On the Detection of Powerline Elements with Efficient Transformers

Emmaouil Patsiouras
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
emmanoup@csd.auth.gr

Vasileios Mygdalis
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
mygdalisv@csd.auth.gr

Ioannis Pitas
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
pitas@csd.auth.gr

***Abstract*—Powerline inspection operations involve capturing and inspecting visual (RBG/thermal/LiDAR) footage of powerline elements from elevated positions above and around the powerline, which are currently performed with the help of helicopters and/or Unmanned Aerial Vehicles (UAV). Current technological advances in the areas of robotics and machine learning are towards enabling fully autonomous operations. To this end, one of the tasks to be addressed is the robust, precise and fast powerline object detection problem. End-to-end Object Detection with Transformers (DETR) was a recently introduced method that demonstrates time and accuracy advances with respect to other detectors. However, this architecture comes with some computational complexity issues, which can mainly be attributed to Transformer encoder/decoder components, limiting its applicability in fast processing high-resolution feature maps and long sequences in general. To address these issues, we propose incorporating low-complexity Transformer implementations and evaluate them in a recently captured powerline detection dataset.**

I. Introduction

The task of object detection consists of classifying and localizing every object of interest depicted in an image frame, in the form of rectangular Regions of Interest (ROI). We focus on the application scenario of linear infrastructure inspection, which requires the precise localization of powerline elements and components on the power transmission lines (i.e., insulators, dumpers and towers). The most common way of representing these objects is through predicting a set of bounding boxes associated with category labels. Most of the popular Deep Learning (DL)-based object detectors [1] treat this task as a combination of classification and bounding box regression and usually employ many trivial hand-crafted components such as region proposal or anchor generation and non-maximum suppression post-processing. Detection with Transformers (DETR) [2] is a recently introduced state-of-the-art object detector that totally reshaped the way object detection is viewed. DETR addresses the object detection as a direct set prediction problem and completely eliminates the need of any geometric priors resulting in the first fully differentiable end-to-end object detector.

One major novelty introduced in DETR was the incorporation of Transformer [3] encoder-decoder modules in conjunction with a convolutional neural network (CNN) model, in an object detection pipeline. However, DETR, despite its fascinating design and formidable performance suffers from two important issues: (i) In order for its model to converge it requires a much longer training schedule than many of the existing object detectors. As the authors claim, in order for DETR to converge in the COCO [4] dataset it required over 300-500 epochs using the most advanced processing units. (ii) DETR models perform rather poorly in detecting small objects, such as in our experimental application scenario. The latter issue could easily be addressed by utilizing higher resolution feature maps but simultaneously leading to unacceptable complexities, as far time and memory are concerned. The aforementioned bottlenecks can mainly be attributed to the limitation of Transformer components in fast processing large image feature maps and sequences in general.

In this paper, we propose the incorporation of an efficient Transformer architecture, based on the Linformer [5], and we benchmark the overall detection pipeline in our composed dataset of powerline elements. By examining this straightforward but non-trivial incorporation, we answer the question of what do we gain and what do we miss by minimizing the computational complexity of the Transformer architecture, in the powerline element detection task. Our experimental evaluation shows that efficient transformer implementations will eventually allow the implementation of Trasformer-based architectures in embedded computing devices, in the near future.

The remainder of this paper is structured as follows. Section 2 revisits the attention mechanism introduced in Transformers and presents how DETR utilizes a Transformer encoder-decoder module. In Section 3 we analyse why Transformers constitute a computational expensive module and we describe an efficient DETR-based detection framework that compromises between accuracy and speed. In Section 4 we present and discuss the experiments conducted on our powerline element dataset. Finally, Section 5 concludes our findings and summarizes for future research and improvements.

II. Revisiting attention and DETR

In this section we review the attention mechanism encountered in Transformers and describe how DETR utilizes this mechanism in a Transformer encoder-decoder module.

