A Survey on Vision Transformer

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 *Fellow, IEEE*

Abstract—Transformer, first applied to the field of natural language processing, is a type of deep neural network mainly based on the self-attention mechanism. Thanks to its strong representation capabilities, researchers are looking at ways to apply transformer to computer vision tasks. In a variety of visual benchmarks, transformer-based models perform similar to or better than other types of networks such as convolutional and recurrent neural networks. Given its high performance and less need for vision-specific inductive bias, transformer is receiving more and more attention from the computer vision community. In this paper, we review these vision transformer models by categorizing them in different tasks and analyzing their advantages and disadvantages. The main categories we explore include the backbone network, high/mid-level vision, low-level vision, and video processing. We also include efficient transformer methods for pushing transformer into real device-based applications. Furthermore, we also take a brief look at the self-attention mechanism in computer vision, as it is the base component in transformer. Toward the end of this paper, we discuss the challenges and provide several further research directions for vision transformers.

Index Terms—Transformer, Self-attention, Computer Vision, High-level vision, Low-level vision, Video.

1 Introduction

Deep neural networks (DNNs) have become the fundamental infrastructure in today's artificial intelligence (AI) systems. Different types of tasks have typically involved different types of networks. For example, multi-layer perceptron (MLP) or the fully connected (FC) network is the classical type of neural network, which is composed of multiple linear layers and non-linear activations stacked together [1], [2]. Convolutional neural networks (CNNs) introduce convolutional layers and pooling layers for processing shift-invariant data such as images [3], [4]. And recurrent neural networks (RNNs) utilize recurrent cells to process sequential data or time series data [5], [6]. Transformer is a new type of neural network. It mainly utilizes the self-attention mechanism [7], [8] to extract intrinsic features [9] and shows great potential for extensive use in AI applications.

Transformer was first applied to natural language processing (NLP) tasks where it achieved significant improvements [9], [10], [11]. For example, Vaswani *et al.* [9] first proposed transformer based on attention mechanism for machine translation and English constituency parsing tasks. Devlin *et al.* [10] introduced a new language representation model called BERT (short for Bidirectional Encoder Representations from Transformers), which pre-trains a transformer on unlabeled text taking into account the context of each word as it is bidirectional. When BERT was published, it obtained state-of-the-art performance on 11 NLP tasks. Brown *et al.* [11] pre-trained a massive transformer-based model called GPT-3 (short for Generative Pre-trained Transformer 3) on 45 TB of compressed plaintext data using 175 billion parameters. It achieved strong performance on different types of downstream natural language tasks without requiring any fine-tuning. These transformer-based models, with their strong representation capacity, have achieved significant breakthroughs in NLP.

Inspired by the major success of transformer architectures in the field of NLP, researchers have recently applied transformer to computer vision (CV) tasks. In vision applications, CNNs are considered the fundamental component [12], [13], but nowadays transformer is showing it is a potential alternative to CNN. Chen *et al.* [14] trained a sequence transformer to auto-regressively predict pixels, achieving results comparable to CNNs on image classification tasks. Another vision transformer model is ViT, which applies a pure transformer directly to sequences of image patches to classify the full image. Recently proposed by Dosovitskiy *et al.* [15], it has achieved state-of-the-art performance on multiple image recognition benchmarks. In addition to image classification, transformer has been utilized to address a variety of other vision problems, including object detection [16], [17], semantic segmentation [18], image processing [19], and video understanding [20]. Thanks to its exceptional performance, more and more researchers are proposing transformer-based models for improving a wide range of visual tasks.

Due to the rapid increase in the number of transformer-based vision models, keeping pace with the rate of new progress is becoming increasingly difficult. As such, a survey of the existing works is urgent and would be beneficial for the community. In this paper, we focus on providing a comprehensive overview of the recent advances in vision transformers and discuss the potential directions for further improvement. To facilitate future research on different topics, we categorize the transformer models by their application scenarios, as listed in Table 1. The main categories include backbone network, high/mid-level vision, low-level vision, and video processing. High-level vision deals with the interpretation and use of what is seen in the image [21], whereas mid-level vision deals with how this information is organized into what we experience as objects and surfaces [22]. Given the gap

- *Kai Han, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Yunhe Wang are with Huawei Noah's Ark Lab. E-mail: {kai.han, yunhe.wang}@huawei.com.*
- *Hanting Chen, Zhenhua Liu, Yehui Tang, and Zhaohui Yang are also with School of EECS, Peking University.*
- *Dacheng Tao is with the School of Computer Science, in the Faculty of Engineering, at The University of Sydney, 6 Cleveland St, Darlington, NSW 2008, Australia. E-mail: dacheng.tao@sydney.edu.au.*
- *Corresponding to Yunhe Wang and Dacheng Tao.*
- *All authors are listed in alphabetical order of last name (except the primary and corresponding authors).*

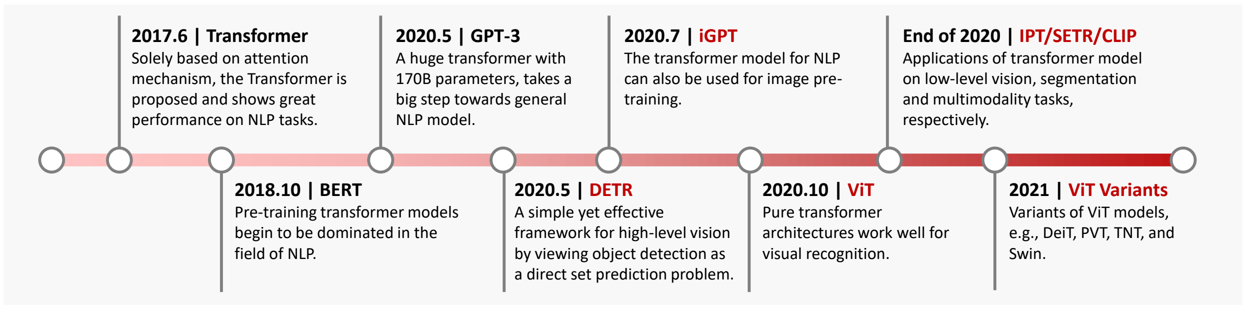

2017.6 | Transformer
Solely based on attention mechanism, the Transformer is proposed and shows great performance on NLP tasks.
2020.5 | GPT-3
A huge transformer with 170B parameters, takes a big step towards general NLP model.
2020.7 | iGPT
The transformer model for NLP can also be used for image pre-training.
End of 2020 | IPT/SETR/CLIP
Applications of transformer model on low-level vision, segmentation and multimodality tasks, respectively.
2018.10 | BERT
Pre-training transformer models begin to be dominated in the field of NLP.
2020.5 | DETR
A simple yet effective framework for high-level vision by viewing object detection as a direct set prediction problem.
2020.10 | ViT
Pure transformer architectures work well for visual recognition.
2021 | ViT Variants
Variants of ViT models, e.g., DeiT, PVT, TNT, and Swin.

Fig. 1: Key milestones in the development of transformer. The vision transformer models are marked in red.

between high- and mid-level vision is becoming more obscure in DNN-based vision systems [23], [24], we treat them as a single category here. A few examples of transformer models that address these high/mid-level vision tasks include DETR [16], deformable DETR [17] for object detection, and Max-DeepLab [25] for segmentation. Low-level image processing mainly deals with extracting descriptions from images (such descriptions are usually represented as images themselves) [26]. Typical applications of low-level image processing include super-resolution, image denoising, and style transfer. At present, only a few works [19], [27] in low-level vision use transformers, creating the need for further investigation. Another category is video processing, which is an important part in both computer vision and image-based tasks. Due to the sequential property of video, transformer is inherently well suited for use on video tasks [20], [28], in which it is beginning to perform on par with conventional CNNs and RNNs. Here, we survey the works associated with transformer-based visual models in order to track the progress in this field. Figure 1 shows the development timeline of vision transformer — undoubtedly, there will be many more milestones in the future.

The rest of the paper is organized as follows. Section 2 discusses the formulation of the standard transformer and the self-attention mechanism. Section 4 is the main part of the paper, in which we summarize the vision transformer models on backbone, high/mid-level vision, low-level vision, and video tasks. We also briefly describe efficient transformer methods, as they are closely related to our main topic. In the final section, we give our conclusion and discuss several research directions and challenges. Due to the page limit, we describe the methods of transformer in NLP in the supplemental material, as the research experience may be beneficial for vision tasks. In the supplemental material, we also review the self-attention mechanism for CV as the supplementary of vision transformer models. In this survey, we mainly include the representative works (early, pioneering, novel, or inspiring works) since there are many preprinted works on arXiv and we cannot include them all in limited pages.

2 Formulation of Transformer

Transformer [9] was first used in the field of natural language processing (NLP) on machine translation tasks. As shown in Figure 2, it consists of an encoder and a decoder with several transformer blocks of the same architecture. The encoder generates encodings of inputs, while the decoder takes all the encodings and using their incorporated contextual information to generate the output sequence. Each transformer block is composed of a multi-head attention layer, a feed-forward neural network, shortcut connection and layer normalization. In the following, we describe each component of the transformer in detail.

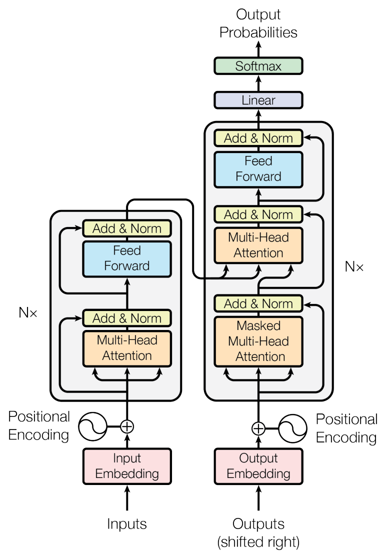


Fig. 2: Structure of the original transformer (image from [9]).

2.1 Self-Attention

In the self-attention layer, the input vector is first transformed into three different vectors: the query vector $\mathbf{q}$, the key vector $\mathbf{k}$ and the value vector $\mathbf{v}$ with dimension $d_q = d_k = d_v = d_{model} = 512$. Vectors derived from different inputs are then packed together into three different matrices, namely, $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$. Subsequently, the attention function between different input vectors is calculated as follows (and shown in Figure 3 left):

- **Step 1:** Compute scores between different input vectors with $\mathbf{S} = \mathbf{Q} \cdot \mathbf{K}^\top$;
- **Step 2:** Normalize the scores for the stability of gradient with $\mathbf{S}_n = \mathbf{S}/\sqrt{d_k}$;
- **Step 3:** Translate the scores into probabilities with softmax function $\mathbf{P} = \mathrm{softmax}(\mathbf{S}_n)$;
- **Step 4:** Obtain the weighted value matrix with $\mathbf{Z} = \mathbf{V} \cdot \mathbf{P}$.

The process can be unified into a single function:

$$\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}(\frac{\mathbf{Q} \cdot \mathbf{K}^\top}{\sqrt{d_k}}) \cdot \mathbf{V}. \quad (1)$$

The logic behind Eq. 1 is simple. Step 1 computes scores between each pair of different vectors, and these scores determine the degree of attention that we give other words when encoding the word at the current position. Step 2 normalizes the scores to enhance gradient stability for improved training, and step 3 translates the scores into probabilities. Finally, each value vector is multiplied by the sum of the probabilities. Vectors with larger probabilities receive additional focus from the following layers.

