2025 IEEE International Conference on Image Processing (ICIP) | 979-8-3315-2379-4/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICIP55913.2025.11084680

ENACT: ENTROPY-BASED CLUSTERING OF ATTENTION INPUT FOR REDUCING THE COMPUTATIONAL NEEDS OF OBJECT DETECTION TRANSFORMERS

Giorgos Savathrakis[1], *Antonis Argyros*[1,2]

[1]Institute of Computer Science, FORTH [2]Computer Science Department, University of Crete

{gsav, argyros}@ics.forth.gr

ABSTRACT

Transformers demonstrate competitive performance in terms of precision on the problem of vision-based object detection. However, they require considerable computational resources due to the quadratic size of the attention weights. In this work, we propose to cluster the transformer input on the basis of its entropy, due to its similarity between same object pixels. This is expected to reduce GPU usage during training, while maintaining reasonable accuracy. This idea is realized with an implemented module that is called ENtropy-based Attention Clustering for detection Transformers (ENACT), which serves as a plug-in to any multi-head self-attention based transformer network. Experiments on the COCO object detection dataset and three detection transformers demonstrate that the requirements on memory are reduced, while the detection accuracy is degraded only slightly. The code of the ENACT module is available at https://github.com/GSavathrakis/ENACT.

Index Terms— Clustering, Detection, Entropy, Transformers

1. INTRODUCTION

Object detection is an important and challenging problem in computer vision, with a wide range of detectors, mostly Deep Learning based, having been proposed over the years. These consist of Convolutional Neural Network (CNN) backbones, and networks that aim to localize object regions and classes.

1.1. Transformer-based object detectors

The development of object detectors with a transformer-like architecture [1] resulted from the need to remove manually crafted components like NMS suppression or anchor generation, existing in previous detectors [2, 3]. Maintaining the CNN backbone from older architectures, the rest of the network follows an encoder-decoder architecture as proposed by Vaswani et al [4]. Both the encoder and decoder layers include multi-head self-attention (MHSA) modules that quantify the correlation between a set of Queries and Keys, and match the resulting weights to a set of corresponding Values. All three originate from the CNN output in the encoder layers. The decoder input comes from learnable object queries and from the encoder output. Computing the attention has quadratic space and time complexity $\mathcal{O}(N^2)$, which is a main limitation of this transformer architecture. Ideas aiming to solve this problem either restricted the number of pixels each pixel should focus on [5], or clustered pixels depending on the similarity of their features (e.g., [6, 7]).

1.2. Information-based clustering

Most of the works that implemented attention clustering, used the feature vectors' Euclidean distance as a metric, assuming a connection between dot product similarity and object class. However, this approach is slow and requires domain-driven hyperparameter tuning (e.g., number of clusters).

To overcome these problems, we propose the Shannon entropy of the Keys' pixels as the clustering criterion, because pixels with similar information will very likely belong to the same object. Furthermore, the sum of the pixels' self-information is the information of the sum, which helps at merging pixels, while optimally retaining their information.

Therefore, in this work, we propose the ENtropy-based Attention Clustering for detection Transformers (ENACT). ENACT clusters the Keys and Values by estimating the p.d.f of the Key input using a linear layer, computing the entropy using the p.d.f, and finally implementing grouping on the basis of the information signal's second derivative. The major contributions of our approach are the following:

- ENACT can serve as a *plug-in to any detection transformer* architecture with an MHSA module in the encoder, and is orthogonal to any other memory reducing method (e.g., smaller arithmetic precision). Additionally, the clustering is based on the Key pixels' self-information similarity. Therefore, *the number of clusters is not a hyper-parameter* and instead, its computation is data-driven.
- To the best of our knowledge, ENACT is the only method which handles variable-length tensors in the same batch, in a GPU parallelizable manner (see supplementary material for more details).