A. Attention

Transformers, originally proposed for natural language processing [6] tasks, have made their way across a range of different domains such as speech recognition [7] and image

processing [8], [9], [10]. In general, Transformers constitute an attention mechanism that receive two sequences of elements as inputs (e.g. texts or image feature maps), and updates the elements of one of them by aggregating information from every element of the other.

Let $\mathbf{X} \in \mathbb{R}^{N \times n}$ and $\mathbf{Y} \in \mathbb{R}^{M \times n}$ be two input matrices consisting of N and M elements respectively of n dimension each. Based on $\mathbf{X}, \mathbf{Y}$ the following three matrices can be created: i) a query matrix $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q + \mathbf{1}_{N\times 1}\mathbf{b}_Q$, ii) a key matrix $\mathbf{K} = \mathbf{Y}\mathbf{W}_K + \mathbf{1}_{M\times 1}\mathbf{b}_K$, and iii) a value matrix $\mathbf{V} = \mathbf{Y}\mathbf{W}_V + \mathbf{1}_{M\times 1}\mathbf{b}_V$. $\mathbf{W}_Q \in \mathbb{R}^{n\times n_q}, \mathbf{W}_K \in \mathbb{R}^{n\times n_k}$ and $\mathbf{W}_V \in \mathbb{R}^{n\times n_v}$ are linear transformation matrices applied on the input matrices and $\mathbf{b}_Q \in \mathbb{R}^{n_q}, \mathbf{b}_K \in \mathbb{R}^{n_k}, \mathbf{b}_V \in \mathbb{R}^{n_v}_v$ are their respective biases. For simplicity we can consider $n_q = n_k = n_v = n$. Based on the above, the operation of attention as defined in [2] is expressed as:

$$\mathbf{A} = \overbrace{\sigma(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{n}})}^{\mathbf{S}}\mathbf{V}, \tag{1}$$

where $\sigma(\cdot)$ is the softmax operator applied on every row of the matrix $\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{n}}$ and $\mathbf{S} = \sigma(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{n}})$. This type of attention is also called cross-attention. Self-attention is defined when $\mathbf{Y}$ coincides with $\mathbf{X}$ (i.e. $\mathbf{Y} = \mathbf{X}$) or vice versa. Self-attention is the basic building block of a Transformer model and allows it to associate each element in an input sequence to every other element in the same sequence.

The above definition describes a naive attention computation. In practice, however, Transformers usually leverage a multi-headed attention mechanism. In this case we have N_h number of attention heads and we split the $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$ matrices into N_h matrices of dimensions $n \times \frac{n}{N_h}$ (n should be divisible by N_h). The attention for every single head, $\mathbf{A}_h, h = 1, \ldots, N_h$ is defined as:

$$\mathbf{A}_h = \overbrace{\sigma(\frac{\mathbf{Q}_h\mathbf{K}_h^T}{\sqrt{n}})}^{\mathbf{S}_h}\mathbf{V}_h, \tag{2}$$

where $\mathbf{Q}_h = \mathbf{X}\mathbf{W}_{Q_h}, \mathbf{K}_h = \mathbf{Y}\mathbf{W}_{K_h}$, $\mathbf{V}_h = \mathbf{Y}\mathbf{W}_{V_h}$ (biases are omitted for simplicity) for $h = 1, \ldots, N_h$, and $\mathbf{S}_h = \sigma(\frac{\mathbf{Q}_h\mathbf{K}_h^T}{\sqrt{n}})$. The overall attention is the concatenation of the attention of every single head and is defined as:

$$\mathbf{A} = (\mathbf{A}_1 \oplus \mathbf{A}_2 \oplus \cdots \oplus \mathbf{A}_h)\mathbf{W}_O, \tag{3}$$

where $\oplus$ represents the concatenation of two 2D matrices along the horizontal dimension and $\mathbf{W}_O \in \mathbb{R}^{n\times n_o}$ is a linear projection matrix. Again we can consider $n_o = n$. In the following section we will discuss how the multi-head attention mechanism is employed in the DETR framework.