TABLE 1: Representative works of vision transformers.

Category	Sub-category	Method	Highlights	Publication
Backbone	Supervised pretraining	ViT [15]	Image patches, standard transformer	ICLR 2021
		TNT [29]	Transformer in transformer, local attention	NeurIPS 2021
		Swin [30]	Shifted window, window-based self-attention	ICCV 2021
	Self-supervised pretraining	iGPT [14]	Pixel prediction self-supervised learning, GPT model	ICML 2020
		MoCo v3 [31]	Contrastive self-supervised learning, ViT	ICCV 2021
High/Mid-level vision	Object detection	DETR [16]	Set-based prediction, bipartite matching, transformer	ECCV 2020
		Deformable DETR [17]	DETR, deformable attention module	ICLR 2021
		UP-DETR [32]	Unsupervised pre-training, random query patch detection	CVPR 2021
	Segmentation	Max-DeepLab [25]	PQ-style bipartite matching, dual-path transformer	CVPR 2021
		VisTR [33]	Instance sequence matching and segmentation	CVPR 2021
		SETR [18]	Sequence-to-sequence prediction, standard transformer	CVPR 2021
	Pose Estimation	Hand-Transformer [34]	Non-autoregressive transformer, 3D point set	ECCV 2020
		HOT-Net [35]	Structured-reference extractor	MM 2020
		METRO [36]	Progressive dimensionality reduction	CVPR 2021
Low-level vision	Image generation	Image Transformer [27]	Pixel generation using transformer	ICML 2018
		Taming transformer [37]	VQ-GAN, auto-regressive transformer	CVPR 2021
		TransGAN [38]	GAN using pure transformer architecture	NeurIPS 2021
	Image enhancement	IPT [19]	Multi-task, ImageNet pre-training, transformer model	CVPR 2021
		TTSR [39]	Texture transformer, RefSR	CVPR 2020
Video processing	Video inpainting	STTN [28]	Spatial-temporal adversarial loss	ECCV 2020
	Video captioning	Masked Transformer [20]	Masking network, event proposal	CVPR 2018
Multimodality	Classification	CLIP [40]	NLP supervision for images, zero-shot transfer	arXiv 2021
	Image generation	DALL-E [41]	Zero-shot text-to image generation	ICML 2021
		Cogview [42]	VQ-VAE, Chinese input	NeurIPS 2021
	Multi-task	UniT [43]	Different NLP & CV tasks, shared model parameters	ICCV 2021
Efficient transformer	Decomposition	ASH [44]	Number of heads, importance estimation	NeurIPS 2019
	Distillation	TinyBert [45]	Various losses for different modules	EMNLP Findings 2020
	Quantization	FullyQT [46]	Fully quantized transformer	EMNLP Findings 2020
	Architecture design	ConvBert [47]	Local dependence, dynamic convolution	NeurIPS 2020

The encoder-decoder attention layer in the decoder module is similar to the self-attention layer in the encoder module with the following exceptions: The key matrix K and value matrix V are derived from the encoder module, and the query matrix Q is derived from the previous layer.

Note that the preceding process is invariant to the position of each word, meaning that the self-attention layer lacks the ability to capture the positional information of words in a sentence. However, the sequential nature of sentences in a language requires us to incorporate the positional information within our encoding. To address this issue and allow the final input vector of the word to be obtained, a positional encoding with dimension d_{model} is added to the original input embedding. Specifically, the position is encoded with the following equations:

$$PE(pos, 2i) = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}); \tag{2}$$

$$PE(pos, 2i+1) = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}), \tag{3}$$

in which pos denotes the position of the word in a sentence, and i represents the current dimension of the positional encoding. In this way, each element of the positional encoding corresponds to a sinusoid, and it allows the transformer model to learn to attend by relative positions and extrapolate to longer sequence lengths during inference. In apart from the fixed positional encoding in the vanilla transformer, learned positional encoding [48] and relative positional encoding [49] are also utilized in various models [10], [15].

Multi-Head Attention. Multi-head attention is a mechanism that can be used to boost the performance of the vanilla self-attention layer. Note that for a given reference word, we often want to focus on several other words when going through the sentence. A single-head self-attention layer limits our ability to focus on one or more specific positions without influencing the attention on other equally important positions at the same time.

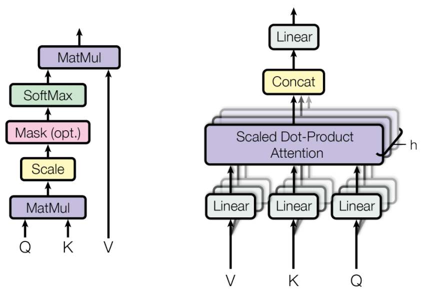


Fig. 3: (Left) Self-attention process. (Right) Multi-head attention. The image is from [9].

This is achieved by giving attention layers different representation subspace. Specifically, different query, key and value matrices are used for different heads, and these matrices can project the input vectors into different representation subspace after training due to random initialization.

To elaborate on this in greater detail, given an input vector and the number of heads h, the input vector is first transformed into three different groups of vectors: the query group, the key group and the value group. In each group, there are h vectors with dimension $d_{q'} = d_{k'} = d_{v'} = d_{model}/h = 64$. The vectors derived from different inputs are then packed together into three different groups of matrices: $\{\mathbf{Q}_i\}_{i=1}^h$, $\{\mathbf{K}_i\}_{i=1}^h$ and $\{\mathbf{V}_i\}_{i=1}^h$. The multi-head attention process is shown as follows:

$$\begin{aligned}\text{MultiHead}(\mathbf{Q}', \mathbf{K}', \mathbf{V}') &= \text{Concat}(\text{head}_1, \cdots, \text{head}_h)\mathbf{W}^o, \\ \text{where head}_i &= \text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i).\end{aligned} \tag{4}$$

Here, $\mathbf{Q}'$ (and similarly $\mathbf{K}'$ and $\mathbf{V}'$) is the concatenation of $\{\mathbf{Q}_i\}_{i=1}^h$, and $\mathbf{W}^o \in \mathbb{R}^{d_{model} \times d_{model}}$ is the projection weight.

2.2 Other Key Concepts in Transformer

Feed-Forward Network. A feed-forward network (FFN) is applied after the self-attention layers in each encoder and decoder. It consists of two linear transformation layers and a nonlinear activation function within them, and can be denoted as the following function:

$$\mathrm{FFN}(\mathbf{X}) = \mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{X}), \tag{5}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are the two parameter matrices of the two linear transformation layers, and σ represents the nonlinear activation function, such as GELU [50]. The dimensionality of the hidden layer is $d_h = 2048$.

Residual Connection in the Encoder and Decoder. As shown in Figure 2, a residual connection is added to each sub-layer in the encoder and decoder. This strengthens the flow of information in order to achieve higher performance. A layer-normalization [51] is followed after the residual connection. The output of these operations can be described as:

$$\mathrm{LayerNorm}(\mathbf{X} + \mathrm{Attention}(\mathbf{X})). \tag{6}$$

Here, $\mathbf{X}$ is used as the input of self-attention layer, and the query, key and value matrices $\mathbf{Q}, \mathbf{K}$ and $\mathbf{V}$ are all derived from the same input matrix $\mathbf{X}$. A variant pre-layer normalization (Pre-LN) is also widely-used [52], [53], [15]. Pre-LN inserts the layer normalization inside the residual connection and before multi-head attention or FFN. For the normalization layer, there are several alternatives such as batch normalization [54]. Batch normalization usually perform worse when applied on transformer as the feature values change acutely [55]. Some other normalization algorithms [56], [55], [57] have been proposed to improve training of transformer.

Final Layer in the Decoder. The final layer in the decoder is used to turn the stack of vectors back into a word. This is achieved by a linear layer followed by a softmax layer. The linear layer projects the vector into a logits vector with d_{word} dimensions, in which d_{word} is the number of words in the vocabulary. The softmax layer is then used to transform the logits vector into probabilities.

When used for CV tasks, most transformers adopt the original transformer's encoder module. Such transformers can be treated as a new type of feature extractor. Compared with CNNs which focus only on local characteristics, transformer can capture long-distance characteristics, meaning that it can easily derive global information. And in contrast to RNNs, whose hidden state must be computed sequentially, transformer is more efficient because the output of the self-attention layer and the fully connected layers can be computed in parallel and easily accelerated. From this, we can conclude that further study into using transformer in computer vision as well as NLP would yield beneficial results.

3 Vision Transformer

In this section, we review the applications of transformer-based models in computer vision, including image classification, high/mid-level vision, low-level vision and video processing. We also briefly summarize the applications of the self-attention mechanism and model compression methods for efficient transformer.

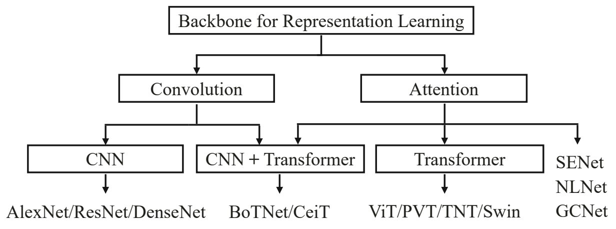


Fig. 4: A taxonomy of backbone using convolution and attention.

3.1 Backbone for Representation Learning

Inspired by the success that transformer has achieved in the field of NLP, some researchers have explored whether similar models can learn useful representations for images. Given that images involve more dimensions, noise and redundant modality compared to text, they are believed to be more difficult for generative modeling.

Other than CNNs, the transformer can be used as backbone networks for image classification. Wu *et al.* [58] adopted ResNet as a convenient baseline and used vision transformers to replace the last stage of convolutions. Specifically, they apply convolutional layers to extract low-level features that are then fed into the vision transformer. For the vision transformer, they use a *tokenizer* to group pixels into a small number of *visual tokens*, each representing a semantic concept in the image. These *visual tokens* are used directly for image classification, with the transformers being used to model the relationships between tokens. As shown in Figure 4, the works can be divided into purely using transformer for vision and combining CNN and transformer. We summarize the results of these models in Table 2 and Figure 6 to demonstrate the development of the backbones. In addition to supervised learning, self-supervised learning is also explored in vision transformer.

3.1.1 *Pure Transformer*

ViT. Vision Transformer (ViT) [15] is a pure transformer directly applies to the sequences of image patches for image classification task. It follows transformer's original design as much as possible. Figure 5 shows the framework of ViT.

To handle 2D images, the image $X \in \mathbb{R}^{h\times w\times c}$ is reshaped into a sequence of flattened 2D patches $X_p \in \mathbb{R}^{n\times(p^2\cdot c)}$ such that c is the number of channels. (h, w) is the resolution of the original image, while (p, p) is the resolution of each image patch. The effective sequence length for the transformer is therefore $n = hw/p^2$. Because the transformer uses constant widths in all of its layers, a trainable linear projection maps each vectorized path to the model dimension d, the output of which is referred to as patch embeddings.

