetuning, unless training with large learning rates for many epochs

Model	Initialized	2 epochs lr=0.001	30 epochs lr=0.001	20+20 epochs lr=0.001→ 0.01
Baseline	**69.96**	69.96	70.03	69.99
AB	**69.96**	**70.11**	**70.17**	70.47
ECA	3.37	33.36	70.08	70.46
SE	6.46	69.84	70.15	70.62
SRM	6.55	68.46	70.08	**71.03**
Multi-SE	6.47	69.87	70.13	70.71

Best scores are made bold

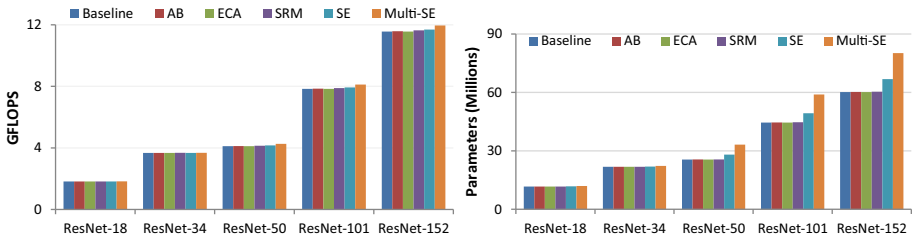
Regardless of the applied attention module, for low learning rate, the model remains stuck in the local minimum of the pretrained model, even after 30 epochs. However, for the 40 epoch test, the networks significantly outperform the baseline pretrained network. Furthermore, the resulting scores are consistently similar or higher than the trained-from-scratch equivalents (Table 2, ResNet-18 Top-1). Concluding, when designing novel attention modules, it is beneficial to fine-tune from a pretrained network, resulting in lower training time (convergence in 40 vs. 90 epochs) and improved accuracy. We investigate whether this property also holds for the object detection task in Sect. 4.4.

Note that in Table 3, the initialized score of the AB module is much higher than that of other modules as explained in Sect. 3.3. Similar functionality can be added to the other attention modules by adding a learnable scaling layer with weight zero and bias unity right after the sigmoid, but this increases their computational complexity further and can impact training stability negatively.

4.3 Computational Complexity

To analyze the computational overhead of adding attention modules, we now look at all commonly used ResNet depths for Imagenet classification: ResNet-18, 34, 50, 101 and 152. Once again, a single attention module is added for each block, which results in the following number of attention modules for each depth: 8 for ResNet-18, 16 for ResNet-34, again 16 for ResNet-50 but now with the deeper bottleneck blocks, 33 for ResNet-101 and 50 for ResNet-152.

Attention modules appear extremely efficient when analyzing the number of FLOPs. As can be seen in Fig. 2a, the impact on the total number of FLOPs is negligible compared to using a deeper network architecture, regardless of the used attention module. Similarly, the impact of attention modules on the total number of parameters in a network is also small compared to using deeper networks, as shown in Fig. 2b. When comparing the number of parameters, only SE and Multi-SE modules result in noticeable overhead, especially for deep networks using bottleneck blocks, but still a smaller overhead than what using a deeper network would result in. This makes attention modules seem to offer a nearly free performance improvement. However, in practice these networks will be running on a GPU, which is not efficient for computing the small inner products inside attention modules, especially for batch size unity,



(a) FLOPs (b) Parameters

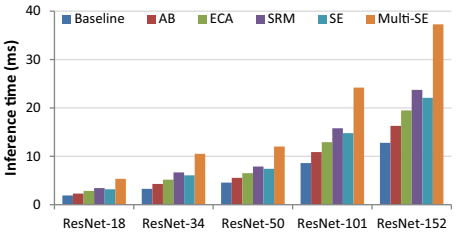
Fig. 2 Comparison of FLOPs and parameters for several ResNet depths. Regardless of depth, the computational overhead of attention modules appears to be minimal, while the overhead in parameters is only noticeable for bottleneck-based networks with SE-based attention

hence the overhead in computation time can be significantly larger than the number of FLOPs suggest.

