A survey on: Application of Transformer in Computer Vision

Zhenghua Zhang^a, Zhangjie Gong^a, Qingqing Hong^{a,*}

^a Yangzhou University, 88 South University Road, Yangzhou City, Jiangsu Province, China

*Corresponding Author: 845114682@qq.com

Abstract

In the past few years, convolutional neural networks have been considered the mainstream network for processing images. Transformer first proposed a brand new deep neural network in 2017, based mainly on the self-attention mechanism, and has achieved amazing results in the field of natural language processing. Compared with traditional convolutional networks and recurrent networks, the model is superior in quality, has stronger parallelism, and requires less training time. Because of these powerful advantages, more and more related workers are expanding how Transformer is applied to computer vision. This article aims to provide a comprehensive overview of the application of Transformer in computer vision. We first introduce the self-attention mechanism, because it is an important component of Transformer, namely single-headed attention mechanism, multi-headed attention mechanism, position coding, etc. And introduces the reformer model after the transformer is improved. We then introduced some applications of Transformer in computer vision, image classification, object detection, and image processing. At the end of this article, we studied the future research direction and development of Transformer in computer vision, hoping that this article can arouse further interest in Transformer.

Keywords: Transformer, Self-attention, Reformer.

1. Introduction

Transformer is a new model architecture proposed in a 2017 paper "Attention is All You Need"⁽¹⁾. In this paper, experiments were conducted only on the scenario of machine translation, which completely defeated the SOTA model at the time. On WMT 2014 English-to-German translation task achieved 28.4 BLEUs, which is more than 2 BLEUs higher than the best existing results. And because the encoder side is calculated in parallel, the training time is greatly shortened.

Its pioneering thought overturned the previous idea of equating sequence modeling and RNNs, and is now widely used in various fields of NLP. The language models that are currently in full bloom in various NLP businesses, such as GPT⁽²⁾, BERT⁽³⁾, Roberta⁽⁴⁾, XLNet⁽⁵⁾ are all based on the Transformer model. Therefore, it is particularly important to clarify every detail inside the Transformer model.

Machine translation started from RNN⁽⁶⁾ and entered the era of neural network machine translation. The following briefly introduces the difference between RNN and Transformer. Transformer is a Sequence to Sequence model. The special feature is that it uses a lot of self-attention. In dealing with end-to-end models⁽⁷⁾, the most commonly used is the RNN. Its input is a string of vector sequences, and its output is another string of vector sequences, as shown in the Figure 1 below. RNN is very good at handling situations where input is a sequence. The problem is that RNN is not easy to be parallelized, which is not conducive to large-scale and fast training and deployment. Transformer abandoned the traditional CNN and RNN and put the attention mechanism into full play. The entire network structure is entirely composed of the attention mechanism, which is a relatively big improvement. His input and output are the same as RNN and are end-to-end. Every output of his has seen the entire input, but the magic is that every input of his can be calculated in parallel.

However, CNN^(8,9), as a model that is good at capturing local features, also has many shortcomings. Visual data cannot satisfy higher-order spatiotemporal relationships.

Inspired by Transformer's powerful representation

Fig. 1. Encoder-Decoder model.

capabilities in the field of natural language processing, the Transformer model and its variants have been successfully used in Computer Vision (CV) task, such as Image classification^(10,11), object detection^(12,13), segmentation^(14,15), Pose Estimation⁽¹⁶⁾, Image enhancement^(17,18), image generation⁽¹⁹⁾, Video inpainting⁽²⁰⁾, Video captioning⁽²¹⁾ and Efficient transformer⁽²²⁻²⁵⁾.

More and more researchers engaged in computer vision pay attention to Transformer, which has led to a rapid increase in the number of improved models based on Transformer. According to the survey, there have been 800 searches in arxiv using transformer keywords in 2021. After investigation, most of the literature reviews on Transformer mainly focus on the field of NLP^(26,27) or generic attention-based methods⁽²⁸⁾. This article aims to provide a comprehensive overview of the computer vision application for Transformer model development. This article takes the emerging field of visual Transformer as the research object, we first introduce the prominent concept and self-attention mechanism of the transformer network, introduce the reformer model after the transformer is improved, and then introduce the specific situation of the recent visual transformer in detail, which is also the main content of this document. At the end of this study, several issues to be studied are detailed and prospects for possible future work are given.