Similar to BERT's $[class]$ token, a learnable embedding is applied to the sequence of embedding patches. The state of this embedding serves as the image representation. During both pre-training and fine-tuning stage, the classification heads are attached to the same size. In addition, 1D position embeddings are added to the patch embeddings in order to retain positional information. It is worth noting that ViT utilizes only the standard transformer's encoder (except for the place for the layer normalization), whose output precedes an MLP head. In most cases, ViT is pre-trained on large datasets, and then fine-tuned for downstream tasks with smaller data.

ViT yields modest results when trained on mid-sized datasets such as ImageNet, achieving accuracies of a few percentage points below ResNets of comparable size. Because transformers lack some inductive biases inherent to CNNs–such as translation equivariance and locality–they do not generalize well when trained on insufficient amounts of data. However, the authors found that training the models on large datasets (14 million to 300 million images) surpassed inductive bias. When pre-trained at sufficient scale, transformers achieve excellent results on tasks with fewer datapoints. For example, when pre-trained on the JFT-300M dataset, ViT approached or even exceeded state of the art performance on multiple image recognition benchmarks.

Specifically, it reached an accuracy of 88.36% on ImageNet, and 77.16% on the VTAB suite of 19 tasks.

Touvron *et al.* [59] proposed a competitive convolution-free transformer, called Data-efficient image transformer (DeiT), by training on only the ImageNet database. DeiT-B, the reference vision transformer, has the same architecture as ViT-B and employs 86 million parameters. With a strong data augmentation, DeiT-B achieves top-1 accuracy of 83.1% (single-crop evaluation) on ImageNet with no external data. In addition, the authors observe that using a CNN teacher gives better performance than using a transformer. Specifically, DeiT-B can achieve top-1 accuracy 84.40% with the help of a token-based distillation.

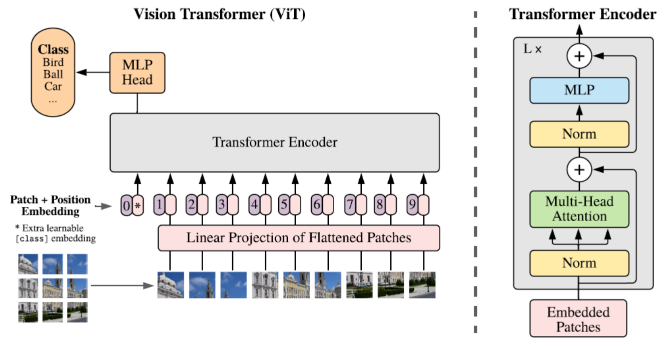


Fig. 5: The framework of ViT (image from [15]).

Variants of ViT. Following the paradigm of ViT, a series of variants of ViT have been proposed to improve the performance on vision tasks. The main approaches include enhancing locality, self-attention improvement and architecture design.

The original vision transformer is good at capturing long-range dependencies between patches, but disregard the local feature extraction as the 2D patch is projected to a vector with simple linear layer. Recently, the researchers begin to pay attention to improve the modeling capacity for local information [29], [60], [61]. TNT [29] further divides the patch into a number of sub-patches and introduces a novel transformer-in-transformer architecture which utilizes an inner transformer block to model the relationship between sub-patches and an outer transformer block for patch-level information exchange. Twins [62] and CAT [63] alternately perform local and global attention layer-by-layer. Swin Transformers [60], [64] performs local attention within a window and introduces a shifted window partitioning approach for cross-window connections. Shuffle Transformer [65], [66] further utilizes the spatial shuffle operation instead of shifted window partitioning to allow cross-window connections. RegionViT [61] generates regional tokens and local tokens from an image, and local tokens receive global information via attention with regional tokens. In addition to the local attention, some other works propose to boost local information through local feature aggregation, *e.g.*, T2T [67]. These works demonstrate the benefit of the local information exchange and global information exchange in vision transformer.

As a key component of transformer, self-attention layer provides the ability for global interaction between image patches. Improving the calculation of self-attention layer has attracted many researchers. DeepViT [68] proposes to establish cross-head communication to re-generate the attention maps to increase the diversity at different layers. KVT [69] introduces the k-NN attention to utilize locality of images patches and ignore noisy tokens by only computing attentions with top-k similar tokens. Refiner [70] explores attention expansion in higher-dimension space and applied convolution to augment local patterns of the attention maps. XCiT [71] performs self-attention calculation across feature channels rather than tokens, which allows efficient processing of high-resolution images. The computation complexity and attention precision of the self-attention mechanism are two key-points for future optimization.

The network architecture is an important factor as demonstrated in the field of CNNs. The original architecture of ViT is a simple stack of the same-shape transformer block. New architecture design for vision transformer has been an interesting topic. The pyramid-like architecture is utilized by many vision transformer models [72], [60], [73], [74], [75], [76] including PVT [72], HVT [77], Swin Transformer [60] and PiT [78]. There are also other types of architectures, such as two-stream architecture [79] and U-net architecture [80], [30]. Neural architecture search (NAS) has also been investigated to search for better transformer architectures, *e.g.*, Scaling-ViT [81], ViTAS [82], AutoFormer [83] and GLiT [84]. Currently, both network design and NAS for vision transformer mainly draw on the experience of CNN. In the future, we expect the specific and novel architectures appear in the filed of vision transformer.

In addition to the aforementioned approaches, there are some other directions to further improve vision transformer, *e.g.*, positional encoding [85], [86], normalization strategy [87], shortcut connection [88] and removing attention [89], [90], [91], [92].

3.1.2 Transformer with Convolution

Although vision transformers have been successfully applied to various visual tasks due to their ability to capture long-range dependencies within the input, there are still gaps in performance between transformers and existing CNNs. One main reason can be the lack of ability to extract local information. Except the above mentioned variants of ViT that enhance the locality, combining the transformer with convolution can be a more straightforward way to introduce the locality into the conventional transformer.

There are plenty of works trying to augment a conventional transformer block or self-attention layer with convolution. For example, CPVT [85] proposed a conditional positional encoding (CPE) scheme, which is conditioned on the local neighborhood of input tokens and adaptable to arbitrary input sizes, to leverage convolutions for fine-level feature encoding. CvT [96], CeiT [97], LocalViT [98] and CMT [94] analyzed the potential drawbacks when directly borrowing Transformer architectures from NLP and combined the convolutions with transformers together. Specifically, the feed-forward network (FFN) in each transformer block is combined with a convolutional layer that promotes the correlation among neighboring tokens. LeViT [99] revisited principles from extensive literature on CNNs and applied them to transformers, proposing a hybrid neural network for fast inference image classification. BoTNet [100] replaced the spatial convolutions with global self-attention in the final three bottleneck blocks of a ResNet, and improved upon the baselines significantly on both instance segmentation and object detection tasks with minimal overhead in latency.

Besides, some researchers have demonstrated that transformer based models can be more difficult to enjoy a favorable ability of fitting data [15], [101], [102], in other words, they are sensitive

1.

TABLE 2: ImageNet result comparison of representative CNN and vision transformer models. Pure transformer means only using a few convolutions in the stem stage. CNN + Transformer means using convolutions in the intermediate layers. Following [59], [60], the throughput is measured on NVIDIA V100 GPU and Pytorch, with 224×224 input size.

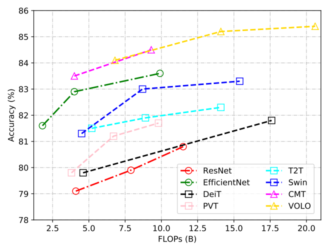
Model	Params (M)	FLOPs (B)	Throughput (image/s)	Top-1 (%)
CNN				
ResNet-50 [12], [67]	25.6	4.1	1226	79.1
ResNet-101 [12], [67]	44.7	7.9	753	79.9
ResNet-152 [12], [67]	60.2	11.5	526	80.8
EfficientNet-B0 [93]	5.3	0.39	2694	77.1
EfficientNet-B1 [93]	7.8	0.70	1662	79.1
EfficientNet-B2 [93]	9.2	1.0	1255	80.1
EfficientNet-B3 [93]	12	1.8	732	81.6
EfficientNet-B4 [93]	19	4.2	349	82.9
Pure Transformer				
DeiT-Ti [15], [59]	5	1.3	2536	72.2
DeiT-S [15], [59]	22	4.6	940	79.8
DeiT-B [15], [59]	86	17.6	292	81.8
T2T-ViT-14 [67]	21.5	5.2	764	81.5
T2T-ViT-19 [67]	39.2	8.9	464	81.9
T2T-ViT-24 [67]	64.1	14.1	312	82.3
PVT-Small [72]	24.5	3.8	820	79.8
PVT-Medium [72]	44.2	6.7	526	81.2
PVT-Large [72]	61.4	9.8	367	81.7
TNT-S [29]	23.8	5.2	428	81.5
TNT-B [29]	65.6	14.1	246	82.9
CPVT-S [85]	23	4.6	930	80.5
CPVT-B [85]	88	17.6	285	82.3
Swin-T [60]	29	4.5	755	81.3
Swin-S [60]	50	8.7	437	83.0
Swin-B [60]	88	15.4	278	83.3
CNN + Transformer				
Twins-SVT-S [62]	24	2.9	1059	81.7
Twins-SVT-B [62]	56	8.6	469	83.2
Twins-SVT-L [62]	99.2	15.1	288	83.7
Shuffle-T [65]	29	4.6	791	82.5
Shuffle-S [65]	50	8.9	450	83.5
Shuffle-B [65]	88	15.6	279	84.0
CMT-S [94]	25.1	4.0	563	83.5
CMT-B [94]	45.7	9.3	285	84.5
VOLO-D1 [95]	27	6.8	481	84.2
VOLO-D2 [95]	59	14.1	244	85.2
VOLO-D3 [95]	86	20.6	168	85.4
VOLO-D4 [95]	193	43.8	100	85.7
VOLO-D5 [95]	296	69.0	64	86.1

to the choice of optimizer, hyper-parameter, and the schedule of training. Visformer [101] revealed the gap between transformers and CNNs with two different training settings. The first one is the standard setting for CNNs, *i.e.*, the training schedule is shorter and the data augmentation only contains random cropping and horizontal flipping. The other one is the training setting used in [59], *i.e.*, the training schedule is longer and the data augmentation is stronger. [102] changed the early visual processing of ViT by replacing its embedding stem with a standard convolutional stem, and found that this change allows ViT to converge faster and enables the use of either AdamW or SGD without a significant drop in accuracy. In addition to these two works, [99], [94] also choose to add convolutional stem on the top of the transformer.

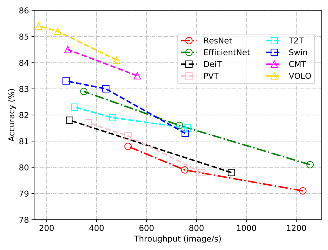
3.1.3 Self-supervised Representation Learning

Generative Based Approach. Generative pre-training methods for images have existed for a long time [103], [104], [105], [106]. Chen *et al.* [14] re-examined this class of methods and combined it with self-supervised methods. After that, several works [107], [108] were proposed to extend generative based self-supervised learning for vision transformer.

We briefly introduce iGPT [14] to demonstrate its mechanism. This approach consists of a pre-training stage followed by a fine-tuning stage. During the pre-training stage, auto-regressive and BERT objectives are explored. To implement pixel prediction, a



(a) Acc v.s. FLOPs.



(b) Acc v.s. throughput.