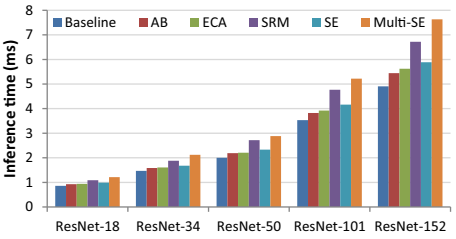
4.3.1 Attention Modules versus Network Depth

To investigate the real impact of attention modules on inference time, we measure the inference time with unity batch size. We perform this experiment for PyTorch and TensorRT. PyTorch is commonly used for experimentation, while TensorRT is a vendor-supplied optimization engine that typically results in the fastest inference. Time is measured without image loading or pre-processing, using a single image from ImageNet (224×224 pixels) on an RTX 2080 Ti and 32-bit precision. We report the lowest time out of 1000 trials to ensure that outliers due to other processes are unlikely to accidentally interfere with the timing measurement. For fair comparison, each attention module has been re-implemented more efficiently with similar operators, typically reducing inference time by 5–30% from the publicly available implementations. The results are shown in Fig. 3a and b for PyTorch and TensorRT, respectively. Despite the negligible FLOPs increase, the attention modules have a comparatively large impact on execution time. For example, ResNet-18 with the Multi-SE module performs inference slower than baseline ResNet-50 in PyTorch. TensorRT reduces total inference time with a factor 2-4 and also reduces the relative impact of attention modules. Lowering precision to 16 bits results in a similar trend, although the relative overhead of attention modules is higher, as shown in Fig. 3c. Hence, when using 16-bit precision inference, attention modules are less interesting from a speed-to-performance point of view. This effect is expected to remain even when efficient 16-bit implementations are available for all modules, because the bottleneck at this resolution is not in the parallel computing. In all cases, even when using TensorRT, the relative growth in inference time (5–50%) remains significantly larger than the FLOPs suggest (0.2–3%).

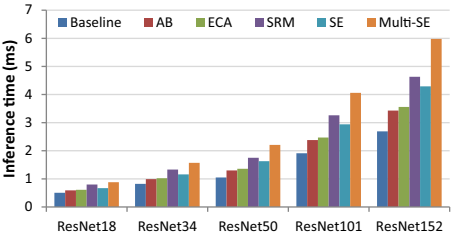
Fig. 4 shows the relative inference time increase of using attention modules versus the Top-1 accuracy achieved on ImageNet. To prevent clutter, only ResNet-18 and ResNet-50 are shown with attention modules. Considering the relatively large computational overheads of attention modules, one may wonder whether they are worth using at all. The answer is *sometimes*. For bottleneck-based ResNets (50, 101 and 152), adding an attention module generally results in an improved accuracy at a reduction in computation time, compared to a deeper ResNet. This is not the case for basic block-based ResNets (18, 34), which validates that the number of feature channels is essential for efficient attention module performance. When using TensorRT, this conclusion remains the same, but overhead is more favorable



(a)PyTorch FP32



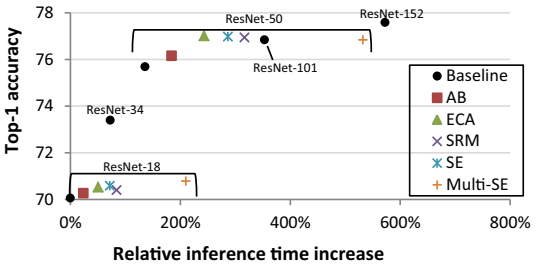
(b)TensorRT FP32



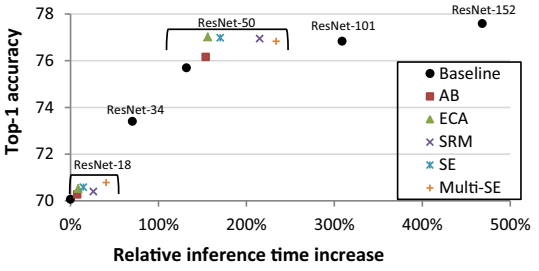
(c)TensorRT FP16

Fig. 3 Impact of attention modules on inference time of a single 224 × 224 image using an RTX 2080 Ti, using two different frameworks