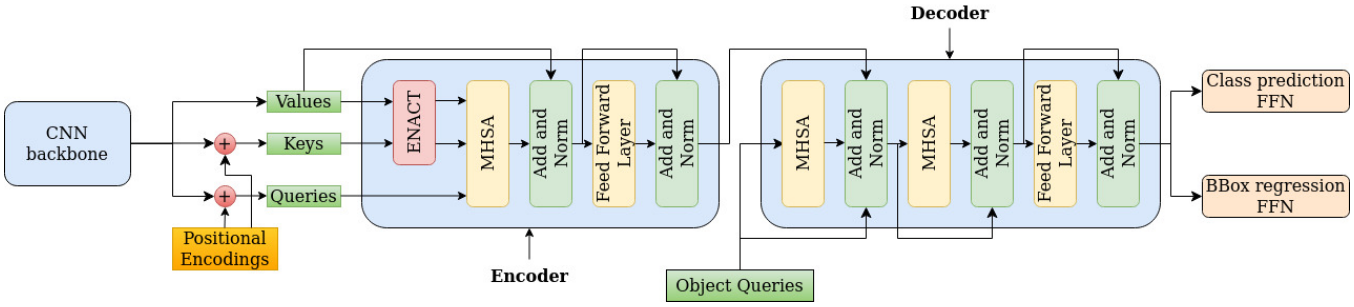

Fig. 1. Visualization of a detection transformer with the addition of the ENACT module. The feature maps obtained from the CNN backbone, are the Queries, Keys and Values, with the former two being supplemented with positional encodings. All three pass through the encoder, with the Keys and Values being clustered by the ENACT module. Subsequently, the Queries, as well as the clustered Keys and Values pass through the multi-head self-attention module (MHSA).

- We plug ENACT to three Detection Transformers [1, 8, 9] and we show that we obtain similar precision, while saving about 15-46% GPU RAM depending on the detector.

2. RELATED WORK

Transformer networks were initially developed to deal with sequence translation problems (i.e. text-to-text translation etc.). Vaswani et al [4] who introduced them used an encoder-decoder architecture where the input passes through a self-attention module in order to learn the connections between queries and keys. Dosovitskiy et al [10] developed the Vision Transformer (ViT), considering Queries and Keys being image patches. Transformers for object detection were initially proposed by Carion et al [1] who developed the DETR model, which implements a bipartite matching loss that unilaterally connects a prediction to a ground truth object, in addition to a transformer-based encoder-decoder structure. This removed the need for Region Proposal Networks and NMS suppression, which are components required by older detectors such as YOLO [2] and Faster R-CNN [3].

To solve the problem of the large space and time complexity Zhu et al proposed the Deformable DETR algorithm [5], where the attention of each pixel is focused on a predetermined number of points, with learnable attention weights and point locations. Other variants such as the Anchor DETR [9] and Conditional DETR [8] also propose methods that lead to reduced training times through modifications in the query design.

Improving transformers in terms of memory can also be done by clustering the input. Vyas et al [7] used k-means to group queries into clusters, whereas works done by Zeng et al [11] and Haurum et al [12], implemented hierarchical clustering, starting with many clusters and gradually reducing them to include the important regions. Zheng et al [6] proposed the Adaptive Clustering Transformer (ACT) for DETR, which clusters the queries into prototypes using locality sensitive hashing to group queries with small Euclidean distance.

3. METHOD

3.1. Overview

In this work we propose the ENACT module, which is a plug-in clustering component to the standard detection transformer architecture (see Figure 1), based on each pixel's self-information. The information computation is data-driven and therefore, there is *no pre-determination of the number of clusters*. Furthermore, it reduces the size of the input, which has a significant impact on the GPU memory required for training.

In detection transformers, the Queries (Q), Keys (K) and Values (V) are the same, with the exception that Queries and Keys are supplemented with positional encodings, that add information about the location of each pixel. The dimensions of Q, K and V, are $\mathrm{N} \times \mathrm{HW} \times \mathrm{d}$ where N is the batch size, HW the total number of pixels in the feature map, and d the dimensions of the feature vector. Thus, computing the attention weights (see equation 1), has quadratic complexity. By clustering the Keys and Values, the resulting $\mathrm{K_{cl}}, \mathrm{V_{cl}} \in \mathbb{R}^{\mathrm{H'W'} \times \mathrm{d}}$ where $\mathrm{H'W'} < \mathrm{NHW}$. This will reduce the size of the attention weights, and since the Values are also clustered, computing the final attention map (see equation 2) is also more memory efficient.

$$\mathrm{A} = \mathrm{softmax}\left(\frac{\mathrm{Q} \cdot \mathrm{K}^{\mathrm{T}}}{\sqrt{\mathrm{d}}}\right), \; \mathrm{A_{cl}} = \mathrm{softmax}\left(\frac{\mathrm{Q} \cdot \mathrm{K}_{\mathrm{cl}}^{\mathrm{T}}}{\sqrt{\mathrm{d}}}\right). \tag{1}$$

$$\mathcal{A} = \mathrm{A} \cdot \mathrm{V}, \quad \mathcal{A}_{\mathrm{cl}} = \mathrm{A_{cl}} \cdot \mathrm{V_{cl}}. \tag{2}$$

The Queries are left un-clustered because we want the final attention map to maintain its size after each encoder layer, in order to dimensionally match the positional encodings.