Fig. 6: FLOPs and throughput comparison of representative CNN and vision transformer models.

sequence transformer architecture is adopted instead of language tokens (as used in NLP). Pre-training can be thought of as a favorable initialization or regularizer when used in combination with early stopping. During the fine-tuning stage, they add a small classification head to the model. This helps optimize a classification objective and adapts all weights.

The image pixels are transformed into a sequential data by k-means clustering. Given an unlabeled dataset X consisting of high dimensional data $\mathbf{x} = (x_1, \cdots, x_n)$, they train the model by minimizing the negative log-likelihood of the data:

$$L_{AR} = \mathop{\mathbb{E}}_{\mathbf{x}\sim X}[-\log p(\mathbf{x})], \tag{7}$$

where $p(\mathbf{x})$ is the probability density of the data of images, which can be modeled as:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_{\pi_i}|x_{\pi_1}, \cdots, x_{\pi_{i-1}}, \theta). \tag{8}$$

Here, the identity permutation $\pi_i = i$ is adopted for $1 \leqslant i \leqslant n$, which is also known as raster order. Chen *et al.* also considered the BERT objective, which samples a sub-sequence $M \subset [1, n]$ such that each index i independently has probability 0.15 of appearing in M. M is called the BERT mask, and the model is trained by minimizing the negative log-likelihood of the "masked" elements x_M conditioned on the "unmasked" ones $x_{[1,n]\backslash M}$:

$$L_{BERT} = \mathop{\mathbb{E}}_{\mathbf{x}\sim X} \mathop{\mathbb{E}}_{M} \sum_{i\in M}[-\log p(x_i|x_{[1,n]\backslash M})]. \tag{9}$$

During the pre-training stage, they pick either L_{AR} or L_{BERT} and minimize the loss over the pre-training dataset.

GPT-2 [109] formulation of the transformer decoder block is used. To ensure proper conditioning when training the AR objective, Chen *et al.* apply the standard upper triangular mask to the $n \times n$ matrix of attention logits. No attention logit masking is required when the BERT objective is used: Chen *et al.* zero out the positions after the content embeddings are applied to the input sequence. Following the final transformer layer, they apply a layer norm and learn a projection from the output to logits parameterizing the conditional distributions at each sequence element. When training BERT, they simply ignore the logits at unmasked positions.

During the fine-tuning stage, they average pool the output of the final layer normalization layer across the sequence dimension to extract a d-dimensional vector of features per example. They learn a projection from the pooled feature to class logits and use this projection to minimize a cross entropy loss. Practical applications offer empirical evidence that the joint objective of cross entropy loss and pretraining loss (L_{AR} or L_{BERT}) works even better.

iGPT and ViT are two pioneering works to apply transformer for visual tasks. The difference of iGPT and ViT-like models mainly lies on 3 aspects: 1) The input of iGPT is a sequence of color palettes by clustering pixels, while ViT uniformly divided the image into a number of local patches; 2) The architecture of iGPT is an encoder-decoder framework, while ViT only has transformer encoder; 3) iGPT utilizes auto-regressive self-supervised loss for training, while ViT is trained by supervised image classification task.

Contrastive Learning Based Approach. Currently, contrastive learning is the most popular manner of self-supervised learning for computer vision. Contrastive learning has been applied on vision transformer for unsupervised pretraining [31], [110], [111].

Chen *et al.* [31] investigate the effects of several fundamental components for training self-supervised ViT. The authors observe that instability is a major issue that degrades accuracy, and these results are indeed partial failure and they can be improved when training is made more stable.

They introduce a "MoCo v3" framework, which is an incremental improvement of MoCo [112]. Specifically, the authors take two crops for each image under random data augmentation. They are encodes by two encoders, f_q and f_k, with output vectors $\mathbf{q}$ and $\mathbf{k}$. Intuitively, $\mathbf{q}$ behaves like a "query" and the goal of learning is to retrieve the corresponding "key". This is formulated as minimizing a contrastive loss function, which can be written as:

$$\mathcal{L}_q = -\log \frac{\exp(\mathbf{q}\cdot\mathbf{k}^+/\tau)}{\exp(\mathbf{q}\cdot\mathbf{k}^+/\tau) + \sum_{\mathbf{k}^-}\exp(\mathbf{q}\cdot\mathbf{k}^-/\tau)}. \qquad (10)$$

Here $\mathbf{k}^+$ is f_k's output on the same image as $\mathbf{q}$, known as $\mathbf{q}$'s positive sample. The set $\mathbf{k}^-$ consists of f_k's outputs from other images, known as $\mathbf{q}$'s negative samples. τ is a temperature hyper-parameter for l_2-normalized $\mathbf{q}$, $\mathbf{k}$. MoCo v3 uses the keys that naturally co-exist in the same batch and abandon the memory queue, which they find has diminishing gain if the batch is sufficiently large (*e.g.*, 4096). With this simplification, the contrastive loss can be implemented in a simple way. The encoder f_q consists of a backbone (*e.g.*, ViT), a projection head and an extra prediction head; while the encoder f_k has the backbone and projection head, but not the prediction head. f_k is updated by the moving-average of f_q, excluding the prediction head.

MoCo v3 shows that the instability is a major issue of training the self-supervised ViT, thus they describe a simple trick that can improve the stability in various cases of the experiments. They observe that it is not necessary to train the patch projection layer. For the standard ViT patch size, the patch projection matrix is complete or over-complete. And in this case, random projection should be sufficient to preserve the information of the original patches. However, the trick alleviates the issue, but does not solve it. The model can still be unstable if the learning rate is too big and the first layer is unlikely the essential reason for the instability.

3.1.4 *Discussions*

All of the components of vision transformer including multi-head self-attention, multi-layer perceptron, shortcut connection, layer normalization, positional encoding and network topology, play key roles in visual recognition. As stated above, a number of works have been proposed to improve the effectiveness and efficiency of vision transformer. From the results in Figure 6, we can see that combining CNN and transformer achieve the better performance, indicating their complementation to each other through local connection and global connection. Further investigation on backbone networks can lead to the improvement for the whole vision community. As for the self-supervised representation learning for vision transformer, we still need to make effort to pursue the success of large-scale pretraining in the filed of NLP.

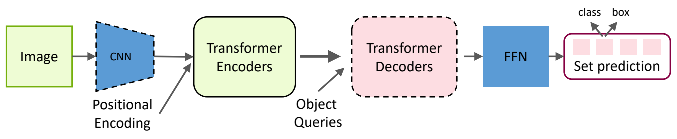
3.2 High/Mid-level Vision

Recently there has been growing interest in using transformer for high/mid-level computer vision tasks, such as object detection [16], [17], [113], [114], [115], lane detection [116], segmentation [33], [25], [18] and pose estimation [34], [35], [36], [117]. We review these methods in this section.

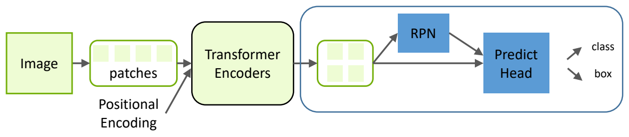
3.2.1 *Generic Object Detection*

Traditional object detectors are mainly built upon CNNs, but transformer-based object detection has gained significant interest recently due to its advantageous capability.

Some object detection methods have attempted to use transformer's self-attention mechanism and then enhance the specific modules for modern detectors, such as feature fusion module [118] and prediction head [119]. We discuss this in the supplemental material. Transformer-based object detection methods are broadly categorized into two groups: transformer-based set prediction methods [16], [17], [120], [121], [122] and transformer-based backbone methods [113], [115], as shown in Fig. 7. Transformer-based methods have shown strong performance compared with CNN-based detectors, in terms of both accuracy and running speed. Table 3 shows the detection results for different transformer-based object detectors mentioned earlier on the COCO 2012 val set.



(a) Transformer-based set prediction for detection



(b) Transformer-based backbone for detection

Fig. 7: General framework of transformer-based object detection.

Transformer-based Set Prediction for Detection. As a pioneer for transformer-based detection method, the detection transformer (DETR) proposed by Carion *et al.* [16] redesigns the framework of object detection. DETR, a simple and fully end-to-end object detector, treats the object detection task as an intuitive set prediction problem, eliminating traditional hand-crafted components such as anchor generation and non-maximum suppression (NMS) post-processing. As shown in Fig. 8, DETR starts with a CNN backbone to extract features from the input image. To supplement the image features with position information, fixed positional encodings are added to the flattened features before the features are fed into the encoder-decoder transformer. The decoder consumes the embeddings from the encoder along with N learned positional encodings (object queries), and produces N output embeddings. Here N is a predefined parameter and typically larger than the number of objects in an image. Simple feed-forward networks (FFNs) are used to compute the final predictions, which include the bounding box coordinates and class labels to indicate the specific class of object (or to indicate that no object exists). Unlike the original transformer, which computes predictions sequentially, DETR decodes N objects in parallel. DETR employs a bipartite matching algorithm to assign the predicted and ground-truth objects. As shown in Eq. 11, the Hungarian loss is exploited to compute the loss function for all matched pairs of objects.

$$\mathcal{L}_{\text{Hungarian}}(y,\hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}}(i)) \right] , \qquad (11)$$

where $\hat{\sigma}$ is the optimal assignment, c_i and $\hat{p}_{\hat{\sigma}(i)}(c_i)$ are the target class label and predicted label, respectively, and b_i and $\hat{b}_{\hat{\sigma}}(i)$ are the ground truth and predicted bounding box, $y = \{(c_i, b_i)\}$ and $\hat{y}$ are the ground truth and prediction of objects, respectively. DETR shows impressive performance on object detection, delivering comparable accuracy and speed with the popular and well-established Faster R-CNN [13] baseline on COCO benchmark.

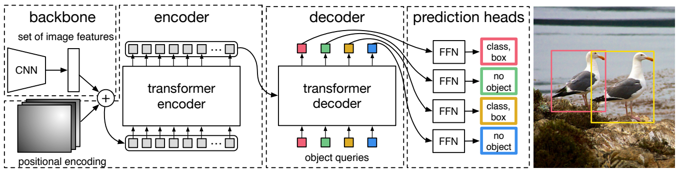


Fig. 8: The overall architecture of DETR (image from [16]).

DETR is a new design for the object detection framework based on transformer and empowers the community to develop fully end-to-end detectors. However, the vanilla DETR poses several challenges, specifically, longer training schedule and poor performance for small objects. To address these challenges, Zhu *et al.* [17] proposed Deformable DETR, which has become a popular method that significantly improves the detection performance. The deformable attention module attends to a small set of key positions around a reference point rather than looking at all spatial locations on image feature maps as performed by the original multi-head attention mechanism in transformer. This approach significantly reduces the computational complexity and brings benefits in terms of fast convergence. More importantly, the deformable attention module can be easily applied for fusing multi-scale features. Deformable DETR achieves better performance than DETR with $10\times$ less training cost and $1.6\times$ faster inference speed. And by using an iterative bounding box refinement method and two-stage scheme, Deformable DETR can further improve the detection performance.