Fig. 4 Trade-off between accuracy and computation time for ResNets with attention modules, on ImageNet, using the PyTorch or TensorRT framework. Our AB module is an efficient improvement in PyTorch, but starts falling behind other modules when using TensorRT

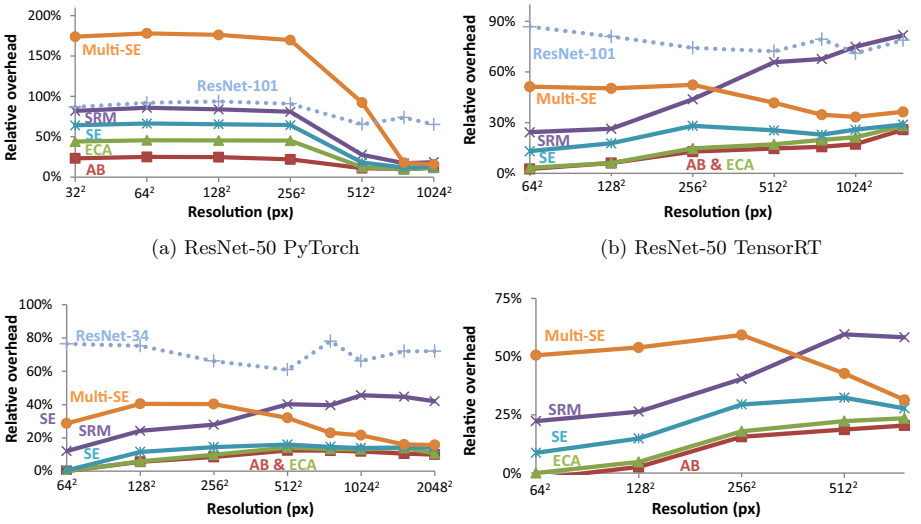


(a) PyTorch



(b) TensorRT

for attention modules. When using TensorRT, ECA-net has an especially good speed-to-accuracy trade-off, due to its computational cost becoming nearly equivalent to our proposed AB module after TensorRT optimization. However, note that this graph only shows the ImageNet results and different attention modules perform best for different datasets.



(a) ResNet-50 PyTorch

(b) ResNet-50 TensorRT

(c) ResNet-18 TensorRT

(d) ResNet-152 TensorRT

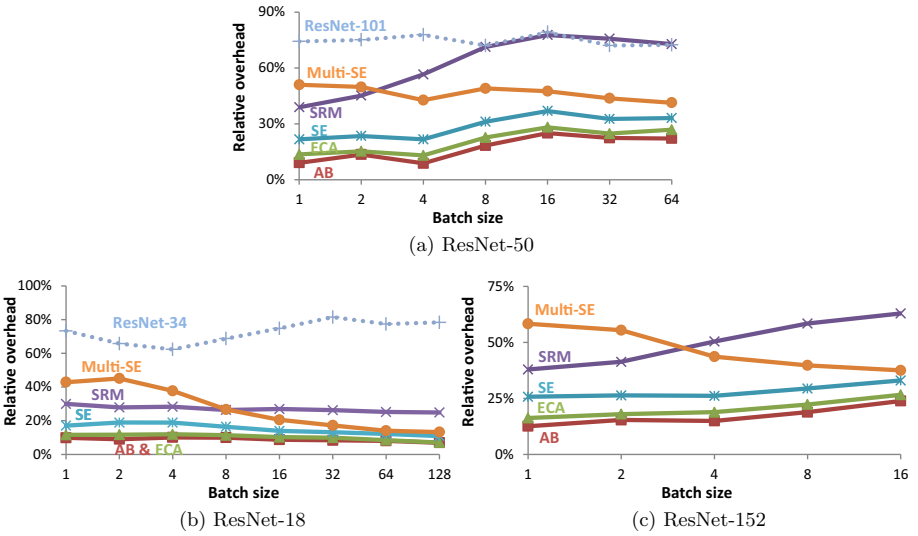
Fig. 5 Relative inference time overhead of various attention modules using a ResNet-50 and single image inference in PyTorch and TensorRT, for square images of varying size. The inference time increase of a ResNet-101 relative to ResNet-50 is also depicted, to show that PyTorch specifically has trouble with efficiently computing attention modules. In subfigure (b), ECA and AB overlap almost completely