B. DETR architecture

DETR is mainly composed of two components: a CNN backbone for extracting image feature representations from input images and an encoder-decoder Transformer. We review the network's architecture and function in detail as follows:

Backbone: It is a typical CNN backbone which receives an image $\mathbf{x} \in \mathbb{R}^{H_0\times W_0\times 3}$, where H_0, W_0 are the height and width of the input image, and produces lower-resolution feature maps $\mathbf{f} \in \mathbb{R}^{H\times W\times C}$, where C is the number of output channels. Following the main backbone the output $\mathbf{f}$ passes through a convolutional layer with 1×1 kernels which reduces the feature maps from C to a smaller dimension d, thus creating a new feature map $\mathbf{z} \in \mathbb{R}^{H\times W\times d}$. Typical values used are $C = 2048, H = \frac{H_0}{32}, W = \frac{W_0}{32}$ and $d = 256$.

Transformer encoder: The encoder expects a 2-dimensio nal tensor as input so the spatial dimensions of the previously produced 3-dimensional tensor $\mathbf{z}$ are unrolled resulting in a new feature map $\mathbf{z'} \in \mathbb{R}^{HW\times d}$. The new tensor is then additively combined with a positional encoding matrix $\mathbf{p} \in \mathbb{R}^{HW\times d}$ which holds information regarding the position of every pixel in the feature map $\mathbf{z'}$, resulting in a new feature map $\mathbf{z'_p} \in \mathbb{R}^{HW\times d}$. This feature map is then fed to a standard Transformer encoder layer. An encoder layer contains two sub-modules, a multi-head self-attention layer followed by a fully connected feed forward network (FFN). Given the query, key and value matrices $\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h$, which in the case of DETR are $\mathbf{Q}_h = \mathbf{K}_h = \mathbf{z}'_p\mathbf{W}_{Q_h}$ and $\mathbf{V}_h = \mathbf{z'}\mathbf{W}_{V_h}$, and N_h attention heads, the attention matrix is calculated as described by equations (2) and (3). At this point, the input and output of the multi-head attention module are connected by residual connections and a layer normalization layer [11]. The output of the normalization layer is then passed through a FFN with two linear layers which has its input/ouput similarly connected in a residual fashion with a normalization layer producing the final output. The encoder could consist of several such layers stack on top of each other.

Transformer decoder: The decoder is of similar structure as the encoder but consists of two multi-head attention layers. The first multi-head self-attention layer receives N_q in number $n-$dimensional object queries represented by learnable, and initially random initialized, query embeddings $\mathbf{o} \in \mathbb{R}^{N_q\times n}$. After passing trough a normalization layer the output of the first self-attention layer is then fed to a second cross-attention layer that also receives input from the final layer of the encoder combined with positional encodings. Finally, the output of the second multi-head cross-attention is fed to normalization layers and a FFN with similar residual connections as described in the encoder. Decoder layers can also be stacked on top of each other.

Finally, DETR passes the output of the decoder to a FFN that predicts either a detection (class and bounding box) or a no object class.

III. Detection with efficient transformers

As demonstrated by the analysis in Section 2, DETR bottlenecks include its high computational and memory complexity derived from the self-attention heads in the encoder. It is apparent that the memory and time complexity required to compute the attention matrix, $\mathbf{A}_h$, of each attention head is quadratic w.r.t the length of sequence N or M (from now on reffered to as just N). In particular, computation of the similarity matrix $\mathbf{S}_h = \sigma(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{n}})$ requires multiplying two $N \times \frac{n}{N_h}$ matrices leading to an overall complexity of $\mathcal{O}(N^2)$. In the case of DETR, where N corresponds to HW, the above

complexity heavily affects the inference response especially when high-resolution feature maps are produced from the CNN backbone. Hence, a mechanism that could reduce the overall complexity would be of great importance, since it would not only allow for speeding up inference time, but simultaneously, allow the utilization of higher-resolution feature maps, thus increasing the details of small depicted objects.