2. Foundations of Transformer

As shown in the figure 2, the overall structure of transformer includes six encoders and six decoders with the same structure⁽²⁹⁾. The design idea of each encoder and decoder is the same. Each decoder and encoder have two identical sub layers: self-attention and feed-forward neural network, and the decoder has an additional encoder-decoder attention layer. For convenience, all sub layers and embedded layers in the model generate output with d_{model} = 512 dimensions. So how does self-attention work?

2.1 Positional Encoding

First of all, the input data needs to be transformed into 512-dimensional vector, but it is not enough to input the vector after word embedding into the encoder, because there is no loop structure similar to RNN in the transformer, and there is no ability to capture the sequence, or no matter

Fig. 2. Pipeline of transformer. The image is from (29)

how the sentence structure is disturbed, the transformer will get similar results. In order to solve this problem, position encoding vector is introduced to represent the distance between two words. In short, position encoding vector is added to the word vector. The sin cos function is used here, which makes it easy to understand the relative position of the model. Specifically, the following rules are used for coding:

$$PE_{(pos,2i)} = \sin(pos / 10000^{2i/d_{model}}) \quad (1)$$

$$PE_{(pos,2i+1)} = \cos(pos / 10000^{2i/d_{model}}) \quad (2)$$

Where pos represents the position of the word in the sentence, and i represents the current dimension of the position code.

2.2 Self-attention

2.2.1 Scaled Dot-Product Attention

In the self-attention layer, suppose that our input is a sequence. Each input is multiplied by a matrix ω to get embedding, in order to get an embedded with a dimension of 512. Then the embedding goes into the self-attention layer. Each vector is multiplied by three different transformation matrices ($\omega_q, \omega_k, \omega_v$) to get three different vectors:

query vector q , key vector k and value vector v . The

three vectors have the same dimensions as d_{model} . All q ,

k , v vectors are packed into matrices Q, K, V .

Next is the attention process of Q and K then divided by the square root of the dimension, making the gradient more stable. Then the result is passed through softmax, which determines the contribution of each word to the encoding current position, and finally multiplies with V

to get the weighted value matrix.

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right) \cdot V \quad (3)$$

This calculation is obviously the calculation process of the attention mechanism mentioned earlier. The coding output of each input word will introduce the coding information of the remaining words through the attention mechanism.

2.2.2 Multi-Head Attention

Although the single-head attention mechanism is very good, in practice, we usually focus on information in multiple locations. Although each code is more or less reflected in it, it may be dominated by the actual word itself. It will affect the attention of other equally important positions. On the basis of single-headed attention, a mechanism called "multi-headed" attention is proposed to further improve the self-attention layer and improve the performance of the attention layer. The specific mechanism is as follows:

It gives multiple "representation subspaces" of the attention layer. For the "multi-head" attention mechanism, there are multiple query/key/value weight matrix sets (Transformer usually uses 8 attention heads, so we have eight matrix sets for each encoder / decoder).

From the figure (3), we can see that Multi-Head Attention contains multiple Self-Attention layers. Since there are a total of 8 self-Attention layers, we use

$$d_k = d_q = d_v = d_{model/8} = 64 \text{ dimensions to reduce the size}$$

of the head and greatly reduce the amount of calculation. For each one, the input is first passed to 8 different Self-Attentions, and 8 output results are calculated. After getting the 8 output matrices, Multi-Head Attention stitches them together and then passes them into a Linear layer to get the final output of Multi-Head Attention. It can be seen that the dimensions of the output matrix of Multi-Head Attention and the input matrix are the same.

