ISBN: 978-981-18-7950-0
2023 the 13th International Workshop on Computer Science and Engineering (WCSE 2023)
doi: 10.18178/wcse.2023.06.022

T-SSD: A Transformer-based Single-Stage Multi-Scale Sampling Object Detector

Kailai Huang+, Mi Wen, Chen Wang, Lina Ling

College of Computer Science and Technology, Shanghai University of Electric Power, China

Abstract. The object detection algorithms are the cornerstones of autonomous driving systems, they are mostly based on convolutional neural networks (CNNs) with one or two stages. Since its strong correlation with the life safety of the driver, the accuracy of object detectors is crucial and limited by its foundation, CNN, which is hard to improve nowadays. But at the same time, the basic transformer shows its better performance compared with the advanced CNN. To improve the accuracy, using transformers seems to be a better choice. However, most transformer-based detectors are only backbone replacements, ViT concept extension, or a fusion with CNN, cannot give a full play to the performance referring to the characteristics of the transformer. We proposed a single-stage object detector T-SSD (Transformer-based Single-Stage Detector) that comes with a multi-scale feature modeling ability. The transformer backbone extracts feature in different scales and aggregates them into an intermediate representation. The transformer neck then directly queries the semantic information from the aggregated representation and feed them to heads to make prediction once and for all. After training on COCO2017, by combining the construction philosophy of the object detector and the characteristics of transformers, our T-SSD-Tiny gives an AP (Average Precision) up to 9.0 higher than the CNN-based detectors with 100 fewer epochs, better than YOLOv3-Base and SSD-300. Also, the AP given by our T-SSD-Small is up to 4.7 higher than the transformer-based detector with the same epoch, indicating a comparable performance with DETR-ResNet-101 and YOLOS-Small.

Keywords: Transformer, Vision Transformer, Object Detection, One-Stage Algorithm.

1. Introduction

Object detectors are generally composed by three components: backbone, neck, and heads. The backbone is the foundation of the overall architecture, it extracts features whereas the neck is normally an optional module to perform further operations.

Fig. 1: The attention of T-SSD on an image of COCO2017 dataset. The area under the white halo is exactly the area the model pays more attention to.

Backbones are often high-performance CNN-based image classifiers during the last decade, but transformers [1,2] based on multi-head attention mechanisms have progressively arisen now, showing its stronger generalization ability compared with the most used CNN while transferring from language modeling to vision tasks. Various basic transformers [3,4] have outperformed widely adopted CNNs [5,6]. Providing us better backbone choices, building a transformer-based object detector has become a new trendy task.

Traditional CNN-based object detectors include two categories: one-stage and two-stage, with the migration to transformer architecture, some detectors have followed the lead of these design philosophies.

+ Corresponding author. Tel.: +86-16628586267; fax: +86-16628586267.
E-mail address: yyyyyoc@hotmail.com.

By using a transformer as the backbone substitution of a two-stage detector, a transformer-based detector can be obtained [7], it not only inherits the drawbacks of its father but also bears the constraints of its own shortcomings. Also, there are other models applying the vanilla transformer as a one-stage transformer-based detector [8], it suffers the unfortunate consequence of being difficult to train which is brought by the simplicity of its own structure and needs a lot of fancy training strategies to achieve the goal of single-stage detection. Moreover, some models hybrid the CNN feature extractor with a prediction querying transformer [9]. Even though CNN has an innate advantage in processing 3D image data, the transformer is designed to process 2D sequence data. In such a mixed structure, the inductive bias definitely suffers a hurting loss during the transforming action. So, we can draw a conclusion that these transformer-based object detection models just mentioned still got defects that can be improved or avoided, some even cannot be called transformer-based strictly.

To solve and avoid these annoying problems, we proposed T-SSD, whose name pays tribute to SSD [10], without borrowing any help from CNN, fully built by transformer. It comes with multi-scale sampling ability plus a feature aggregation strategy, and directly queries a certain number of boxes and classes from the aggregated representation, which makes it a single-stage object detector. T-SSD mainly consists of three parts: a transformer backbone with multi-scale sampling and modeling capabilities, a transformer neck that can query results directly from intermediate representation, and a set of MLP predictors that give us the final prediction.