Recently, there has been quite a surge of proposed efficient Transformer variants [12], [13], [14], [15], [16]. Efficient models are of crucial importance in applications that model long sequences such as in the computer vision and image processing domains. The efficiency of such models could refer to either the memory footprint of the model or to its computational costs, e.g. number of FLOPS. To this end, the primary goal of most of the above mentioned efficient models is to propose a way to approximate the quadratic cost of the similarity matrix $\mathbf{S}_h$. One of the emerging techniques towards improving the efficiency is leveraging low-rank approximations of the similarity matrix $\mathbf{S}_h$ [5]. The main idea behind such approaches is to assume low-rank structure in the $N \times N$ matrix. Another popular method is to view the attention mechanism through kernelization [13], [14]. The usage of kernels enables intelligent mathematical ways of reforming the self-attention mechanism in order to avoid explicitly computing the $N \times N$ matrix $\mathbf{S}_h$.

At this point we introduce the incorporation of an efficient Transformer model, namely the Linformer [5], that demonstrates the reduction of complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. In [5], authors suggest an approximate way of calculating the self-attention that has linear memory and time complexity in terms of the sequence length. More specifically, authors, show both theoretically and empirically that the similarity matrix, $\mathbf{S}_h$, is low rank and therefore can be constructed by an approximate low rank matrix $\overline{\mathbf{S}}_h$. The main idea of their proposed linear self-attention is to add two linear projection matrices $\mathbf{E}_h, \mathbf{F}_h \in \mathbb{R}^{K \times N}$ that serve to reduce the dimension of key and value matrices $\mathbf{K}_h, \mathbf{V}_h$ from N to a lower dimension K. The new attention matrix is defined as [5]:

$$\overline{\mathbf{A}}_h = \sigma(\frac{\overbrace{\mathbf{Q}_h\overbrace{(\mathbf{E}_h\mathbf{K}_h)^T}^{\overline{\mathbf{K}}_h^T}}^{\overline{\mathbf{S}}_h}}{\sqrt{n}})\overbrace{(\mathbf{F}_h\mathbf{V}_h)}^{\overline{\mathbf{V}}_h}, \tag{4}$$

where $\overline{\mathbf{K}}_h^T = (\mathbf{E}_h\mathbf{K}_h)^T, \overline{\mathbf{V}}_h = \mathbf{F}_h\mathbf{V}_h$, and $\overline{\mathbf{S}}_h = \sigma(\frac{\mathbf{E}_h\overline{\mathbf{K}}_h^T}{\sqrt{n}})$.

The above computation requires only $\mathcal{O}(N)$ time and memory complexity because the $N \times N$ matrix $\mathbf{S}_h$ is now decomposed to the $N \times K$ matrix $\overline{\mathbf{S}}_h$. Hence, if a very small projected dimension K is chosen, such that $K \ll N$, the time and memory consumption is significantly reduced. As an additional optimization technique for our experiments, we use layer-wise sharing such that a single projection matrix $\mathbf{E}_h$ is used, i.e. $\mathbf{E}_h = \mathbf{F}_h$, for both key and value matrices, for all self-attention heads in an encoder layer and across all the encoder layers.

At this point, we should also point out that computational and memory complexity are of utmost importance in UAVs, which incorporate embedded computers with limited capacity. Fast performing methods in powerline inspection tasks are important not only for being applied in embedded devices, but will also allow higher input image resolutions to be processed. This is tightly coupled with the successful detection of powerline elements from longer distances, thus enabling the coupling with autonomous tracking [17]/UAV control methods [18].

IV. Experimental study

A. Dataset

As mentioned in the introduction Section, our application scenario for testing our detection models is that of detecting elements and components most commonly found in high-voltage transmission lines for inspection purposes. However, this detection task comes with quite a few challenges. Maybe the most prominent challenge is the requirement to simultaneously detect small and big electrical components that appear from short and long distances, diverse viewing angles and with various illumination conditions. Further challenges could be imposed when the objects of interest are clutched against an indistinguishable background. Moreover, in order to accurately detect the desired elements any DL-based detector would require a large amount of training data. To the best of our knowledge there are no publicly available datasets big enough for satisfactory results. Individual researcher efforts so far, such as the CPLID datasets [19] only include a few hundred images and insufficient amount of classes.