There are also several methods to deal with the slow convergence problem of the original DETR. For example, Sun *et al.* [120] investigated why the DETR model has slow convergence and discovered that this is mainly due to the cross-attention module in the transformer decoder. To address this issue, an encoder-only version of DETR is proposed, achieving considerable improvement in terms of detection accuracy and training convergence. In addition, a new bipartite matching scheme is designed for greater training stability and faster convergence and two transformer-based set prediction models, *i.e.* TSP-FCOS and TSP-RCNN, are proposed to improve encoder-only DETR with feature pyramids. These new models achieve better performance compared with the original DETR model. Gao *et al.* [123] proposed the Spatially Modulated Co-Attention (SMCA) mechanism to accelerate the convergence by constraining co-attention responses to be high near initially estimated bounding box locations. By integrating the proposed SMCA module into DETR, similar mAP could be obtained with about $10\times$ less training epochs under comparable inference cost.

Given the high computation complexity associated with DETR, Zheng *et al.* [121] proposed an Adaptive Clustering Transformer (ACT) to reduce the computation cost of pre-trained DETR. ACT adaptively clusters the query features using a locality sensitivity hashing (LSH) method and broadcasts the attention output to the queries represented by the selected prototypes. ACT is used to replace the self-attention module of the pre-trained DETR model without requiring any re-training. This approach significantly reduces the computational cost while the accuracy slides slightly. The performance drop can be further reduced by utilizing a multi-task knowledge distillation (MTKD) method, which exploits the original transformer to distill the ACT module with a few epochs of fine-tuning. Yao *et al.* [124] pointed out that the random initialization in DETR is the main reason for the requirement of multiple decoder layers and slow convergence. To this end, they proposed the Efficient DETR to incorporate the dense prior into the detection pipeline via an additional region proposal network. The better initialization enables them to use only one decoder layers instead of six layers to achieve competitive performance with a more compact network.

Transformer-based Backbone for Detection. Unlike DETR which redesigns object detection as a set prediction tasks via transformer, Beal *et al.* [113] proposed to utilize transformer as a backbone for common detection frameworks such as Faster R-CNN [13]. The input image is divided into several patches and fed into a vision transformer, whose output embedding features are reorganized according to spatial information before passing through a detection head for the final results. A massive pre-training transformer backbone could bring benefits to the proposed ViT-FRCNN. There are also quite a few methods to explore versatile vision transformer backbone design [29], [72], [60], [62] and transfer these backbones to traditional detection frameworks like RetinaNet [127] and Cascade R-CNN [128]. For example, Swin Transformer [60] obtains about 4 box AP gains over ResNet-50 backbone with similar FLOPs for various detection frameworks.

Pre-training for Transformer-based Object Detection. Inspired by the pre-training transformer scheme in NLP, several methods have been proposed to explore different pre-training scheme for transformer-based object detection [32], [126], [129]. Dai *et al.* [32] proposed unsupervised pre-training for object detection (UP-DETR). Specifically, a novel unsupervised pretext task named random query patch detection is proposed to pre-train the DETR model. With this unsupervised pre-training scheme, UP-DETR significantly improves the detection accuracy on a relatively small dataset (PASCAL VOC). On the COCO benchmark with sufficient training data, UP-DETR still outperforms DETR, demonstrating the effectiveness of the unsupervised pre-training scheme.

Fang *et al.* [126] explored how to transfer the pure ViT structure that is pre-trained on ImageNet to the more challenging object detection task and proposed the YOLOS detector. To cope with the object detection task, the proposed YOLOS first drops the classification tokens in ViT and appends learnable detection tokens. Besides, the bipartite matching loss is utilized to perform set prediction for objects. With this simple pre-training scheme on ImageNet dataset, the proposed YOLOS shows competitive performance for object detection on COCO benchmark.

TABLE 3: Comparison of different transformer-based object detectors on COCO 2017 val set. Running speed (FPS) is evaluated on an NVIDIA Tesla V100 GPU as reported in [17]. †Estimated speed according to the reported number in the paper. ‡ViT backbone is pre-trained on ImageNet-21k. *ViT backbone is pre-trained on an private dataset with 1.3 billion images.

Method	Epochs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	#Params (M)	GFLOPs	FPS
CNN based										
FCOS [125]	36	41.0	59.8	44.1	26.2	44.6	52.2	-	177	23†
Faster R-CNN + FPN [13]	109	42.0	62.1	45.5	26.6	45.4	53.4	42	180	26
CNN Backbone + Transformer Head										
DETR [16]	500	42.0	62.4	44.2	20.5	45.8	61.1	41	86	28
DETR-DC5 [16]	500	43.3	63.1	45.9	22.5	47.3	61.1	41	187	12
Deformable DETR [17]	50	46.2	65.2	50.0	28.8	49.2	61.7	40	173	19
TSP-FCOS [120]	36	43.1	62.3	47.0	26.6	46.8	55.9	-	189	20†
TSP-RCNN [120]	96	45.0	64.5	49.6	29.7	47.7	58.0	-	188	15†
ACT+MKKD (L=32) [121]	-	43.1	-	-	61.4	47.1	22.2	-	169	14†
SMCA [123]	108	45.6	65.5	49.1	25.9	49.3	62.6	-	-	-
Efficient DETR [124]	36	45.1	63.1	49.1	28.3	48.4	59.0	35	210	-
UP-DETR [32]	150	40.5	60.8	42.6	19.0	44.4	60.0	41	-	-
UP-DETR [32]	300	42.8	63.0	45.3	20.8	47.1	61.7	41	-	-
Transformer Backbone + CNN Head										
ViT-B/16-FRCNN‡ [113]	21	36.6	56.3	39.3	17.4	40.0	55.5	-	-	-
ViT-B/16-FRCNN* [113]	21	37.8	57.4	40.1	17.8	41.4	57.3	-	-	-
PVT-Small+RetinaNet [72]	12	40.4	61.3	43.0	25.0	42.9	55.7	34.2	118	-
Twins-SVT-S+RetinaNet [62]	12	43.0	64.2	46.3	28.0	46.4	57.5	34.3	104	-
Swin-T+RetinaNet [60]	12	41.5	62.1	44.2	25.1	44.9	55.5	38.5	118	-
Swin-T+ATSS [60]	36	47.2	66.5	51.3	-	-	-	36	215	-
Pure Transformer based										
PVT-Small+DETR [72]	50	34.7	55.7	35.4	12.0	36.4	56.7	40	-	-
TNT-S+DETR [29]	50	38.2	58.9	39.4	15.5	41.1	58.8	39	-	-
YOLOS-Ti [126]	300	30.0	-	-	-	-	-	6.5	21	-
YOLOS-S [126]	150	37.6	57.6	39.2	15.9	40.2	57.3	28	179	-
YOLOS-B [126]	150	42.0	62.2	44.5	19.5	45.3	62.1	127	537	-

3.2.2 Segmentation

Segmentation is an important topic in computer vision community, which broadly includes panoptic segmentation, instance segmentation and semantic segmentation *etc*. Vision transformer has also shown impressive potential on the field of segmentation.

Transformer for Panoptic Segmentation. DETR [16] can be naturally extended for panoptic segmentation tasks and achieve competitive results by appending a mask head on the decoder. Wang *et al.* [25] proposed Max-DeepLab to directly predict panoptic segmentation results with a mask transformer, without involving surrogate sub-tasks such as box detection. Similar to DETR, Max-DeepLab streamlines the panoptic segmentation tasks in an end-to-end fashion and directly predicts a set of non-overlapping masks and corresponding labels. Model training is performed using a panoptic quality (PQ) style loss, but unlike prior methods that stack a transformer on top of a CNN backbone, Max-DeepLab adopts a dual-path framework that facilitates combining the CNN and transformer.

Transformer for Instance Segmentation. VisTR, a transformer-based video instance segmentation model, was proposed by Wang *et al.* [33] to produce instance prediction results from a sequence of input images. A strategy for matching instance sequence is proposed to assign the predictions with ground truths. In order to obtain the mask sequence for each instance, VisTR utilizes the instance sequence segmentation module to accumulate the mask features from multiple frames and segment the mask sequence with a 3D CNN. Hu *et al.* [130] proposed an instance segmentation Transformer (ISTR) to predict low-dimensional mask embeddings, and match them with ground truth for the set loss. ISTR conducted detection and segmentation with a recurrent refinement strategy which is different from the existing top-down and bottom-up frameworks. Yang *et al.* [131] investigated how to realize better and more efficient embedding learning to tackle the semi-supervised video object segmentation under challenging multi-object scenarios. Some papers such as [132], [133] also discussed using Transformer to deal with segmentation task.

Transformer for Semantic Segmentation. Zheng *et al.* [18] proposed a transformer-based semantic segmentation network (SETR). SETR utilizes an encoder similar to ViT [15] as the encoder to extract features from an input image. A multi-level feature aggregation module is adopted for performing pixel-wise segmentation. Strudel *et al.* [134] introduced *Segmenter* which relies on the output embedding corresponding to image patches and obtains class labels with a point-wise linear decoder or a mask transformer decoder. Xie *et al.* [135] proposed a simple, efficient yet powerful semantic segmentation framework which unifies Transformers with lightweight multilayer perception (MLP) decoders, which outputs multiscale features and avoids complex decoders.

Transformer for Medical Image Segmentation. Cao *et al.* [30] proposed an Unet-like pure Transformer for medical image segmentation, by feeding the tokenized image patches into the Transformer-based U-shaped Encoder-Decoder architecture with skip-connections for local-global semantic feature learning. Valanarasu *et al.* [136] explored transformer-based solutions and study the feasibility of using transformer-based network architectures for medical image segmentation tasks and proposed a Gated Axial-Attention model which extends the existing architectures by introducing an additional control mechanism in the self-attention module. Cell-DETR [137], based on the DETR panoptic segmentation model, is an attempt to use transformer for cell instance segmentation. It adds skip connections that bridge features between the backbone CNN and the CNN decoder in the segmentation head in order to enhance feature fusion. Cell-DETR achieves state-of-the-art performance for cell instance segmentation from microscopy imagery.

3.2.3 Pose Estimation

Human pose and hand pose estimation are foundational topics that have attracted significant interest from the research community. Articulated pose estimation is akin to a structured prediction task, aiming to predict the joint coordinates or mesh vertices from input RGB/D images. Here we discuss some methods [34], [35], [36], [117] that explore how to utilize transformer for modeling the global structure information of human poses and hand poses.

Transformer for Hand Pose Estimation. Huang *et al.* [34] proposed a transformer based network for 3D hand pose estimation from point sets. The encoder first utilizes a PointNet [138] to extract point-wise features from input point clouds and then adopts standard multi-head self-attention module to produce embeddings. In order to expose more global pose-related information to the decoder, a feature extractor such as PointNet++ [139] is used to extract hand joint-wise features, which are then fed into the decoder as positional encodings. Similarly, Huang *et al.* [35] proposed HOT-Net (short for hand-object transformer network) for 3D hand-object pose estimation. Unlike the preceding method which employs transformer to directly predict 3D hand pose from input point clouds, HOT-Net uses a ResNet to generate initial 2D hand-object pose and then feeds it into a transformer to predict the 3D hand-object pose. A spectral graph convolution network is therefore used to extract input embeddings for the encoder. Hampali *et al.* [140] proposed to estimate the 3D poses of two hands given a single color image. Specifically, appearance and spatial encodings of a set of potential 2D locations for the joints of both hands were inputted to a transformer, and the attention mechanisms were used to sort out the correct configuration of the joints and outputted the 3D poses of both hands.