4.3.2 Resolution Dependence

Different applications employ images of variable sizes. Hence, we investigate the impact of different image resolutions on inference time. Generally, attention modules consist of resolution-dependent components ("pooling" and "integration" in Fig. 1) and resolution-independent components ("processing" in Fig. 1). Intuitively, it may be expected that the impact is especially large for small resolutions. Image size versus relative inference time overhead is shown in Fig. 5a and b when using PyTorch and TensorRT, respectively. PyTorch introduces significant overhead for small image resolutions, whereas TensorRT shows more constant overhead. The relative computation overhead in PyTorch drops drastically for images larger than 512×512 pixels. The increased computation time of ResNet-101 compared to ResNet-50 is constant with respect to resolution, for both PyTorch and TensorRT. This suggests that the PyTorch implementation of attention modules is inefficient for low resolutions.

As expected, for both PyTorch and TensorRT, the resolution-dependent components dominate the overhead for large resolutions. The resolution-independent component is clearly visible from the difference between Multi-SE and SE in Fig. 5b. Except for Multi-SE with its expensive resolution-independent component, attention modules are, surprisingly, less attractive when using larger image resolutions when using TensorRT. This effect persists when using different network depths, as shown in Fig 5c and d.

Deeper analysis into computational complexity reveals that at high resolutions, for all modules except SRM, the majority of the computation power is used for the "integration" part of the module (see the bottom of Fig. 1). Hu et al. [3] show that a reduction in computation time for this part can be achieved with a more efficient CUDA implementation. Using this optimization is possibly more efficient than the summation that we use in AB. However,



(a) ResNet-50

(b) ResNet-18

(c) ResNet-152

Fig. 6 Relative inference time overhead of various attention modules using a ResNet-18, ResNet-50 or ResNet-152 using TensorRT, for varying batch size with fixed resolution of 224×224 pixels. Note that the x-axis ends at different batch sizes for different network depths due to memory constraints

without this optimization, summation saves the computation of the sigmoid, hence is slightly faster.

As a final experiment, we scale the batch size at fixed 224×224 pixel resolution, shown in Fig. 6. Larger batch sizes allow the GPU to execute the small inner products of the resolution-independent components in parallel, allowing more efficient computation, similar to scaling resolution. However, in contrast to resolution scaling, the resolution-dependent components scale linearly (constant relative overhead) with larger batch sizes, which makes the use of attention modules more attractive. Because of this scaling effect, high resolutions (above 1024^2 pixels) and at least medium batch sizes (above 16) are most beneficial for complex attention modules such as Multi-SE, while simple attention modules are preferred for low-resolution single-image inference.

4.4 Object Detection and Domain Generalization

In addition to our classification experiments, we evaluate two different object detection architectures on the BDD100k [29] and COCO [30] datasets.

On the COCO dataset, we have trained a standard Faster-RCNN [31] with a feature pyramid [32] and a ResNet-50 backbone, using the MMDetection framework [33]. The attention modules are only added to the backbone, the same as before at each ResNet block, resulting in a total of 16 attention modules. First, we have compared all attention modules by initializing the network with our pretrained ImageNet networks, containing pretrained attention modules. Second, we have repeated these experiments, but initialized with pretrained baseline ResNet-50 weights and applied default initialization for the attention modules. This comparison allows us to verify the hypothesis from the ImageNet fine-tuning experiment, as in Sect. 4.2. Resulting Average Precision (AP) scores are computed on the COCO minival dataset, using 5k images for validation and adding the remaining 35k validation images to the