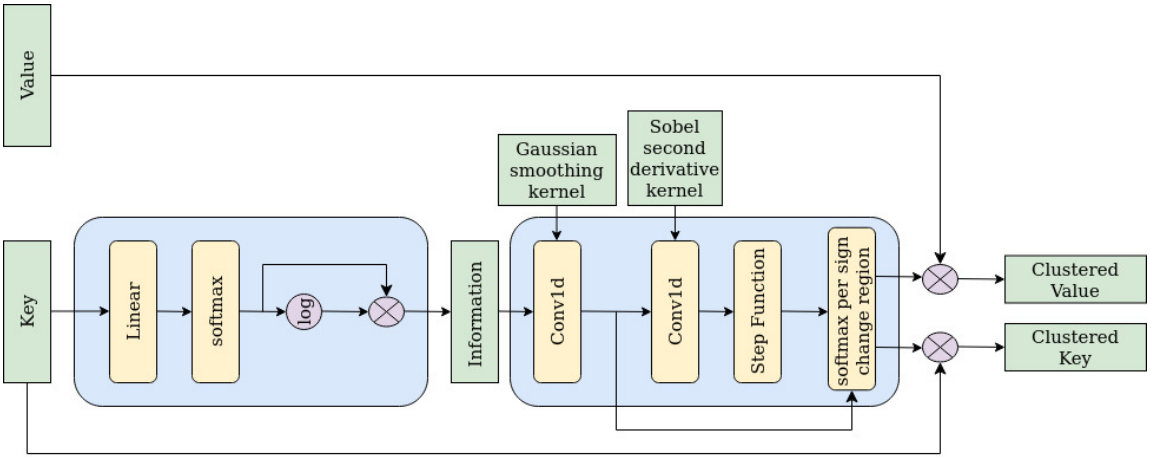

Fig. 2. Overview of the ENACT module. The key passes through the network responsible for the calculation of the information and subsequently the information is smoothed by a Gaussian kernel. Then, the regions where the information is a convex or concave function are identified by the second derivative, and a softmax function is applied to the information values of each separate region of same sign in second derivative. The clustering is done by summing the weighted information values multiplied by the respective feature vectors of the values and keys, per region.

The ENACT module (see Figure 2), consists of two main components. The first computes the self-information of the input pixels, and the second clusters regions of information gain and loss. To that end, we compute the entropy of the Keys because the positional encodings add locality information which influences the entropy. Also, it is plugged in the encoder only, since the decoder input is trainable and initialized as Gaussian noise. Therefore, clustering will damage training and the entropy of the noise will be meaningless.

3.2. Clustering process

The clustering is done as shown in Eq. 3. $\mathcal{H}(\mathrm{x})$ is the self-information (i.e. the quantity inside the Shannon entropy sum), x is the feature vector of a pixel, and $\mathrm{p}(\mathrm{x})$ is the learnable p.d.f. of the vector distribution. The p.d.f. is learned using a linear layer $\mathbb{R}^{\mathrm{d}\to 1}$ and a softmax function.

$$\mathcal{H}(\mathrm{x}) = -\mathrm{p}(\mathrm{x})\log(\mathrm{p}(\mathrm{x})). \tag{3}$$

Therefore, the final output of the information module is a signal with shape (batch size $\times$ HW) where HW is the total number of pixels in the feature map.

Subsequently, we cluster K and V using the computed entropy, and a 1D convolution with a Gaussian kernel to smoothen the signal. The standard deviation σ is a hyperparameter and the kernel ranges from -3σ to 3σ.

After the smoothing is completed, we compute the regions where the signal is concave and convex. To do so, we first estimate its second derivative using the Sobel kernel in Eq. 4.

$$\mathrm{Sobel}''(\mathrm{x}) = [-1, 2, -1]. \tag{4}$$

The reason we do this is because we consider the regions where the local information gain or loss is apparent as the most suitable for clustering, since, for example, a pixel with an information local maximum/minimum and its surroundings could be clustered as one entity.

Subsequently, we pass the second derivative of the information to a step function which attributes 1 to the convex parts of the information curve, and -1 to the concave ones. Concave regions signify information gain, and convex ones signify loss.