Our main contributions can be summarized as follows:

- T-SSD was purely made by transformer with the one-stage operation, its backbone and neck are both transformer, while the backbone aggregates multi-scaled representations in the process of sampling and the neck makes queries once and for all. The whole process is conducted by attention operations from the beginning to the end to avoid the problems caused by model fusion to give a full play to the performance of the transformer.
- With the aggregated representations given by the process of sampling from backbone, the neck makes queries once and for all. The whole process is conducted by attention operations from the beginning to the end, this pure transformer was designed to avoid the problems caused by model fusion to give a full play to the performance of the transformer.
- Whether under fewer epochs or the same training settings, T-SSD has a higher AP than CNN-based or transformer-based detectors respectively. Not only this, the training strategy is clear and intuitive compared with the sophisticated distilled training method.

2. The Proposed T-SSD Method

The overall architecture of our proposed T-SSD pipeline is illustrated in Fig.2. Following the designing philosophy of classical object detectors, our proposed model also contains three major components: 1) a transformer-based backbone with multi-scale sampling ability, see Sec.2.1, 2) a transformer-based object query neck, see Sec.2.2 and 3) a box prediction head and a class prediction head, see Sec.2.3.

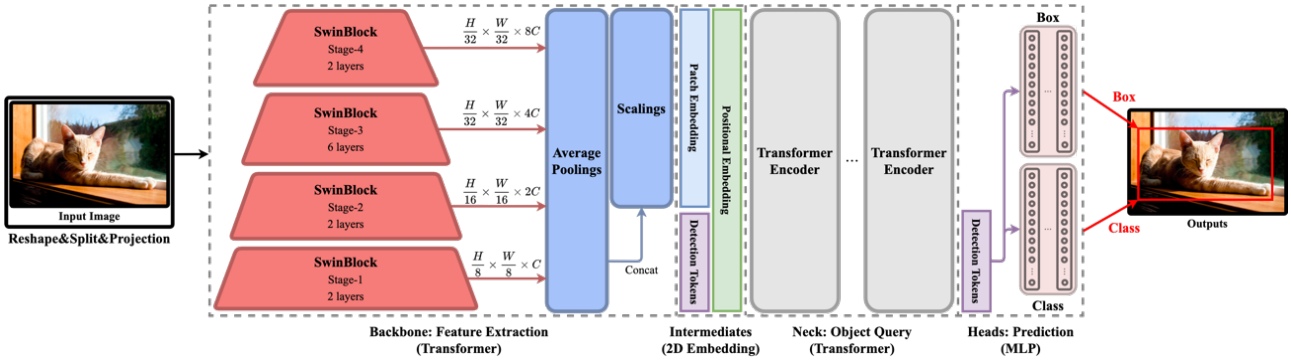

Fig. 2: Overall architecture of T-SSD.

2.1. Backbone: Feature Extraction

Our feature extracting backbone consists of a patch projection layer, a stack of SwinBlocks, and scale controllings: AvgPool layer and Scaling layer. For illustrations, see the feature extraction module in Fig.2.

The input image is $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$ (3 because they are RGB images), before goes to the SwinBlocks, $\mathbf{x}$ needs to be split into non-overlapping and same-sized mini-patches and project them from 3-dimensional patch tokens into 2-dimensional embeddings like ViT does. These patches $\mathbf{p} \in \mathbb{R}^{4 \times 4 \times 3}$ are reshaped to ^{48} (4×4×3=48) and concatenate to a 2-dimensional sequence with sequence length of $L_0 = H/4 \times W/4$. Then a projection layer projects the sequence $\mathbf{x}_{p0} \in \mathbb{R}^{L_0 \times 48}$ from 48 to an arbitrary dimension C, resulting in $\mathbf{x}_{p0} \in \mathbb{R}^{L_0 \times C}$.