Table 4 Average precision on the COCO minival dataset using ResNet-50 Faster-RCNN with attention modules. "Pretrained" refers to the pretrained weights from our ImageNet experiments, "Scratch" refers to training the attention modules from scratch, while still using pretrained weights of the baseline network)

Model	Pretrained		"Scratch"	
	AP@IoU=0.5	AP	AP@IoU=0.5	AP
Baseline	57.4 (+0.0)	35.6	57.4 (+0.0)	35.6
AB	57.8 (+0.4)	35.7	57.6 (+0.2)	35.6
ECA	59.0 (+1.6)	36.5	**58.9** (+1.5)	**36.6**
SE	59.2 (+1.8)	36.9	57.8 (+0.4)	35.8
SRM	57.9 (+0.5)	35.7	57.6 (+0.2)	35.6
Multi-SE	**59.3** (+1.9)	**37.2**	57.8 (+0.4)	35.8

Best scores are made bold

training set, as is common practice. The results of both experiments are shown in Table 4. In contrast to our ImageNet fine-tuning experiment, using pretrained attention module weights in the backbone appears to be required to achieve a large part of the performance gain, except for ECA. This could potentially be alleviated by increasing the number of training epochs [34], but this negates the train-time advantage, hence this approach is not investigated further.

Domain generalization. BDD100k is an object detection dataset for autonomous driving and includes domain labels for time of day and weather. This makes it an excellent dataset for determining the impact of network changes on their domain generalization capabilities. Attention modules are tested for domain generalization because some domain generalization methods are also based on altering means and variances of feature maps, similarly to channel attention modules, and may thus perform a similar role in the network. Two such methods are Instance-Batch Normalization (IBN) [19] and style adversarial training [35].

We have trained a ResNet-34 based 512 × 512 SSD [36] detector, following the network modifications and parameters used in ScratchDet [37], so that it can be trained from scratch. Again, the attention modules are added only to the backbone, which results in 16 modules for ResNet-34. First, for each attention module type, we have trained the SSD from scratch for 20 epochs on the 100,000 training images and report the AP@IoU=0.5 on the validation set. We have then repeated this experiment, but trained only on all "day" and "dawn/dusk" images of the dataset and validated on the "night" images of the validation set. The results are shown in Table 5. Surprisingly, our AB module is the only module that obtains a significant improvement, outperforming the more expensive modules. To verify whether this is an outlier, we have retrained with AB two more times, resulting in a standard deviation of 0.2, which clearly suggests the obtained result is not an outlier.

Next, we compare the effect of domain generalization methods to that of attention modules for the day/night generalization experiment. For IBN blocks, we apply instance normalization on the first two residual blocks, according to the IBN-b scheme of Pan *et al.* [19]. For style adversarial learning, we compute the mean and variance of the final layer of the second residual block, followed by a gradient reversal layer and two FC layers, using the parameters of [35]. For adversarial training, we use both time of day and weather labels, to make the network invariant to these domains. The results are shown in Table 5. The performance improvement of these domain generalization methods is similar to or worse than when using attention modules. Furthermore, combining IBN blocks with AB or SE diminishes the improvement of IBN, suggesting that attention modules by themselves already reach most of the gain that can be achieved from modifying the means and variances of the feature maps.

Table 5 Average precision of ResNet-34 SSD detector on the BDD100k dataset, trained from scratch ("All" represents evaluation on full dataset, "Day/night" shows training on "day" and "dawn/dusk" , testing on "night")

Model	All AP@IoU=0.5	Day/night AP@IoU=0.5
Baseline	43.9	30.1
AB	**44.8**	30.7
ECA	44.1	31.0
SE	44.0	30.9
SRM	43.6	28.0
Multi-SE	44.0	30.7
Style-adversarial	–	30.2
IBN	–	30.9
AB + IBN	–	30.5
SE + IBN	–	**31.1**

Best scores are made bold

5 Discussion

This paper provides an objective comparison of several attention modules under identical conditions. Generally, attention module papers only compare their numerical scores with previously reported results, instead of experimentally validating the different modules in an identical experimental setup. We have re-implemented each attention module, aligning common operators by replacing them with identical, more computationally efficient alternatives. Although more work, this results in a more objective comparison.