At that point, we take the smoothed entropy and we run a softmax function on each separate region whose indices correspond to regions of the same sign in the output of the step function. The final step is to multiply the resulting weighted entropies with the respective feature vectors from the Keys and Values, and sum the results per cluster. To achieve this goal, we implement methods that take advantage of GPU parallelization by creating CUDA kernels that make the necessary operations in an optimal manner. Further information is provided in the supplementary material[1].

4. EXPERIMENTS

4.1. Dataset and Settings

We evaluate our model on the MS COCO 2017 dataset, using the train2017 subset for training, and the val2017 for validation.

As it was mentioned in Section 3.1, the Keys and Values pass through the ENACT module for dimensional consistency

[1] https://sigport.org/sites/default/files/docs/ICIP_2025_Supplementary_Material.pdf

Model	GPU RAM (GB)	GPU Gain (%)	training time (s/image)	Inference time (s/image)
DETR [1]	36.5	-	**0.0541**	**0.0482**
DETR + ACT [6]	28.9	20.8	0.0634	0.0494
DAB-DETR [13]	26.1	28.5	0.0965	0.0590
MS-DETR [14]	21.0	42.5	0.1557	0.1088
DETR + ENACT	**19.7**	**46.0**	0.0693	0.0538
Anchor DETR [9]	29.7	-	**0.0999**	**0.0707**
Anchor DETR + ENACT	**25.1**	15.5	0.1170	0.0779
Conditional DETR [8]	46.6	-	**0.0826**	**0.0646**
Conditional DETR + ENACT	**37.3**	20.0	0.0930	0.0693

Table 1. Comparison of GPU RAM consumption in GB, training and inference time in seconds per image, for three detection transformers, with and without the plugging of the ENACT module. We also show the GPU memory percentage gain when using ENACT. For DETR we also provide the comparisons with the ACT clustering module.

Model	Epochs	AP	AP_{50}	AP_{75}	AP_s	AP_m	AP_l
RetinaNet [15]	36	38.7	58.0	41.5	23.3	42.3	50.3
Faster RCNN [3]	36	40.2	61.0	43.8	24.2	43.5	52.0
RSDNet† [16]	12	40.3	60.1	43.0	22.1	43.5	51.5
Cascade R-CNN† [17]	12	42.7	61.6	46.6	23.8	46.2	57.4
DETR-C5 [1]	300	40.6	61.6	-	19.9	44.3	60.2
DETR-C5 + ENACT (ours)	300	40.1	60.5	41.9	19.4	43.5	58.2
Anchor DETR-DC5 [9]	50	44.3	64.9	47.7	25.1	48.1	61.1
Anchor DETR-DC5 + ENACT (ours)	50	42.9	63.4	46.2	24.6	46.9	58.5
Conditional DETR-C5† [8]	50	42.8	63.7	46.0	21.7	46.6	60.9
Conditional DETR-C5 + ENACT† (ours)	50	41.5	62.6	44.1	21.9	45.3	59.2

Table 2. Detection performance results in terms of precision, as well as number of training epochs, for different object detectors, on the COCO val2017. DC5 and C5 denote whether the last convolutional layer in the DETR based models is dilated or not, respectively. All models share the ResNet-50 backbone, unless denoted by † which means they were trained using ResNet-101 backbone. (AP_{75} not available in DETR 300 epoch schedule).

reasons, and the entropy is computed only from the Keys. The parameters are initialized from a Xavier uniform distribution and the biases are initially zero. We plug ENACT to three Detection Transformers, which are the DETR [1], Conditional DETR [8] and Anchor DETR [9], and we use the ResNet 50 and 101 backbones [18]. DETR is trained for 300 epochs, dropping the learning rate at 200, and the other two for 50 epochs with the drop at 40. The batch size is set to 4 for the Anchor-DETR and 8 for the other two transformers. For our experiments, we used an NVIDIA RTX A6000 GPU with 49GB RAM.

The standard deviation σ of the Gaussian smoothing kernel, is set to 5 for DETR and Conditional DETR, and to 3 for Anchor DETR. To evaluate the proposed ENACT module, we report GPU memory usage, training and inference time, and average precision (AP), as well as average precision at different IoU thresholds and different object area sizes.