During the process of the following stages of shift-windowed operation blocks, each stage of them will give us an intermediate output in different shapes. Inspired by SSD, we want to make the maximum utilization of these 4 different scaled intermediate outputs in Eqs.(1):

$$\begin{aligned}
&\mathbf{x}_{p1} \in \mathbb{R}^{L_1 \times C}, \quad L_1 = \frac{H}{8} \times \frac{W}{8}. \\
&\mathbf{x}_{p2} \in \mathbb{R}^{L_2 \times 2C}, \quad L_2 = \frac{H}{16} \times \frac{W}{16}. \\
&\mathbf{x}_{p3} \in \mathbb{R}^{L_3 \times 4C}, \quad L_3 = \frac{H}{32} \times \frac{W}{32}. \\
&\mathbf{x}_{p4} \in \mathbb{R}^{L_4 \times 8C}, \quad L_4 = \frac{H}{32} \times \frac{W}{32}.
\end{aligned} \tag{1}$$

For dimensional considerations, the following module requires $\mathbf{x}_{p5} \in \mathbb{R}^{L_5 \times D}$ (L_5 and D are hyperparameters, the former is the scale of your embedding scale, the latter is the dimension that the detecting transformer neck expects). In the consideration of concatenation, these 4 embeddings must be fixed to a unified channel dimension.

Meanwhile, the length of these 4 features embedding sequences needs to be condensed: before unified-scaling, the feature sequence is first reshaped from a 2-dimensional sequence to a 3-dimensional feature map in a squared shape, avgpool then shrinks the 3-dimensional feature into a fixed smaller edge size E in the condition of $L_5 = 4 \times E^2$. In this case, the output size of feature extraction is $\mathbf{x}_{p5} \in \mathbb{R}^{L_5 \times D}$. Process illustration equations see Eq.(2).

$$\begin{aligned}
&\mathbf{z}_{0i} = \mathrm{Linear}(\mathrm{AvgPool}_i(\mathbf{x}_{pi})), \quad i = 1...4 \\
&\mathbf{z}_0 = [\mathbf{z}_{01}; \ldots; \mathbf{z}_{04}].
\end{aligned} \tag{2}$$

2.2. Neck: Object Query

The step of feature extraction gives us an aggregated patch embedding $\mathbf{x}_{p5} \in \mathbb{R}^{L_5 \times D}$. Now, we have to think about how to predict the boxes and classes from it. Since we are designing a single-stage detector and not using default boxes or anchor boxes. Directly querying all of the boxes and classes from the feature sequence seems to be a perfect solution for us. The equation of the standard self-attention mechanism is illustrated in Eq.(3).

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3}$$

Where Q, K, and V are model parameters which have the same shape as their input, and d_k is the dimension of Q and K. Considering the expressive ability of T-SSD, we use multi-head self-attention, which can be seen in Eq.(4), where k is the number of heads.

$$\mathrm{MultiHead}(\mathbf{z}) = \left[\mathrm{Attention}_1(z); \cdots; \mathrm{Attention}_k(z)\right] \tag{4}$$

Therefore, for one layer of encoder, see Eq.(5), note that there are layer normalizations before each suboperations in the equation.

$$\mathbf{z}' = \mathrm{MultiHead}(LN(\mathbf{z}_0)),$$
$$\mathbf{z} = \mathrm{MLP}(LN(\mathbf{z}')). \tag{5}$$

In BERT and ViT, the projected patch embedding is concatenated with a learnable class embedding $[\mathbf{cls}] \in \mathbb{R}^{1\times D}$, it is used for making attention to the patch embedding and understand the class of the image. In our proposed method, we follow this instinct and use a number of detection tokens to fulfil the goal of querying N objects (N is also the number of detection tokens) from $\mathbf{x}_{p5}$. At this point, the patch embedding is concatenated with $[\mathbf{det}] \in \mathbb{R}^{N\times D}$, then a same-sized positional embedding $\mathbf{PE} \in \mathbb{R}^{(L_5+N)\times D}$ is added to the former embedding, thus, the general integral embedding $\mathbf{x}_{p6} \in \mathbb{R}^{(L_5+N)\times D}$. Operation logic see Eq.(6).

$$\mathbf{z}_1' = [\mathbf{z}_0; \mathbf{det}],$$
$$\mathbf{z}_1 = \mathbf{z}_1 + \mathbf{PE}. \tag{6}$$

Since we want to query the embedding which concatenated with $[\mathbf{det}]$, here, for detecting query purpose, we use stacked layers of transformer encoder, eventually, output the $[\mathbf{det}]$ solely. Notice that, the dimension of the embedding do not change. The processing logic illustrated in Eq.(7).

$$\mathbf{z}_2' = [\mathrm{Encoder}(\mathrm{Encoder}(\ldots(\mathrm{Encoder}(\mathbf{z}_1))))],$$
$$\mathbf{z}_2 = [\mathbf{z}_2'^{L_5}; \ldots; \mathbf{z}_2'^{L_5+N}]. \tag{7}$$