Adding attention modules to a network changes the location of its global minimum, but not very strongly, as shown by the results in Sects. 4.2 and 4.4. This allows adding randomly initialized attention modules to a pretrained network and fine-tuning the extended network with a lower learning rate than training from scratch. This fine-tuning always outperforms a network that is fully trained from scratch, in half as many epochs (40 epochs fine-tuning vs. 90 epochs from scratch).

From an execution-time perspective, adding attention modules should be viewed as an alternative to increasing network depth. In either way the network becomes larger, but the impact on the trade-off of inference time versus accuracy is different. The number of FLOPs suggest that attention modules are always the more efficient option, but measuring the inference time shows that this is highly dependent on other factors, such as architecture, resolution, batch size and framework.

Channel attention modules may be a replacement of certain domain generalization techniques, as they serve a similar role and modify feature maps in a similar way. For example, in one of our experiments it was found that using both simultaneously does not improve performance further. As future work, it is interesting to investigate this relation between domain generalization and channel attention and discover whether related techniques have a complementary nature or not.

6 Conclusion

We have performed an extensive comparison in terms of speed and accuracy with computationally inexpensive attention modules, for both classification and object detection. All

attention modules have been re-implemented more efficiently and similarly, under identical conditions, for a fair comparison (code available). With respect to the speed-to-performance trade-off, we propose a novel attention module, the Attention Bias (AB) module, which achieves the lowest overhead when used for inference. This module can be easily integrated into pretrained networks, achieving higher performance after minimal fine-tuning. However, none of the tested modules, including our AB module, are consistently the best over different datasets and network architectures and every tested module performed best in at least one experiment. This means that selecting the best attention module for a practical use case cannot be simplified to choosing one with e.g. the best performance on ImageNet.

We extend the theoretical complexity measurement of FLOPs with inference time measurements in the commonly used PyTorch and TensorRT frameworks. Our measurements show that inference time increases far more (5–50%) than the increase in FLOPs suggests (0.2–3%). This means that attention modules are not always a computationally efficient addition to the architecture. Their effectiveness depends on framework, network architecture, batch size and input resolution. For a ResNet with bottleneck blocks, using attention modules is generally preferred over using a deeper architecture. In contrast, ResNets of basic blocks perform poorly, suggesting sufficient feature channels are essential. Complex attention modules (Multi-SE) are preferred for larger resolutions and batch sizes, because the computation bottleneck is in the resolution-independent components. It can be concluded that simple modules (ECA, AB) are preferred for low-resolution single-image inference.

Finally, we have found that the difference in increased accuracy is often relatively smaller than the difference in inference time. This simplifies the choice of the selection of a suitable module for specific use cases. Merely benchmarking the inference time of several modules without any training can provide a suitable selection of modules for further evaluation in the complete training pipeline. Moreover, then these modules can be integrated by fine-tuning a pretrained network, reducing training time by a factor of two compared to training from scratch.

Author Contributions All authors contributed to the study conception and design. Execution of the experiments and the writing of the first draft were performed by SRK. Regular discussions performed with RGJW and PHNdW allowed for refining the experimental design and for improving the draft text to its current state.

Funding This study was funded by the Efficient Deep Learning (EDL) program (https://efficientdeeplearning.nl/), which itself is funded by the The Dutch Research Council (NWO), and 35 Dutch companies. Authors S.R. Klomp and R.G.J. Wijnhoven are employees of ViNotion B.V., a company that performs traffic surveillance image analysis using neural networks and who provided part of the computational power for running the experiments. The rest of the computational resources have been provided by the Eindhoven University of Technology, where P.H.N. de With is a professor and S.R. Klomp a PhD student.

Data Availability All datasets used by this study are already public and can easily be found online on their respective web pages: ImageNet (https://www.image-net.org/), COCO (https://cocodataset.org) and BDD100k (https://www.bdd100k.com/).