Transformer for Human Pose Estimation. Lin *et al.* [36] proposed a mesh transformer (METRO) for predicting 3D human pose and mesh from a single RGB image. METRO extracts image features via a CNN and then perform position encoding by concatenating a template human mesh to the image feature. A multi-layer transformer encoder with progressive dimensionality reduction is proposed to gradually reduce the embedding dimensions and finally produce 3D coordinates of human joint and mesh vertices. To encourage the learning of non-local relationships between human joints, METRO randomly mask some input queries during training. Yang *et al.* [117] constructed an explainable model named TransPose based on Transformer architecture and low-level convolutional blocks. The attention layers built in Transformer can capture long-range spatial relationships between keypoints and explain what dependencies the predicted keypoints locations highly rely on. Li *et al.* [141] proposed a novel approach based on Token representation for human Pose estimation (TokenPose). Each keypoint was explicitly embedded as a token to simultaneously learn constraint relationships and appearance cues from images. Mao *et al.* [142] proposed a human pose estimation framework that solved the task in the regression-based fashion. They formulated the pose estimation task into a sequence prediction problem and solve it by transformers, which bypass the drawbacks of the heatmap-based pose estimator. Jiang *et al.* [143] proposed a novel transformer based network that can learn a distribution over both pose and motion in an unsupervised fashion rather than tracking body parts and trying to temporally smooth them. The method overcame inaccuracies in detection and corrected partial or entire skeleton corruption. Hao *et al.* [144] proposed to personalize a human pose estimator given a set of test images of a person without using any manual annotations. The method adapted the pose estimator during test time to exploit person-specific information, and used a Transformer model to build a transformation between the self-supervised keypoints and the supervised keypoints.

3.2.4 *Other Tasks*

There are also quite a lot different high/mid-level vision tasks that have explored the usage of vision transformer for better performance. We briefly review several tasks below.

Pedestrian Detection. Because the distribution of objects is very dense in occlusion and crowd scenes, additional analysis and adaptation are often required when common detection networks are applied to pedestrian detection tasks. Lin *et al.* [145] revealed that sparse uniform queries and a weak attention field in the decoder result in performance degradation when directly applying DETR or Deformable DETR to pedestrian detection tasks. To alleviate these drawbacks, the authors proposes Pedestrian End-to-end Detector (PED), which employs a new decoder called Dense Queries and Rectified Attention field (DQRF) to support dense queries and alleviate the noisy or narrow attention field of the queries. They also proposed V-Match, which achieves additional performance improvements by fully leveraging visible annotations.

Lane Detection. Based on PolyLaneNet [146], Liu *et al.* [116] proposed a method called LSTR, which improves performance of curve lane detection by learning the global context with a transformer network. Similar to PolyLaneNet, LSTR regards lane detection as a task of fitting lanes with polynomials and uses neural networks to predict the parameters of polynomials. To capture slender structures for lanes and the global context, LSTR introduces a transformer network into the architecture. This enables processing of low-level features extracted by CNNs. In addition, LSTR uses Hungarian loss to optimize network parameters. As demonstrated in [116], LSTR outperforms PolyLaneNet, with 2.82% higher accuracy and $3.65\times$ higher FPS using 5-times fewer parameters. The combination of a transformer network, CNN and Hungarian Loss culminates in a lane detection framework that is precise, fast, and tiny. Considering that the entire lane line generally has an elongated shape and long-range, Liu *et al.* [147] utilized a transformer encoder structure for more efficient context feature extraction. This transformer encoder structure improves the detection of the proposal points a lot, which rely on contextual features and global information, especially in the case where the backbone network is a small model.

Scene Graph. Scene graph is a structured representation of a scene that can clearly express the objects, attributes, and relationships between objects in the scene [148]. To generate scene graph, most of existing methods first extract image-based object representations and then do message propagation between them. Graph R-CNN [149] utilizes self-attention to integrate contextual information from neighboring nodes in the graph. Recently, Sharifzadeh *et al.* [150] employed transformers over the extracted object embedding. Sharifzadeh *et al.* [151] proposed a new pipeline called *Texema* and employed a pre-trained Text-to-Text Transfer Transformer (T5) [152] to create structured graphs from textual input and utilized them to improve the relational reasoning module. The T5 model enables *Texema* to utilize the knowledge in texts.

Tracking. Some researchers also explored to use transformer encoder-decoder architecture in template-based discriminative trackers, such as TMT [153], TrTr [154] and TransT [155]. All these work use a Siamese-like tracking pipeline to do video object tracking and utilize the encoder-decoder network to replace explicit cross-correlation operation for global and rich contextual inter-dependencies. Specifically, the transformer encoder and decoder are assigned to the template branch and the searching branch, respectively. In addition, Sun *et al.* proposed TransTrack [156], which is an online joint-detection-and-tracking pipeline. It utilizes the query-key mechanism to track pre-existing objects and introduces a set of learned object queries into the pipeline to detect new-coming objects. The proposed TransTrack achieves 74.5% and 64.5% MOTA on MOT17 and MOT20 benchmark.

Re-Identification. He *et al.* [157] proposed TransReID to investigate the application of pure transformers in the field of object re-identification (ReID). While introducing transformer network into object ReID, TransReID slices with overlap to reserve local neighboring structures around the patches and introduces 2D bilinear interpolation to help handle any given input resolution. With the transformer module and the loss function, a strong baseline was proposed to achieve comparable performance with CNN-based frameworks. Moreover, The jigsaw patch module (JPM) was designed to facilitate perturbation-invariant and robust feature representation of objects and the side information embeddings (SIE) was introduced to encode side information. The final framework TransReID achieves state-of-the-art performance on both person and vehicle ReID benchmarks. Both Liu *et al.* [158] and Zhang *et al.* [159] provided solutions for introducing transformer network into video-based person Re-ID. And similarly, both of the them utilized separated transformer networks to refine spatial and temporal features, and then utilized a cross view transformer to aggregate multi-view features.

Point Cloud Learning. A number of other works exploring transformer architecture for point cloud learning [160], [161], [162] have also emerged recently. For example, Guo *et al.* [161] proposed a novel framework that replaces the original self-attention module with a more suitable offset-attention module, which includes implicit Laplace operator and normalization refinement. In addition, Zhao *et al.* [162] designed a novel transformer architecture called Point Transformer. The proposed self-attention layer is invariant to the permutation of the point set, making it suitable for point set processing tasks. Point Transformer shows strong performance for semantic segmentation task from 3D point clouds.

3.2.5 Discussions

As discussed in the preceding sections, transformers have shown strong performance on several high-level tasks, including detection, segmentation and pose estimation. The key issues that need to be resolved before transformer can be adopted for high-level tasks relate to input embedding, position encoding, and prediction loss. Some methods propose improving the self-attention module from different perspectives, for example, deformable attention [17], adaptive clustering [121] and point transformer [162]. Nevertheless, exploration into the use of transformers for high-level vision tasks is still in the preliminary stages and so further research may prove beneficial. For example, is it necessary to use feature extraction modules such as CNN and PointNet before transformer for potential better performance? How can vision transformer be fully leveraged using large scale pre-training datasets as BERT and GPT-3 do in the NLP field? And is it possible to pre-train a single transformer model and fine-tune it for different downstream tasks with only a few epochs of fine-tuning? How to design more powerful architecture by incorporating prior knowledge of the specific tasks? Several prior works have performed preliminary discussions for the aforementioned topics and We hope more further research effort is conducted into exploring more powerful transformers for high-level vision.

3.3 Low-level Vision

Few works apply transformers on low-level vision fields, such as image super-resolution and generation. These tasks often take images as outputs (*e.g.*, high-resolution or denoised images), which is more challenging than high-level vision tasks such as classification, segmentation, and detection, whose outputs are labels or boxes.

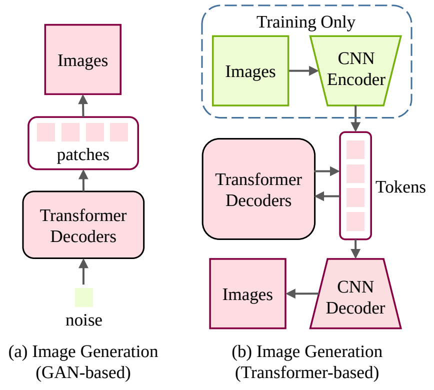


Fig. 9: A generic framework for transformer in image generation.

3.3.1 Image Generation

An simple yet effective to apply transformer model to the image generation task is to directly change the architectures from CNNs to transformers, as shown in Figure 9 (a). Jiang *et al.* [38] proposed TransGAN, which build GAN using the transformer architecture. Since the it is difficult to generate high-resolution images pixel-wise, a memory-friendly generator is utilized by gradually increasing the feature map resolution at different stages. Correspondingly, a multi-scale discriminator is designed to handle the varying size of inputs in different stages. Various training recipes are introduced including grid self-attention, data augmentation, relative position encoding and modified normalization to stabilize the training and improve its performance. Experiments on various benchmark datasets demonstrate the effectiveness and potential of the transformer-based GAN model in image generation tasks. Kwonjoon Lee *et al.* [163] proposed ViTGAN, which introduce several technique to both generator and discriminator to stabilize the training procedure and convergence. Euclidean distance is introduced for the self-attention module to enforce the Lipschitzness of transformer discriminator. Self-modulated layernorm and implicit neural representation are proposed to enhance the training for the generator. As a result, ViTGAN is the first work to demonstrate transformer-based GANs can achieve comparable performance to state-of-the-art CNN-based GANs.

Parmar *et al.* [27] proposed Image Transformer, taking the first step toward generalizing the transformer model to formulate image translation and generation tasks in an auto-regressive manner. Image Transformer consists of two parts: an encoder for extracting image representation and a decoder to generate pixels. For each pixel with value $0 - 255$, a $256 \times d$ dimensional embedding is learned for encoding each value into a d dimensional vector, which is fed into the encoder as input. The encoder and decoder adopt the same architecture as that in [9]. Each output pixel q' is generated by calculating self-attention between the input pixel q and previously generated pixels $m_1, m_2, ...$ with position embedding $p_1, p_2, \ldots$. For image-conditioned generation, such as super-resolution and inpainting, an encoder-decoder architecture is used, where the encoder's input is the low-resolution or corrupted images. For unconditional and class-conditional generation (*i.e.*, noise to image), only the decoder is used for inputting noise vectors. Because the decoder's input is the previously generated pixels (involving high computation cost when producing high-resolution images), a local self-attention scheme is proposed. This scheme uses only the closest generated pixels as input for the decoder, enabling Image Transformer to achieve performance on par with CNN-based models for image generation and translation tasks, demonstrating the effectiveness of transformer-based models on low-level vision tasks.

Since it is difficult to directly generate high-resolution images by transformer models, Esser *et al.* [37] proposed Taming Transformer. Taming Transformer consists of two parts: a VQGAN and a transformer. VQGAN is a variant of VQVAE [164], which uses a discriminator and perceptual loss to improve the visual quality. Through VQGAN, the image can be represented by a series of context-rich discrete vectors and therefore these vectors can be easily predicted by a transformer model through an autoregression way. The transformer model can learn the long-range interactions for generating high-resolution images. As a result, the proposed Taming Transformer achieves state-of-the-art results on a wide variety of image synthesis tasks.