2.3. Heads: Final Prediction

The neck of T-SSD only output the detection tokens $[\mathbf{det}]$, the job of heads is to predict the coordinates of the bounding-boxes and the classes of the objects eventually. So, in the end of the whole procedure, lies two heads: a box head and a class head. They are made of MLPs (Multi-Layer Perceptron, stacking of linear layers), heads can be seen in Eq.(8).

$$\hat{y}_{box} = \mathrm{MLP}(\mathbf{z}_2),$$
$$\hat{y}_{cls} = \mathrm{MLP}(\mathbf{z}_2). \tag{8}$$

The heads are giving us the prediction of all objects at the same time, predicted boxes $\hat{y}_{box} \in \mathbb{R}^{N\times 4}$ and $\hat{y}_{cls} \in \mathbb{R}^{N\times \mathrm{NumClasses}}$. Finally, the prediction of T-SSD, $\hat{\mathbf{y}}$, is shown in Eq.(9).

$$\hat{\mathbf{y}} = [\hat{y}_{box}; \hat{y}_{cls}] \tag{9}$$

By assembling these 3 main components together, we will get the T-SSD model that predict a set of bounding-boxes and category classes with number of N in a single query operation.

3. Result Analysis

Fig. 3: The boxes, classes and attention map resulting on image. The attention map is the accumulation of the activations of intermediate output from backbone.

By assembling these 3 main components together, we will get the T-SSD model that predict a set of bounding-boxes and category classes with number of N in a single query operation.

3.1. Dataset

We trained T-SSD on COCO 2017 [11] dataset. The quantity of training set and validation set contains 118287 and 5000 RGB images separately. COCO 2017 contains different types of annotation for different

tasks, for instance: object detection, panoptic segmentation, keypoint caption and etc., we choose the annotations for object instances detection. Each image has an average of 7 instances, in the training set, there are up to 63 instances in a single image, from small to large. The AP in tablecnn and tabletrans are bounding box AP, it's the accumulation of GIoU [12] threshold from 0.5 to 0.95 at every object scale.

3.2. Model setup

Table 1: Parameter composition

Model Name	Input Size	Backbone	Neck	Heads		Total
				Layers	Params	
T-SSD-Tiny	672	27.4M	5.7G	3	0.1M	**33.2M(33,206,688)**
T-SSD-Small	672	48.8M	22.0G	3	0.6M	**71.4M(71,379,648)**

As we illustrated in the above chapters, T-SSD mainly contains 3 parts of components: 1) backbone, for feature extraction, 2) neck, for bounding box regression, and 3) heads, for box and class prediction. Parameter composition see Table 1, the choice of our backbone is SwinTransformer, instead of migrating it intact, we only use a specific stack of SwinBlocks. The neck consists of classic transformer encoders, structurally, it is similar to DeiT, since we use a number of detection tokens, it is now more like YOLOS, and still, it is not a migrate of intact. So, if you are interested in it, you will find the parameter of our backbone and neck is less than the equivalent SwinTransformer and YOLOS.

Table 2: Pooling configurations

Model Name	Input Size	AvgPool		Output Size	Total Patches
		Filter Size	Stride		
T-SSD-Tiny	84	7	6	13	**728**
	42	5	3	13	
	21	9	1	13	
	21	9	1	13	
T-SSD-Small	84	10	5	15	**900**
	42	13	2	15	
	21	7	1	15	
	21	7	1	15	

As mentioned above, the aggregation of intermediate outputs is an important step in our model. Table 2 listed the hyperparameters of the average pooling, they are designed to shrunk the length of the embedding to save the computational cost.

3.3. Training details

The bipartite matching loss from [9] is used as the criterion measurement of T-SSD. For optimization, we use AdamW [13] and set the learning rate and weight decay to 1×10^{-4} with *cosine* learning rate decay. The trends of loss and AP can be seen in Fig.4. All of the parameters are initialized with truncated normal distribution. For both train and validation images, we convert them uniformly to the size of 672×672×3 which were normalized after to match the input requirements of SwinBlocks and raise the performance, meanwhile, a horizontal flip was used as the augmentation.