4.2. Ablation studies

The only tunable hyperparameter is the standard deviation of the Gaussian smoothing kernel. It is determined by the AP obtained after a certain number of epochs for each detector. For DETR, after 10 epochs of training, we get an AP of 12.8, 12.9 and 12.7 for $\sigma = 3, 5, 7$ respectively. For Anchor-DETR we get 25.5 and 25.1 AP at epoch 5 for $\sigma = 3, 5$ respectively, and finally for Conditional-DETR at epoch 5 we get AP 18.6 and 19.4 for $\sigma = 3, 5$ respectively. Therefore, σ is set to 5, 3 and 5 for the respective detectors.

4.3. Results

Quantitative results: We present the overall performance of the ENACT module, when plugged into the three mentioned detection transformers. Starting with the GPU requirements, in Table 1 we show the RAM consumption of the GPU when using the three detection transformer variants, and when plugging in the ENACT module to each of them. In the case of DETR, we also provide results for DAB-DETR [13], MS-DETR [14] and the ACT module [6]. They are all algorithms that tweak the transformer input, and we present them in order to compare the respective gains in memory and time. We can observe that by supplementing each of the detection transformers with the ENACT module, we consistently reduce the

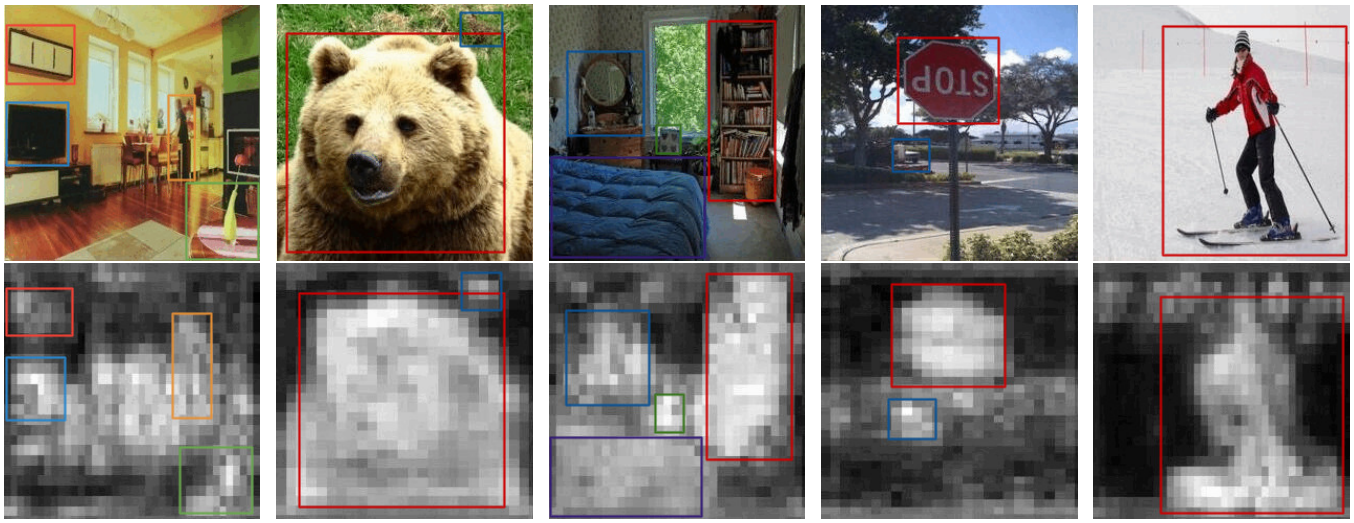

Fig. 3. A selection of images from the COCO dataset (above), with the respective output of the self-information module prior to the Gaussian smoothing (below). Bright pixels in the bottom images, correspond to regions that yield higher information. We use same-color bounding boxes, to show corresponding objects between images and information maps.

GPU consumption by a margin which ranges from 15% to 46%. In addition, compared to ACT, DAB-DETR and MS-DETR, we surpass them with respect to GPU gain by 26%, 17.5% and 3.5% respectively. We note that the GPU memory values for DAB-DETR and MS-DETR are conservative estimates, obtained by adjusting the values claimed by the respective authors, to our batch size. Actual experiments using these two detectors, may yield GPU memory values that are larger than the ones presented in this paper.

Finally, we evaluate ENACT's performance in terms of precision and present the results in Table 2. We observe that the obtained APs are very close to those of the original transformer models, with the reductions being 0.5% for the DETR, 1.3% for the Conditional DETR and 1.4% for the Anchor DETR. Those are fairly slight drops considering the GPU gains, and the resulting precisions still surpass several existing object detectors in the same dataset.