We perform experiments on our own composed powerline inspection dataset. The objects of interest encountered in this dataset are insulators, dumpers and electric towers. Our dataset consists of 11587 images, acquired from video footage of aerial inspection, 8422 of which were used for training and the remaining 3165 for evaluation. The images where then annotated with ROIs of the aforementioned object classes. Examples of the dataset along with visual results of the experimented detection models, discussed in the following section, are presented in Fig. 1.

B. Experiments and results

At this point, we present the experimental setup and we discuss the obtained results. DETR-R50 was chosen as our baseline model. It uses Resnet50 [20] as backbone for image feature extraction and 6 encoder/decoder layers with 8 attention heads each. We then replace the initial quadratic attention head with the linear one while keeping all of the other parts unchanged. For all of our DETR-based models a COCO pre-trained model was used for capturing general image features, which was then fine-tuned on our powerline inspection dataset in order to capture task specific patterns. During fine-tuning we discarded the initial random resize transforms and replaced them with a fixed resize that maintains the aspect ratio of our images. Specifically, our experiments were conducted for two different image input sizes, i.e. 256×448 and 448×796. Feeding images of the aforementioned sizes to the CNN backbone resulted in $\frac{H}{32} \times \frac{W}{32}$ downscale leading to feature maps of size $8 \times 14 (= 112)$ and $14 \times 24 (= 336)$, respectively. For the linear self-attention models (Lin-DETR entries in Table 1) the corresponding values of K were set to 64 and 128 respectively.

Fig. 1. Examples of our powerline inspection dataset and elements detected by DETR.

Lastly, the models were trained using AdamW [21] optimizer with a learning rate of 10^{-5}, weight decay of 10^{-2} and dropout set to 10^{-1}. The comprehensive results of our experimental study for different number of input image's size are presented in Table 1.

TABLE I. COMPARISON OF SSD AND DETR-BASED MODELS WITH A RESNET50 BACKBONE. THE MIDDLE AND BOT SECTION SHOWS RESULTS FOR DETR MODELS WITH QUADRATIC AND LINEAR ATTENTION, NOTED AS DETR AND LIN-DETR RESPECTIVELY. TOP SECTION SHOWS THE RESULTS FOR SSD MODELS. FPS WERE MEASURED IN AN NVIDIA 2080 SUPER GPU.

Model	FPS	Input Size	AP	AP_{50}	AP_{75}
SSD R50	40	256×448	48.3	73.6	55.5
SSD R50	31	448×796	52.3	79.8	54.7
DETR R50	35	256×448	52.4	83.1	55.3
Lin-DETR R50	37	256×448	50.7	81.2	51.8
DETR R50	27	448×796	56.3	86.0	61.3
Lin-DETR R50	30	448×796	53.2	83.1	55.5

For performance metrics we use average precision (AP) and frames per second (FPS). As can been noticed, by incorporating the linear attention mechanism we obtained some noticeable increase in FPS with some moderate dropout in AP compared to the baseline models and for fixed input size. These results leads us to the assumption that increasing the input size of an image combined with an efficient attention mechanism could result in even greater performances in the speed section while maintaining an acceptable detection accuracy. For comparison purposes, all of DETR-based models are compared to a single-stage detector, namely the Single Shot multibox Detector (SSD) [22]. We observe, that except for the case when low input size is used where SSD dominates in FPS, in all of the other cases DETR models, regardless of quadratic or linear attention, are superior in detection accuracy while keeping in par in speed.