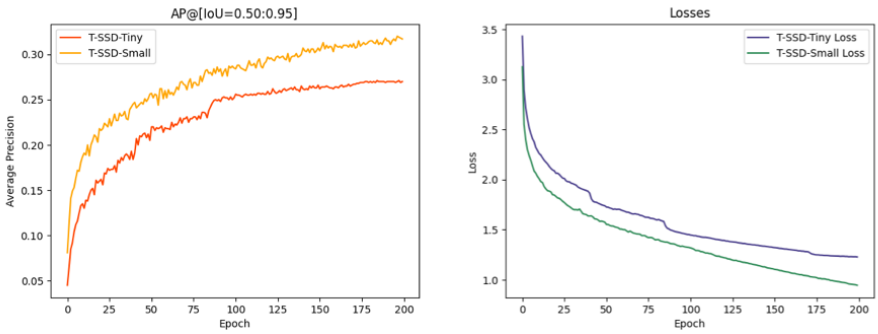

Fig. 4: The AP and Loss variation curve of T-SSD-Tiny and T-SSD-Small.

In order to accelerate the speed of model convergence and curve fitting, we used some pre-trained weights. Thanks to the awesome work of SwinTransformer and YOLOS, we can use their pre-trained weights. A number of weights of layers in SwinTransformer can be used as the initial weights of a part of weights of our backbone. As for some weights of detection tokens from YOLOS can save both some time and some labour.

The whole training and evaluation process of these various models were performed on a single node, with a machine containing 4 NVIDIA A40 GPU. The whole training time of tiny model with batch size of 31 and 200 epochs is about 60 hours, and about 96 hours for small model with 16 batch size and 200 epochs.

3.4. Comparisions with CNN-based object detectors

Table 3: Comparisions with some classical CNN-based detectors

Model Name	Input Size	Epoch	Parameters	FLOPs	AP
YOLOv3-Tiny	640	300	8.9M	14.6G	18.1
YOLOv3-Base	640	300	61.9M	162.3G	23.3
SSD-300	300	300	22.9M	197.8G	24.6
T-SSD-Tiny	**672**	**200**	**33.2M**	**86.7G**	**27.1**

We choose 3 CNN-based detectors to compare with: YOLOv3-Tiny, YOLOv3-Base and SSD-300. The setups and results can be seen in Table 3. The implementations and training method of YOLOv3s are from Ultralytics, and SSD-300's are from NVIDIA. After 300 epochs of train with an input size of 640×640, YOLOv3-Tiny and YOLOv3-Base achieved an AP of 18.1 and 23.3, which have a gap of 10.0 and 3.8 compared with the 27.1 AP of T-SSD-Tiny. The AP of T-SSD-Tiny is also 2.5 higher than SSD-300 with 300 epochs of training and an input size of 300×300. Among these classical CNN-based object detectors, our T-SSD-Tiny has the best result of 27.1AP, higher than any of them, at the same time, 100 less epochs with a decent quantity of parameter and FLOPs.

3.5. Comparisions with transformer-based object detectors

By assembling these 3 main components together, we will get

Table 4: Comparisions with some transformer-based detectors

Model Name	Input Size	Epoch	Parameters	FLOPs	AP
YOLOS-Tiny	672	300	6.5M	20.4G	21.8
YOLOS-Small	672	200	27.6M	80.4G	27.3
DETR-R34	672	200	38.7M	73.0G	24.3
DETR-R101	672	200	60.2M	147.8G	28.4
T-SSD-Small	**672**	**200**	**71.4M**	**196.1G**	**32.0**

Compared with 4 transformer-based detectors, the input size are all fixed to 672×672, total epochs are 200 except YOLOS-Tiny, which have 300 epochs. Our T-SSD-Small also gained the highest 32.0 AP, 10.2 and 4.7 higher compared with YOLOS-Tiny and YOLOS-Small, and, compared with DETR-ResNet-34 and

DETR-ResNet-101, which 7.7 and 3.6 higher. However, its shortcomings are the larger parameter quantity and higher FLOPs. The implementations and training methods are strictly followed by the official code, but with a very mild change of input image size and augmentations to be on the same starting line with our T-SSD-Small. For statistical details, see Table 4.