Besides image generation, DALL·E [41] proposed the transformer model for text-to-image generation, which synthesizes images according to the given captions. The whole framework consists of two stages. In the first stage, a discrete VAE is utilized to learn the visual codebook. In the second stage, the text is decoded by BPE-encode and the corresponding image is decoded by dVAE learned in the first stage. Then an autoregression transformer is used to learn the prior between the encoded text and image. During the inference procedure, tokens of images are predicted by the transformer and decoded by the learned decoder. The CLIP model [40] is introduced to rank generated samples. Experiments on text-to-image generation task demonstrate the powerful ability of the proposed model. Note that our survey mainly focus on pure vision tasks, we do not include the framework of DALL·E in Figure 9.

3.3.2 *Image Processing*

A number of recent works eschew using each pixel as the input for transformer models and instead use patches (set of pixels) as input. For example, Yang *et al.* [39] proposed Texture Transformer Network for Image Super-Resolution (TTSR), using the transformer architecture in the reference-based image super-resolution problem. It aims to transfer relevant textures from reference images to low-resolution images. Taking a low-resolution image and reference image as the query $\mathbf{Q}$ and key $\mathbf{K}$, respectively, the relevance $r_{i,j}$ is calculated between each patch $\mathbf{q}_i$ in $\mathbf{Q}$ and $\mathbf{k}_i$ in $\mathbf{K}$ as:

$$r_{i,j} = \left\langle \frac{\mathbf{q}_i}{\|\mathbf{q}_i\|}, \frac{\mathbf{k}_i}{\|\mathbf{k}_i\|} \right\rangle. \tag{12}$$

A hard-attention module is proposed to select high-resolution features $\mathbf{V}$ according to the reference image, so that the low-resolution image can be matched by using the relevance. The hard-attention map is calculated as:

$$h_i = \arg\max_j r_{i,j} \tag{13}$$

The most relevant reference patch is $\mathbf{t}_i = \mathbf{v}_{h_i}$, where $\mathbf{t}_i$ in $\mathbf{T}$ is the transferred features. A soft-attention module is then used to transfer $\mathbf{V}$ to the low-resolution feature. The transferred features from the high-resolution texture image and the low-resolution feature are used to generate the output features of the low-resolution image. By leveraging the transformer-based architecture, TTSR can successfully transfer texture information from high-resolution reference images to low-resolution images in super-resolution tasks.

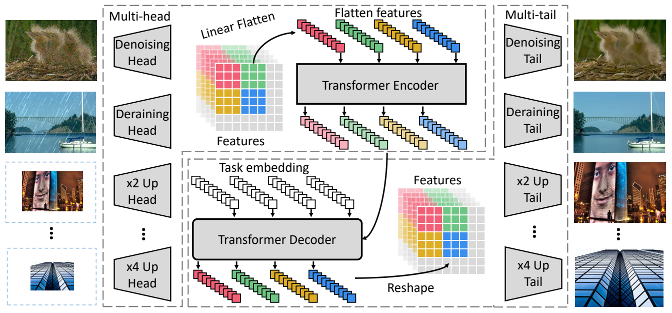


Fig. 10: Diagram of IPT architecture (image from [19]).

Different from the preceding methods that use transformer models on single tasks, Chen *et al.* [19] proposed Image Processing Transformer (IPT), which fully utilizes the advantages of transformers by using large pre-training datasets. It achieves state-of-the-art performance in several image processing tasks, including super-resolution, denoising, and deraining. As shown in Figure 10, IPT consists of multiple heads, an encoder, a decoder, and multiple tails. The multi-head, multi-tail structure and task embeddings are introduced for different image processing tasks. The features are divided into patches, which are fed into the encoder-decoder architecture. Following this, the outputs are reshaped to features with the same size. Given the advantages of pre-training transformer models on large datasets, IPT uses the ImageNet dataset for pre-training. Specifically, images from this dataset are degraded by manually adding noise, rain streaks, or downsampling in order to generate corrupted images. The degraded images are used as inputs for IPT, while the original images are used as the optimization goal of the outputs. A self-supervised method is also introduced to enhance the generalization ability of the IPT model. Once the model is trained, it is fine-tuned on each task by using the corresponding head, tail, and task embedding. IPT largely achieves performance improvements on image processing tasks (*e.g.*, 2 dB in image denoising tasks), demonstrating the huge potential of applying transformer-based models to the field of low-level vision.

Besides single image generation, Wang *et al.* [165] proposed SceneFormer to utilize transformer in 3D indoor scene generation. By treating a scene as a sequence of objects, the transformer decoder can be used to predict series of objects and their location, category, and size. This has enabled SceneFormer to outperform conventional CNN-based methods in user studies.

It should be noted that iGPT [14] is pre-trained on an inpainting-like task. Since iGPT mainly focus on the fine-tuning performance on image classification tasks, we treat this work more like an attempt on image classification task using transformer than low-level vision tasks.

In conclusion, different to classification and detection tasks, the outputs of image generation and processing are images. Figure 11 illustrates using transformers in low-level vision. In image

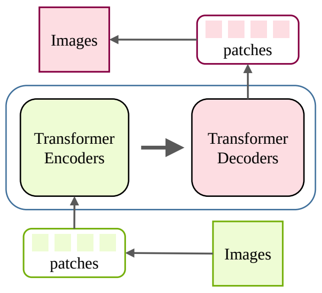


Fig. 11: A generic framework for transformer in image processing.

processing tasks, the images are first encoded into a sequence of tokens or patches and the transformer encoder uses the sequence as input, allowing the transformer decoder to successfully produce desired images. In image generation tasks, the GAN-based models directly learn a decoder to generated patches to outputting images through linear projection, while the transformer-based models train a auto-encoder to learn a codebook for images and use an auto-regression transformer model to predict the encoded tokens. A meaningful direction for future research would be designing a suitable architecture for different image processing tasks.

3.4 Video Processing

Transformer performs surprisingly well on sequence-based tasks and especially on NLP tasks. In computer vision (specifically, video tasks), spatial and temporal dimension information is favored, giving rise to the application of transformer in a number of video tasks, such as frame synthesis [166], action recognition [167], and video retrieval [168].

3.4.1 *High-level Video Processing*

Video Action Recognition. Video human action tasks, as the name suggests, involves identifying and localizing human actions in videos. Context (such as other people and objects) plays a critical role in recognizing human actions. Rohit *et al.* proposed the action transformer [167] to model the underlying relationship between the human of interest and the surrounding context. Specifically, the I3D [169] is used as the backbone to extract high-level feature maps. The features extracted (using RoI pooling) from intermediate feature maps are viewed as the query (Q), while the key (K) and values (V) are calculated from the intermediate features. A self-attention mechanism is applied to the three components, and it outputs the classification and regressions predictions. Lohit *et al.* [170] proposed an interpretable differentiable module, named temporal transformer network, to reduce the intra-class variance and increase the inter-class variance. In addition, Fayyaz and Gall proposed a temporal transformer [171] to perform action recognition tasks under weakly supervised settings. In addition to human action recognition, transformer has been utilized for group activity recognition [172]. Gavrilyuk *et al.* proposed an actor-transformer [173] architecture to learn the representation, using the static and dynamic representations generated by the 2D and 3D networks as input. The output of the transformer is the predicted activity.

Video Retrieval. The key to content-based video retrieval is to find the similarity between videos. Leveraging only the image-level of video-level features to overcome the associated challenges, Shao *et al.* [174] suggested using the transformer to model the long-range semantic dependency. They also introduced the supervised contrastive learning strategy to perform hard negative mining. The results of using this approach on benchmark datasets demonstrate its performance and speed advantages. In addition, Gabeur *et al.* [175] presented a multi-modal transformer to learn different cross-modal cues in order to represent videos.

Video Object Detection. To detect objects in a video, both global and local information is required. Chen *et al.* introduced the memory enhanced global-local aggregation (MEGA) [176] to capture more content. The representative features enhance the overall performance and address the *ineffective* and *insufficient* problems. Furthermore, Yin *et al.* [177] proposed a spatiotemporal transformer to aggregate spatial and temporal information. Together with another spatial feature encoding component, these two components perform well on 3D video object detection tasks.

Multi-task Learning. Untrimmed video usually contains many frames that are irrelevant to the target tasks. It is therefore crucial to mine the relevant information and discard the redundant information. To extract such information, Seong *et al.* proposed the video multi-task transformer network [178], which handles multi-task learning on untrimmed videos. For the CoVieW dataset, the tasks are scene recognition, action recognition and importance score prediction. Two pre-trained networks on ImageNet and Places365 extract the scene features and object features. The multi-task transformers are stacked to implement feature fusion, leveraging the class conversion matrix (CCM).

3.4.2 *Low-level Video Processing*

Frame/Video Synthesis. Frame synthesis tasks involve synthesizing the frames between two consecutive frames or after a frame sequence while video synthesis tasks involve synthesizing a video. Liu *et al.* proposed the ConvTransformer [166], which is comprised of five components: feature embedding, position encoding, encoder, query decoder, and the synthesis feed-forward network. Compared with LSTM based works, the ConvTransformer achieves superior results with a more parallelizable architecture. Another transformer-based approach was proposed by Schatz *et al.* [179], which uses a recurrent transformer network to synthetize human actions from novel views.

Video Inpainting. Video inpainting tasks involve completing any missing regions within a frame. This is challenging, as it requires information along the spatial and temporal dimensions to be merged. Zeng *et al.* proposed a spatial-temporal transformer network [28], which uses all the input frames as input and fills them in parallel. The spatial-temporal adversarial loss is used to optimize the transformer network.

3.4.3 *Discussions*

Compared to image, video has an extra dimension to encode the temporal information. Exploiting both spatial and temporal information helps to have a better understanding of a video. Thanks to the relationship modeling capability of transformer, video processing tasks have been improved by mining spatial and temporal information simultaneously. Nevertheless, due to the high complexity and much redundancy of video data, how to efficiently and accurately modeling both spatial and temporal relationships is still an open problem.

3.5 Multi-Modal Tasks

Owing to the success of transformer across text-based NLP tasks, many researches are keen to exploit its potential for processing multi-modal tasks (*e.g.*, video-text, image-text and audio-text).

One example of this is VideoBERT [180], which uses a CNN-based module to pre-process videos in order to obtain representation tokens. A transformer encoder is then trained on these tokens to learn the video-text representations for downstream tasks, such as video caption. Some other examples include VisualBERT [181] and VL-BERT [182], which adopt a single-stream unified transformer to capture visual elements and image-text relationship for downstream tasks such as visual question answering (VQA) and visual commonsense reasoning (VCR). In addition, several studies such as SpeechBERT [183] explore the possibility of encoding audio and text pairs with a transformer encoder to process auto-text tasks such as speech question answering (SQA).

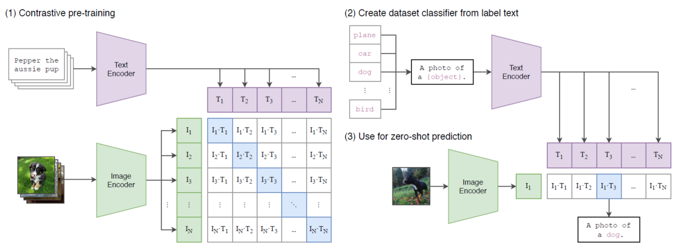

(1) Contrastive pre-training
Pepper the aussie pup
Text Encoder
T_1 T_2 T_3 ... T_N
Image Encoder
I_1 I_2 I_3 ⋮ I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$
(2) Create dataset classifier from label text
plane
car
dog
bird
A photo of a {object}.
Text Encoder
(3) Use for zero-shot prediction
Image Encoder
I_1
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
A photo of a dog.