As an additional experiment, to enhance our claim that exploiting larger feature maps and an a linear-attention mechanism does indeed offer incremental gains we measure the average times it took our models to separately process a single image in the backbone and in the Transformer components only. Those results are presented in Table 2. As we can see, for a lower input resolution, i.e. 256×448, by integrating the linear-attention we notice a 31% gain in the Transformer process time. Moreover, by increasing the input size to 448×796 we obtain a 34% gain.

All of the aforementioned results lead us to conclude that leveraging higher image resolutions along with a more lightweight feature extraction architecture, i.e. Mobilenets, could potentially lead into a faster DETR-based detection pipeline with only some moderate dropout in detection accuracy.

TABLE II. COMPARISON OF THE AVERAGE PROCESSING TIMES IN THE BACKBONE AND THE TRANSFORMER MODULES OF DETR AND LIN-DETR RESPECTIVELY. THE RECORDED TIMES ARE MEASURED IN SECODS.

Model	Input Size	Backbone Time	Transformer Time
DETR	256×448	0.008	0.013
Lin-DETR	256×448	0.008	0.009
DETR	448×796	0.016	0.015
Lin-DETR	448×796	0.016	0.010

V. CONCLUSION

In this paper we studied the DETR object detection method and how it utilizes a Transformer architecture in an overall detection pipeline. Moreover, we discussed the bottlenecks of Transformers and their limitation in fast processing large image feature maps in computer vision applications. We benchmark on an powerline inspection dataset and we demonstrated that the incorporation of an efficient Transformer model with linear attention can provide an acceptable trade-off between detection accuracy and inference speed. Our results showed that exploiting higher resolution images, in conjunction with an efficient Transformer model, could help improve the aforementioned trade-off.

ACKNOWLEDGMENT

This work has received funding from the European Unions Seventh Horizon 2020 research and innovation programme under grant agreement number 871479 (AERIAL-CORE). We would like to thank EDISTRIBUCIN Redes Digitales, S.L. for providing the aerial video footage. This publication reflects the authors views only. The European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *International journal of computer vision*, vol. 128, no. 2, pp. 261–318, 2020.

[2] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," *arXiv preprint arXiv:2005.12872*, 2020.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems (NIPS)*, 2017, pp. 5998–6008.

[4] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision (ECCV)*. Springer, 2014, pp. 740–755.

[5] S. Wang, B. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.

[6] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[7] N. Moritz, T. Hori, and J. Le, "Streaming automatic speech recognition with the transformer model," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6074–6078.

[8] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 3146–3154.

[9] L. Dosovitskiy, Al.and Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, U. Jakob, and H. Neil, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[10] B. Wu, C. Xu, X. Dai, A. Wan, P. Zhang, M. Tomizuka, K. Keutzer, and P. Vajda, "Visual transformers: Token-based image representation and processing for computer vision," *arXiv preprint arXiv:2006.03677*, 2020.

[11] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization. arxiv 2016," *arXiv preprint arXiv:1607.06450*, 2020.

[12] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *arXiv preprint arXiv:2009.06732*, 2020.

[13] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, D. Belanger, L. Colwell, and A. Weller, "Rethinking attention with performers," *arXiv preprint arXiv:2009.14794*, 2020.

[14] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *International Conference on Machine Learning (ICML)*. PMLR, 2020, pp. 5156–5165.

[15] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," *arXiv preprint arXiv:2001.04451*, 2020.

[16] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[17] I. Karakostas, V. Mygdalis, A. Tefas, and I. Pitas, "Occlusion detection and drift-avoidance framework for 2d visual object tracking," *Signal Processing: Image Communication*, vol. 90, p. 116011, 2021.

[18] N. Passalis and A. Tefas, "Deep reinforcement learning for controlling frontal person close-up shooting," *Neurocomputing*, vol. 335, pp. 37–47, 2019.

[19] X. Tao, D. Zhang, Z. Wang, X. Liu, H. Zhang, and D. Xu, "Detection of power line insulator defects using aerial images analyzed with convolutional neural networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 770–778.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision (ECCV)*. Springer, 2016, pp. 21–37.