The only cost comes in terms of training and inference time, where the delay ranges between 11 % to 22% in training time, and between 6 % to 10% in inference time.

Qualitative results: Subsequently, we verify the extent to which the idea of computing the self-information of each pixel is valid. Besides verifying that ENACT drops the needed computational resources without affecting considerably the detection precision, it is also important to visualize the output from ENACT's self-information component. Some indicative results are shown in Figure 3. Specifically, we plug the ENACT module in the Conditional DETR, and we pass the images through the trained model of the final epoch. We observe that for the most part, the visualized information map is brighter in regions where objects are located, and dimmer in the background. Considering the fact that the probability density function is learnable, we can see that the final trained model, in the part responsible for computing the information, correctly learned that the most informative pixels are the ones that correspond to objects of interest.

5. SUMMARY

We presented ENACT, a clustering module for MHSA-based detection transformers, which compresses the encoder input on the basis of its entropy. We showed that the pixel-wise self-information is indeed correlated to the important objects in the image, thereby rendering it as a valid metric for clustering. We trained three detection transformers using ENACT, and we show that we can obtain approximately 15% to 45% decrease in GPU memory, while losing only 0.5% to 1.4% in average precision. This makes ENACT a reliable plug-in for this type of transformer models, which can reduce significantly their computational requirements without compromising considerably their object detection capabilities.

6. ACKNOWLEDGMENTS

This work was co-funded by the European Union (EU - HE Magician – Grant Agreement 101120731). The authors also gratefully acknowledge the support of the framework of the National Recovery and Resilience Plan Greece 2.0, funded by the European Union– NextGenerationEU (project Greece4.0).

7. REFERENCES

[1] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," in *ECCV*. 2020, pp. 213–229, Springer International Publishing. 1, 2, 4

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA, jun 2016, pp. 779–788, IEEE Computer Society. 1, 2

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds. 2015, vol. 28, Curran Associates, Inc. 1, 2, 4

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention Is All You Need," *Advances in Neural Information Processing Systems*, vol. 30, 2017. 1, 2

[5] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai, "Deformable DETR: Deformable Transformers for End-to-End Object Detection," in *International Conference on Learning Representations*, 2021. 1, 2

[6] Minghang Zheng, Peng Gao, Renrui Zhang, Kunchang Li, Hongsheng Li, and Hao Dong, "End-to-End Object Detection with Adaptive Clustering Transformer," in *32nd British Machine Vision Conference 2021, BMVC 2021, Online, November 22-25, 2021*. 2021, p. 226, BMVA Press. 1, 2, 4

[7] Apoorv Vyas, Angelos Katharopoulos, and François Fleuret, "Fast Transformers with Clustered Attention," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21665–21674, 2020. 1, 2

[8] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang, "Conditional DETR for Fast Training Convergence," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 3651–3660. 2, 4

[9] Yingming Wang, Xiangyu Zhang, Tong Yang, and Jian Sun, "Anchor DETR: Query Design for Transformer-Based Detector," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022, vol. 36, pp. 2567–2575. 2, 4

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *International Conference on Learning Representations*, 2021. 2

[11] Wang Zeng, Sheng Jin, Wentao Liu, Chen Qian, Ping Luo, Wanli Ouyang, and Xiaogang Wang, "Not All Tokens are Equal: Human-Centric Visual Analysis via Token Clustering Transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11101–11111. 2

[12] Joakim Bruslund Haurum, Sergio Escalera, Graham W Taylor, and Thomas B Moeslund, "Agglomerative Token Clustering," in *European Conference on Computer Vision*. Springer, 2025, pp. 200–218. 2

[13] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang, "DAB-DETR: Dynamic Anchor Boxes are Better Queries for DETR," in *International Conference on Learning Representations*, 2022. 4

[14] Chuyang Zhao, Yifan Sun, Wenhao Wang, Qiang Chen, Errui Ding, Yi Yang, and Jingdong Wang, "MS-DETR: Efficient DETR Training with Mixed Supervision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 17027–17036. 4

[15] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, "Focal Loss for Dense Object Detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017. 4

[16] Shaoru Wang, Yongchao Gong, Junliang Xing, Lichao Huang, Chang Huang, and Weiming Hu, "RDSNet: A New Deep Architecture for Reciprocal Object Detection and Instance Segmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, pp. 12208–12215. 4

[17] Zhaowei Cai and Nuno Vasconcelos, "Cascade R-CNN: Delving into High Quality Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6154–6162. 4

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. 4