3.6. Visualizations

Fig.3. shows 4 groups of image, for each group, there is an original input image, a predicted image, and an attention masked image. As we can see in Fig.1 and Fig.3, the predicted boxes and attention are majorly in the overlapping area of the image, and these areas are basically the object that model interested in. As we can tell, the objects in the image are most shrouded by the attention mask, which can be inferred is that the model is indeed learning the features and locations of the objects, the method is working smoothly.

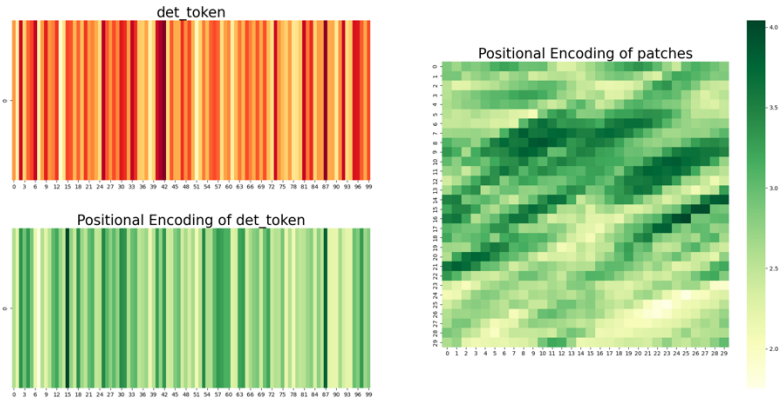

Fig. 5: The detection token, the positional embedding of detection token and aggregated patches.

And we are make a visualization of detection token, its positional embedding and the positional embedding of the aggregated feature map, see Fig.5. As we can see, the positional embedding of the patch aggregated from 4 different scales is largely different from the positional embedding of normal vision transformer [3].

4. Conclusion and Future Work

In this paper, we presented T-SSD, a single-stage object detector purely made by transformer. We demonstrated that using transformers as building blocks, drawing instincts from typical 3-level detection structure, and querying boxes from aggregated multiple-scaled feature maps straight forward, can give us a competitive single-shot detector. Not only can T-SSD give us a better result compared with CNN-based detectors whose structure and training methodology were highly-optimized, but also outperform the transformer-based detectors with the same training methodology.

However, T-SSD also has some drawbacks, such as a big number of parameters and the desire for computational resources, these are the major challenge standing in our way. In the future, exploring the method of simplifying the scale of the model takes and optimizing the FLOPs are our main goals.

Source codes are available at https://github.com/YOCdot/T-SSD.

5. Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No.U1936213, 61872230, Program of Shanghai Academic Research Leader No.21XD1421500, Shanghai Science and Technology Commission Project No.20020500600.

6. References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, \Lukasz Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).

[2] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, ArXiv Prepr. ArXiv181004805. (2018).

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, others, An image is worth 16x16 words: Transformers for image recognition at scale, ArXiv Prepr. ArXiv201011929. (2020).

[4] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou, Training data-efficient image transformers & distillation through attention, in: Int. Conf. Mach. Learn., PMLR, 2021: pp. 10347–10357.

[5] C. Szegedy, S. Ioffe, V. Vanhoucke, A.A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in: Thirty-First AAAI Conf. Artif. Intell., 2017.

[6] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016: pp. 770–778.

[7] J. Beal, E. Kim, E. Tzeng, D.H. Park, A. Zhai, D. Kislyuk, Toward transformer-based object detection, ArXiv Prepr. ArXiv201209958. (2020).

[8] Y. Fang, B. Liao, X. Wang, J. Fang, J. Qi, R. Wu, J. Niu, W. Liu, You only look at one sequence: Rethinking transformer in vision through object detection, Adv. Neural Inf. Process. Syst. 34 (2021) 26183–26197.

[9] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, End-to-end object detection with transformers, in: Eur. Conf. Comput. Vis., Springer, 2020: pp. 213–229.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, Ssd: Single shot multibox detector, in: Eur. Conf. Comput. Vis., Springer, 2016: pp. 21–37.

[11] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: Common objects in context, in: Eur. Conf. Comput. Vis., Springer, 2014: pp. 740–755.

[12] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, S. Savarese, Generalized intersection over union: A metric and a loss for bounding box regression, in: Proc. IEEECVF Conf. Comput. Vis. Pattern Recognit., 2019: pp. 658–666.

[13] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, ArXiv Prepr. ArXiv171105101. (2017).