2.3 Specific process of encoding and decoding

The encoder is responsible for mapping the input sequence to the hidden layer after position encoding, and then the decoder maps the hidden layer to the output sequence. As is shown in Figure 4. The encoder consists of four parts. The first position of the encoder is to convert the input data into vector, and then input it to the multi attention system after position encoding. Compared with the RNN

Fig. 3. Multi-head attention. The image is from (1).

sequential input mentioned above, the transformer method does not need to input the data one by one, it can directly input in parallel, and store the position relationship between the data, which greatly improves the calculation speed and reduces the storage space. In the second part, the purpose of multi attention is to obtain the correlation within the data, which also makes up for the lack of correlation in CNN method. The third part is residual connection and standardization. In the process of mapping relationship transformation, there are always residual errors generated by calculation, and the existence of residual errors will make the mapping relationship of model learning more and more imprecise because of the increase of network layers. Therefore, the third part, multi head attention, also includes an add & norm layer, Add means residual connection to prevent network degradation, and norm means layer normalization to normalize the activation value of each layer. Finally, through the forward feedback layer composed of two fully connected layers, the addition of FFN introduces nonlinearity (relu activation function), which transforms the space of attention output and increases the performance of the model. In this way, the learning results obtained by the encoder are more accurate and representative.

3. Reformer

Transformer is characterized by multi head attention mechanism, feedforward neural network, position coding and residual connection. These different components make each element interact with global information like CNN, ignore distance, and has good scalability and scalability. It breaks through the limitation that RNN model cannot be calculated in parallel. Through parallel computing, the training speed is accelerated and self-attention can be generated to produce more interpretable model. We can

Fig. 4. Multi-head attention. The image is from (1).

check the attention distribution from the model. Attention head can learn to perform different tasks, and it has been widely used in the context semantics of natural language processing⁽³⁰⁾.

Although the transformer model can produce very good results and is used in more and more long sequences, such as 11K text, many of these large models can only be trained on large industrial computing platforms, and cannot run on a single GPU, because their memory requirements are too large. For example, the complete gpt-2 model⁽³¹⁾ contains about 1.5b parameters. The maximum number of parameters is more than 0.5B per layer, and there are 64 layers. In view of some problems existing in the party, Kitaev et al. Optimized some of the structures and proposed a new model reformer⁽³²⁾.

3.1 LSH Attention

In order to solve the problem that the memory required by transformer is too large, reformer improves its attention part and replaces the Scaled Dot-Product Attention with the locality sensitivity hashing attention. Transformer's multi-heads attention is parallel computing and superposition. Assuming that the length of the sequence in the experiment is L , because of $Q \cdot K^T$, it will form a matrix of $L * L$. It computes the attention score between two data points and needs to connect multiple self-attention points, which results in a large amount of computation and a large amount of memory. In reformer, locality sensitivity Hashing attention

is selected to replace multi-heads attention. The basic idea of locally sensitive hashing is shown in Figure 5.

LSH (locality sensitivity hashing) algorithm is a similarity search algorithm in massive data. A special hash function is designed to map two data with high similarity to the same hash value with high probability. We call this function LSH. The most fundamental function of LSH is to efficiently deal with the nearest neighbor problem of massive high-dimensional data. As shown in the figure5, data with similar q , k_1 , k_2 attention scores will be mapped to the same space, and other data with little correlation will be mapped to the other space. After mapping here, q k_1 nearest and similar data will be divided into the same space again, and other data will also be sorted according to the attention score scores, which facilitates data searching and reduces the amount of calculation. Therefore, the computational complexity is greatly reduced, and the efficiency of long sequence training is improved⁽³³⁾.

3.2 Reversible residual network

Although LSH attention solves the problem of too much computation, because of the back-propagation calculation in the training network, a single layer of network model usually needs several GB of memory, but because transformer is a multi-layer model with gradient descent, the activation value of each layer is often retained and used in the next layer. Typical transformer model needs at least 12 or more layers due to FFN and multi-heads attention layer. Therefore, if it is used to cache the value of each layer, the memory is far from enough.

Therefore, reformer improves the Transformer model, and proposes that the reversible residual is used instead of the standard residual layer⁽³⁴⁾ to reduce the memory consumption. In reverse propagation, only the activation

Fig. 5. The basic idea of locally sensitive hash.

value of the last layer needs to be stored. The activation value of the last layer of the network can recover the activation value of any layer in the middle. The basic diagram of reversible residual layer is shown in Figure 6[34].