Code Availability Code for all attention modules and the ImageNet experiments is available on GitHub: https://github.com/SanderKlomp/channel-attention and is an extension of an earlier publicly released code sample of Lee et al. [10].

Declarations

Conflict of interest Outside of the funding discussed in the previous item, there are no additional competing interests for any of the authors that are relevant to the content of this article.

Consent to participate Not applicable, because only publicly available data was used.

Consent for publication All authors read and approved the final manuscript.

References

1. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. Microsoft Research Asia
2. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: 32nd International Conference on Machine Learning, ICML 2015, vol. 1, pp. 448–456
3. Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. CVPR. https://doi.org/10.1109/CVPR.2018.00745
4. Huang Z, Liang S, Liang M, Yang H (2020) DIANet: dense-and-implicit attention network. In: AAAI, pp. 4206–4214. arXiv:1905.10671
5. Zhang H, Wu C, Zhang Z, Zhu Y, Zhang Z, Lin H, Sun Y, He T, Mueller J, Manmatha R, Li M, Smola A (2020) ResNeSt: Split-Attention Networks. arXiv preprint arXiv:2004.08955
6. Chen X, Yu J, Wu Z (2020) Temporally identity-aware SSD with attentional LSTM. IEEE Trans Cybern 50(6):2674–2686. https://doi.org/10.1109/TCYB.2019.2894261
7. Xu Z, Zhuang JBQL, Zhou J, Peng S (2018) domain attention model for domain generalization in object detection. pattern recognition and computer vision. PRCV 2018 **11259**. https://doi.org/10.1007/978-3-030-03341-5
8. Wang X, Cai Z, Gao D, Vasconcelos N (2019) Towards universal object detection by domain attention. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7281–7290. https://doi.org/10.1109/CVPR.2019.00746
9. Wang Q, Teng Z, Xing J, Gao J, Hu W, Maybank S (2018) Learning Attentions: Residual Attentional Siamese Network for High Performance Online Visual Tracking. In: CVPR2018, pp. 4854–4863. https://doi.org/10.1109/CVPR.2018.00510
10. Lee H, Kim H-E, Nam H (2019) SRM : A style-based recalibration module for convolutional neural networks. In: ICCV, pp. 1854–1862. arXiv:1903.10829
11. Wang Q, Wu B, Zhu P, Li P, Zuo W, Hu Q (2020) ECA-Net: Efficient channel attention for deep convolutional neural networks. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 11531–11539. https://doi.org/10.1109/cvpr42600.2020.01155
12. Krizhevsky A, Sutskever I, Hinton GEGE, Sulskever I, Hinton GEGE (2012) ImageNet Classification with Deep Convolutional Neural Networks. In: Advances in Neural Information and Processing Systems (NIPS)
13. Jia Deng, Wei Dong, Socher R, Li-Jia Li, Kai Li, Li Fei-Fei (2009) ImageNet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 248–255. https://doi.org/10.1109/CVPRW.2009.5206848
14. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: ICLR. https://doi.org/10.1016/j.infsof.2008.09.005
15. Xie S, Girshick R, Dollár P, Tu Z, He K (2017) Aggregated residual transformations for deep neural networks. In: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, pp. 5987–5995. https://doi.org/10.1109/CVPR.2017.634
16. Geirhos R, Michaelis C, Wichmann FA, Rubisch P, Bethge M, Brendel W (2019) ImageNet-trained CNNs are biased towards texture. ICLR, increasing shape bias improves accuracy and robustness
17. Huang X, Belongie S (2017) Arbitrary style transfer in real-time with adaptive instance normalization. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2–4. https://doi.org/