As can be seen from figure 6, the left side of the figure is the standard residual layer. The activation value of each layer should be reserved for the next layer update, so it needs more memory. On the right of the figure is the reversible residual layer, which divides an input into two parts, x_1 and x_2 . Here, it is divided into two channels. In the first channel, x_1 remains unchanged, plus the value of x_2 after attention, the first output y_1 is obtained. In the second channel, x_2 remains unchanged, plus the value of y_1 after ffn, the second output y_2 is obtained. As can be seen from the right of the figure, after we know y_1 and y_2 , we can get x_2 by subtracting y_1 from ffn. Similarly, we can get x_1 . We only need to know the activation value of the top layer, then we can know all the activation values of the lower layer. In this way, when training the network, many applications will not be limited because the activation value takes up a lot of memory.

4. Visual transformer

After understanding the standard transformer for NLP, most image researchers find it very simple to use transformer in the field of vision. Therefore, in this chapter, we will introduce the overview of several transformers designed for visual tasks. We will make different classifications to explain these research directions.

Fig. 6. Reversible residual network.

Fig. 7. The framework of ViT. The image is from (11).

4.1 Transformers for Image Classification

This paper is different from the traditional image classification, does not use the traditional CNN, as far as possible the NLP domain transformer does not modify the migration to the CV domain. Experimental results show that only using transformer in large-scale training set, the performance of classification task is very good. After the pre-trained model of large-scale data sets is transferred to the classification task of small data sets, the performance is also very good. The following figure 7 shows the architecture of the vit model⁽¹¹⁾.

Because the language data processed by NLP is serialized, the image data processed in practice is three-dimensional (height, width and channels). Therefore, we need to preprocess the image, block and reduce the dimension, because the transformer wants to input a two-dimensional matrix (N, D) , where n is the length of the sequence and D is the dimension of each vector of the sequence. Therefore, we need to transform these 3D data into serialized data by block method. $H \times W \times C$ needs to change $N \times (p \cdot p \cdot C)$. In fact, P is the resolution of each image after blocking. $N = HW / (P^2)$.

To convert it into 2D input, we need to add another step of patch embedding. The method is to do a linear transformation (i.e. full connection layer) for each vector, and the compressed dimension is D , which is called patch embedding here. Combined with position coding, the formula is as follows:

$$Z_0 = [x_{class}; x_p^1 E; x_p^1 E x_p^2; \dots; x_p^N E] + E_{pos} \quad (4)$$

Because vit only uses the encoder of transformer, for

the classification task, assuming that there are nine input and nine output coding vectors, it is impossible to determine which vector to classify. Therefore, we need to add a learnable embedding vector to find the image categories corresponding to the other nine input vectors.

4.2 Transformers for Image Classification

The task of object detection is to predict the category of a series of objects in a given picture and the coordinates of the bounding box. Most of the existing detectors construct the problem into an unordered set prediction behavior by using some dispose, anchor or windows. In view of this problem⁽³⁶⁾, Carion presents the DETR model⁽³⁵⁾, improves the transformer model, and applies transformers to the object detection field for the first time. When inputting a picture, the feature of the picture is extracted by combining with CNN, and then it is transformed into a feature sequence and input into the transformer model. The disordered set with a specified length of B is obtained. Each set includes the category and coordinate of the object. As the first way to use end to end, the target detection has achieved good results. The following figure 8 shows the overall framework of DETR.

First, DETR starts from a CNN backbone to extract features from the input image. The position information is used to supplement the image features, the fixed position coding is added to the flat features, and then sent to the encoding and decoding transformer. The transformer decoder uses the embedding from the encoder and N learned position coding, and generates N output embedding. The final prediction is calculated by a simple feedforward network (FFN), which includes frame coordinates and class labels to indicate whether the object is of a specific class or not.

The key lies in two parts, the first part is in the design, the first is to use the encoder decoder architecture of transformer to generate N box predictions at one time. Where n is a preset integer whose ratio is much larger than the number of objects in the image. The second is to design bipartite matching loss, which calculates the size of loss based on bipartite graph matching between predicted boxes and ground truth boxes, so that the position and category of

Fig. 8. The architecture of DETR. The image is from (35).

predicted box are closer to ground truth.

4.3 Transformers for Image Processing

IPT⁽¹⁷⁾ is the first pre-training model that uses the powerful representation ability of transformer to deal with the application of de-noising, de rain and super score in the underlying visual tasks. However, due to the limited data set to meet specific tasks. Light, weather, camera and other factors will affect the quality of data. However, in order to meet the multi task, we do not know which type of image processing task to request before the test image comes out, because we must match different heads and tails to complete the corresponding underlying visual tasks. Since IPT is suitable for three tasks, it needs three heads and tails and a shared body.

As shown in Figure 9, the IPT consists of multiple headers, an encoder, a decoder, and multiple tails. For different image processing tasks, multi head and multi tail structures and task embedding are introduced. The head is used to extract features from input damaged images (such as noisy images and low resolution images). These features are divided into patches, which are input into the encoder decoder architecture. The codec converter is used to recover the lost information in the input data, and then the output is reshaped to features of the same size. IPT pre-trains by using the processed ImageNet dataset. IPT greatly improves the performance of image processing tasks. The pre-training model can be effectively used for the required tasks after fine tuning.

5. Conclusions and Discussions

5.1 Conclusions

As a powerful deep neural network model, transformer is firstly used to process sequences in natural language. Compared with CNN, it has been widely used in computer vision. This paper summarizes the self-attention mechanism, and also focuses on the transformer structure composed of

Fig. 9. The architecture of IPT. The image is from (36).

self-attention mechanism. Meanwhile, it also introduces the efficient reformer model improved by transformer. This paper analyzes several important applications of attention mechanism in the development of deep learning algorithm in the field of CV, such as image classification, target detection and so on, Image processing and other attention mechanism algorithms have good performance and high research significance, hoping to promote the further development of transformer.

5.2 Future Prospects

In order to promote the development of visual transformer, we provide several potential directions for future research.

(1) At present, some work has been done to simplify the structure of transformer, but the practical application to hardware is very difficult, so it is necessary to lighten the transformer in the future.

(2) Most of the data sets used by transformer are public data sets, but the number of data sets used in practice is small. In the future, we need to improve how to apply to the transformer with small amount of data.

Based on the attention mechanism algorithm, combined with its own shortcomings and continuous improvement, transformer will be applied to more fields to explore more possibilities

References

- (1) Ashish Vaswani, Noam Shazeer, Niki Parmar Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin : “Attention is all you need” , neural information processing systems, 5998-6008,2017.
- (2) Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever : “Improving language understanding by generative pre-training”, OpenAI. 2018.
- (3) Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova : “Bert: Pre-training of deep bidirectional transformers for language understanding”, arXiv preprint arXiv:1810.04805, 2018.
- (4) Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov : “ Roberta: A robustly optimized bert pretraining approach”, arXiv preprint arXiv:1907.11692, 2019.
- (5) Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V : “Le generalized autoregressive pretraining for language understanding”, arXiv 2019; 1906.08237[J]. 1906.
- (6) Hochreiter, Sepp, and Jürgen Schmidhuber : “Long short-term memory”, Neural computation 9.8,1735-1780.1997.
- (7) Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio : “Learning phrase representations using RNN encoder-decoder for statistical machine translation”, arXiv preprint arXiv:1406.1078, 2014.
- (8) Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner : “Gradient-based learning applied to document recognition”, Proceedings of the IEEE, VOL.86 , NO.11, 2278-2324,1998.
- (9) Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton : “Imagenet classification with deep convolutional neural networks”, Advances in neural information processing systems, VOL.25: 1097-1105,2012.
- (10) Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever : “Generative pretraining from pixels”, International Conference on Machine Learning. PMLR, ,1691-1703,2020.
- (11) 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”, arXiv preprint arXiv:2010.11929, 2020.
- (12) Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai : “Deformable detr: Deformable transformers for end-to-end object detection”, arXiv preprint arXiv:2010.04159, 2020.
- (13) Minghang Zheng, Peng Gao, Xiaogang Wang, Hongsheng Li, and Hao Dong : “End-to-end object detection with adaptive clustering transformer”, arXiv preprint arXiv:2011.09315, 2020.
- (14) Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen, : “Max-deeplab: End-to-end panoptic segmentation with mask transformers”, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,5463-5474,2020.
- (15) Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia : “End-to-end video instance segmentation with transformers”.

- Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 8741-8750,2021.
- (16) Lin Huang, Jianchao Tan, Jingjing Meng , Ji Liu, Junsong Yuan : “HOT-Net: Non-autoregressive transformer for 3D hand-object pose estimation”, Proceedings of the 28th ACM International Conference on Multimedia. 3136-3145,2020.
 - (17) Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao : “Pre-trained image processing transformer” , Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,12299-12310,2020.
 - (18) Fuzhi Yang, Huan Yang, Jianlong Fu, Hongtao Lu, and Baining Guo : “ Learning texture transformer network for image super-resolution”, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,5791-5800,2020.
 - (19) Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran : “ Image transformer”, International Conference on Machine Learning. PMLR, 4055-4064,2020.
 - (20) Yanghong Zeng, Jianlong Fu, and Hongyang Chao : “Learning joint spatial-temporal transformations for video inpainting”, European Conference on Computer Vision. Springer, Cham, 528-543,2020.
 - (21) Luowei Zhou, Yingbo Zhou, Jason J. Corso, Richard Socher, and Caiming Xiong : “End-to-end dense video captioning with masked transformer”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,8739-8748,2018.
 - (22) Paul Michel, Omer Levy, and Graham Neubig: “Are sixteen heads really better than one?”, arXiv preprint arXiv:1905.10650, 2019.
 - (23) Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu : “Tinybert: Distilling bert for natural language understanding”, arXiv preprint arXiv:1909.10351, 2019.
 - (24) Gabriele Prato, Ella Charlaix, and Mehdi Rezagholizadeh, : “Fully quantized transformer for machine translation”, arXiv preprint arXiv:1910.10485, 2019.
 - (25) Zihang Jiang, Weihao Yu, Daquan Zhou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan : “Convbert: Improving bert with span-based dynamic convolution”, arXiv preprint arXiv:2008.02496, 2020.
 - (26) Dichao Hu : “An introductory survey on attention mechanisms in NLP problems”, Proceedings of SAI Intelligent Systems Conference. Springer, Cham, 432-448,2020.
 - (27) Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler : “Efficient transformers: A survey”, arXiv preprint arXiv:2009.06732, 2020.
 - (28) Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath : “An attentive survey of attention models”. arXiv preprint arXiv:1904.02874, 2019.
 - (29) Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao : “A survey on visual transformer”. arXiv preprint arXiv:2012.12556, 2020.
 - (30) Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret : “Transformers are rnns: Fast autoregressive transformers with linear attention”, International Conference on Machine Learning. PMLR, 5156-5165,2020.
 - (31) Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever : “Language models are unsupervised multitask learners”, OpenAI blog, Vol. 1, No. 8, pp.9,2019. Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya: “Reformer: The efficient transformer”, arXiv preprint arXiv:2001.04451, 2020.
 - (32) Kitano, Hugo, and Taide Ding ; “Applying and Adapting the Reformer as a Computationally Efficient Approach to the SQuAD 2.0 Question-Answering Task”, 2020-5-12]. <https://web.stanford.edu/class/cs224n/reports/default/report07.Pdf>,2020.
 - (33) Aidan N. Gomez, Mengye Ren, Urtasun R, Raquel Urtasun, and Roger B. Grosse : “The reversible residual network: Backpropagation without storing activations”, Proceedings of the 31st International Conference on Neural Information Processing Systems. 2211-2221,2020.
 - (34) Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko : “End-to-end object detection with transformers”, European Conference on Computer Vision. Springer, Cham, 213-229,2020.
 - (35) Bharat Singh, Hengduo Li, Abhishek Sharma, and Larry S. Davis : “R-fcn-3000 at 30fps: Decoupling detection and classification”, Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 1081-1090,2020.