10.1109/ICCV.2017.167. http://openaccess.thecvf.com/content_ICCV_2017/papers/Huang_Arbitrary_Style_Transfer_ICCV_2017_paper.pdf
18. Ulyanov D, Vedaldi A, Lempitsky V (2017) Instance Normalization: The missing ingredient for fast stylization. arXiv:1607.08022
19. Pan X, Luo P, Shi J, Tang X (2018) Two at Once : enhancing learning and generalization capacities via IBN-Net. In: CVPR
20. Hu J, Shen L, Albanie S, Sun G, Vedaldi A (2018) Gather-excite: Exploiting feature context in convolutional neural networks. In: advances in neural information processing systems (NeurIPS), pp. 9401–9411
21. Hu X, Zhang Z, Jiang Z, Chaudhuri S, Yang Z, Nevatia R (2020) SPAN: spatial pyramid attention network for image manipulation localization. In: ECCV2020, pp. 312–328
22. Jaderberg M, Simonyan K, Zisserman A (2015) spatial transformer networks. In: Advances in Neural Information Processing Systems (NeurIPS), pp. 2017–2025. https://doi.org/10.1145/2948076.2948084
23. Wang X, Girshick R, Gupta A, He K (2018) Non-local neural networks. In: proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
24. Woo S, Park J, Lee J-y, Kweon IS (2018) CBAM: convolutional block attention module. In: European conference on computer vision (ECCV)
25. Bello I, Zoph B, Le Q, Vaswani A, Shlens J (2019) Attention augmented convolutional networks. In: proceedings of the IEEE international conference on computer vision (CVPR), pp. 3285–3294. https://doi.org/10.1109/ICCV.2019.00338
26. Zhang S, Yang J, Schiele B (2018) Occluded pedestrian detection through guided attention in CNNs. In: CVPR, pp. 6995–7003. https://doi.org/10.1109/ICCChina.2012.6356930
27. Cao Y, Xu J, Lin S, Wei F, Hu H (2019) GCNet: Non-local networks meet squeeze-excitation networks and beyond. In: Proceedings - 2019 international conference on computer vision workshop, ICCVW, pp. 1971–1980. https://doi.org/10.1109/ICCVW.2019.00246
28. Ma X, Guo J, Chen Q, Tang S, Yang Q, Fu S (2020) Attention meets normalization and beyond. In: IEEE international conference on multimedia and expo (ICME). https://doi.org/10.1109/ICME46284.2020.9102909
29. Yu F, Chen H, Wang X, Xian W, Chen Y, Liu F, Madhavan V, Darrell T (2020) BDD100K: A diverse driving dataset for heterogeneous multitask learning. In: CVPR 2020, pp. 2633–2642. https://doi.org/10.1109/cvpr42600.2020.00271
30. Microsoft COCO (2014) Lin, T.-Y.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L. Common objects in context. In: ECCV 8693:740–755
31. Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: Towards real-time object detection with region proposal networks. In: NeurIPS, pp. 91–99. https://doi.org/10.1109/TPAMI.2016.2577031
32. Lin T-Y, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: CVPR. https://doi.org/10.1109/CVPR.2017.106
33. Chen K, Wang J, Pang J, Cao Y, Xiong Y, Li X, Sun S, Feng W, Liu Z, Xu J, Zhang Z, Cheng D, Zhu C, Cheng T, Zhao Q, Li B, Lu X, Zhu R, Wu Y, Dai J, Wang J, Shi J, Ouyang W, Loy CC, Lin D (2019) MMDetection: Open MMLab detection toolbox and benchmark. arXiv:1906.07155
34. He K, Girshick R, Dollar P (2019) Rethinking imageNet pre-training. In: proceedings of the IEEE international conference on computer vision (CVPR), pp. 4917–4926. https://doi.org/10.1109/ICCV.2019.00502
35. Nam H, Lee H, Park J, Yoon W, Yoo D (2019) Reducing domain gap via style-agnostic networks. In: ICCVW. arXiv:1910.11645
36. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-YY, Berg AC (2016) SSD: Single shot multibox detector. In: ECCV, vol. 9905 LNCS, pp. 21–37
37. Zhu R, Zhang S, Wang X, Wen L, Shi H, Bo L, Mei T (2019) Scratchdet: Training single-shot object detectors from scratch. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2263–2272. https://doi.org/10.1109/CVPR.2019.00237