Fig. 12: The framework of the CLIP (image from [40]).

Apart from the aforementioned pioneering multi-modal transformers, Contrastive Language-Image Pre-training (CLIP) [40] takes natural language as supervision to learn more efficient image representation. CLIP jointly trains a text encoder and an image encoder to predict the corresponding training text-image pairs. The text encoder of CLIP is a standard transformer with masked self-attention used to preserve the initialization ability of the pre-trained language models. For the image encoder, CLIP considers two types of architecture, ResNet and Vision Transformer. CLIP is trained on a new dataset containing 400 million (image, text) pairs collected from the Internet. More specifically, given a batch of N (image, text) pairs, CLIP learns both text and image embeddings jointly to maximize the cosine similarity of those N matched embeddings while minimize $N^2 - N$ incorrectly matched embeddings. On Zero-Shot transfer, CLIP demonstrates astonishing zero-shot classification performances, achieving 76.2% top-1 accuracy on ImageNet-1K dataset without using any ImageNet training labels. Concretely, at inference, the text encoder of CLIP first computes the feature embeddings of all ImageNet Labels and the image encoder then computes the embeddings of all images. By calculating the cosine similarity of text and image embeddings, the text-image pair with the highest score should be the image and its corresponding label. Further experiments on 30 various CV benchmarks show the zero-shot transfer ability of CLIP and the feature diversity learned by CLIP.

While CLIP maps images according to the description in text, another work DALL-E [41] synthesizes new images of categories described in an input text. Similar to GPT-3, DALL-E is a multi-modal transformer with 12 billion model parameters autoregressively trained on a dataset of 3.3 million text-image pairs. More specifically, to train DALL-E, a two-stage training procedure is used, where in stage 1, a discrete variational autoencoder is used to compress 256× 256 RGB images into 32×32 image tokens and then in stage 2, an autoregressive transformer is trained to model the joint distribution over the image and text tokens. Experimental results show that DALL-E can generate images of various styles from scratch, including photorealistic imagery, cartoons and emoji or extend an existing image while still matching the description in the text. Subsequently, Ding *et al.* proposes CogView [42], which is a transformer with VQ-VAE tokenizer similar to DALL-E, but supports Chinese text input. They claim CogView outperforms DALL-E and previous GAN-bsed methods and also unlike DALL-E, CogView does not need an additional CLIP model to rerank the samples drawn from transformer, *i.e.* DALL-E.

Recently, a Unified Transformer (UniT) [43] model is proposed to cope with multi-modal multi-task learning, which can simultaneously handle multiple tasks across different domains, including object detection, natural language understanding and vision-language reasoning. Specifically, UniT has two transformer encoders to handle image and text inputs, respectively, and then the transformer decoder takes the single or concatenated encoder outputs according to the task modality. Finally, a task-specific prediction head is applied to the decoder outputs for different tasks. In the training stage, all tasks are jointly trained by randomly selecting a specific task within an iteration. The experiments show UniT achieves satisfactory performance on every task with a compact set of model parameters.

In conclusion, current transformer-based mutil-modal models demonstrates its architectural superiority for unifying data and tasks of various modalities, which demonstrates the potential of transformer to build a general-purpose intelligence agents able to cope with vast amount of applications. Future researches can be conducted in exploring the effective training or the extendability of multi-modal transformers.

3.6 Efficient Transformer

Although transformer models have achieved success in various tasks, their high requirements for memory and computing resources block their implementation on resource-limited devices such as mobile phones. In this section, we review the researches carried out into compressing and accelerating transformer models for efficient implementation. This includes including network pruning, low-rank decomposition, knowledge distillation, network quantization, and compact architecture design. Table 4 lists some representative works for compressing transformer-based models.

3.6.1 Pruning and Decomposition

In transformer based pre-trained models (*e.g.*, BERT), multiple attention operations are performed in parallel to independently model the relationship between different tokens [9], [10]. However, specific tasks do not require all heads to be used. For example, Michel *et al.* [44] presented empirical evidence that a large percentage of attention heads can be removed at test time without impacting performance significantly. The number of heads required varies across different layers — some layers may even require only one head. Considering the redundancy on attention heads, importance scores are defined to estimate the influence of each head on the final output in [44], and unimportant heads can be removed for efficient deployment. Dalvi *et al.* [184] analyzed the redundancy in pre-trained transformer models from two perspectives: general redundancy and task-specific redundancy. Following the lottery ticket hypothesis [185], Prasanna *et al.* [184] analyzed the lotteries in BERT and showed that good sub-networks also exist in transformer-based models, reducing both the FFN layers and attention heads in order to achieve high compression rates. For the vision transformer [15] which splits an image to multiple patches, Tang *et al.* [186] proposed to reduce patch calculation to accelerate the inference, and the redundant patches can be automatically discovered by considering their contributions to the effective output features. Zhu *et al.* [187] extended the network slimming approach [188] to vision transformers for reducing the dimensions of linear projections in both FFN and attention modules.

In addition to the width of transformer models, the depth (*i.e.*, the number of layers) can also be reduced to accelerate the inference process [198], [199]. Differing from the concept that different attention heads in transformer models can be computed in parallel, different layers have to be calculated sequentially because the input of the next layer depends on the output of previous layers. Fan *et al.* [198] proposed a layer-wisely dropping strategy to regularize the training of models, and then the whole layers are removed together at the test phase.

Beyond the pruning methods that directly discard modules in transformer models, matrix decomposition aims to approximate the large matrices with multiple small matrices based on the low-rank assumption. For example, Wang *et al.* [200] decomposed the standard matrix multiplication in transformer models, improving the inference efficiency.

3.6.2 Knowledge Distillation

Knowledge distillation aims to train student networks by transferring knowledge from large teacher networks [201], [202], [203]. Compared with teacher networks, student networks usually have thinner and shallower architectures, which are easier to be deployed on resource-limited resources. Both the output and intermediate features of neural networks can also be used to transfer effective information from teachers to students. Focused on transformer models, Mukherjee *et al.* [204] used the pre-trained BERT [10] as a teacher to guide the training of small models, leveraging large amounts of unlabeled data. Wang *et al.* [205] train the student networks to mimic the output of self-attention layers in the pre-trained teacher models. The dot-product between values is introduced as a new form of knowledge for guiding students. A teacher's assistant [206] is also introduced in [205], reducing the gap between large pre-trained transformer models and compact student networks, thereby facilitating the mimicking process. Due to the various types of layers in the transformer model (*i.e.*, self-attention layer, embedding layer, and prediction layers), Jiao *et al.* [45] design different objective functions to transfer knowledge from teachers to students. For example, the outputs of student models' embedding layers imitate those of teachers via MSE losses. For the vision transformer, Jia *et al.* [207] proposed a fine-grained manifold distillation method, which excavates effective knowledge through the relationship between images and the divided patches.

TABLE 4: List of representative compressed transformer-based models. The data of the Table is from [197].

Models	Compress Type	#Layer	Params	Speed Up
BERT$_{BASE}$ [10]	Baseline	12	110M	×1
ALBERT [189]	Decomposition	12	12M	×5.6
BERT-of-Theseus [190]	Architecture design	6	66M	×1.94
Q-BERT [191]	Quantization	12	-	-
Q8BERT [192]		12		
TinyBERT [45]	Distillation	4	14.5M	×9.4
DistilBERT [193]		6	6.6m	×1.63
BERT-PKD [194]		3∼6	45.7∼67M	×3.73∼1.64
MobileBERT [195]		24	25.3M	×4.0
PD [196]		6	67.5M	×2.0

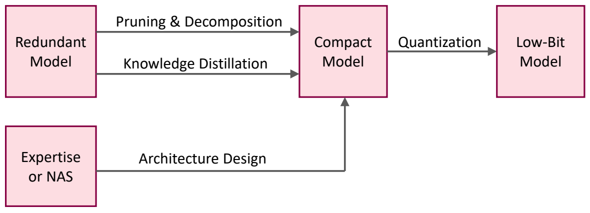


Fig. 13: Different methods for compressing transformers.

3.6.3 Quantization

Quantization aims to reduce the number of bits needed to represent network weight or intermediate features [208], [209]. Quantization methods for general neural networks have been discussed at length and achieve performance on par with the original networks [210], [211], [212]. Recently, there has been growing interest in how to specially quantize transformer models [213], [214]. For example, Shridhar *et al.* [215] suggested embedding the input into binary high-dimensional vectors, and then using the binary input representation to train the binary neural networks. Cheong *et al.* [216] represented the weights in the transformer models by low-bit (*e.g.*, 4-bit) representation. Zhao *et al.* [217] empirically investigated various quantization methods and showed that k-means quantization has a huge development potential. Aimed at machine translation tasks, Prato *et al.* [46] proposed a fully quantized transformer, which, as the paper claims, is the first 8-bit model not to suffer any loss in translation quality. Beside, Liu *et al.* [218] explored a post-training quantization scheme to reduce the memory storage and computational costs of vision transformers.

3.6.4 Compact Architecture Design

Beyond compressing pre-defined transformer models into smaller ones, some works attempt to design compact models directly [219], [47]. Jiang *et al.* [47] simplified the calculation of self-attention by proposing a new module — called span-based dynamic convolution — that combine the fully-connected layers and the convolutional layers. Interesting "hamburger" layers are proposed in [220], using matrix decomposition to substitute the original self-attention layers.Compared with standard self-attention operations, matrix decomposition can be calculated more efficiently while clearly reflecting the dependence between different tokens. The design of efficient transformer architectures can also be automated with neural architecture search (NAS) [221], [222], which automatically searches how to combine different components. For example, Su *et al.* [82] searched patch size and dimensions of linear projections and head number of attention modules to get an efficient vision transformer. Li *et al.* [223] explored a self-supervised search strategy to get a hybrid architecture composing of both convolutional modules and self-attention modules.

The self-attention operation in transformer models calculates the dot product between representations from different input tokens in a given sequence (patches in image recognition tasks [15]), whose complexity is $O(N)$, where N is the length of the sequence. Recently, there has been a targeted focus to reduce the complexity to $O(N)$ in large methods so that transformer models can scale to long sequences [224], [225], [226]. For example, Katharopoulos *et al.* [224] approximated self-attention as a linear dot-product of kernel feature maps and revealed the relationship between tokens via RNNs. Zaheer *et al.* [226] considered each token as a vertex in a graph and defined the inner product calculation between two tokens as an edge. Inspired by graph theories [227], [228], various sparse graph are combined to approximate the dense graph in transformer models, and can achieve $O(N)$ complexity.

Discussion. The preceding methods take different approaches in how they attempt to identify redundancy in transformer models (see Figure 13). Pruning and decomposition methods usually require pre-defined models with redundancy. Specifically, pruning focuses on reducing the number of components (*e.g.*, layers, heads) in transformer models while decomposition represents an original matrix with multiple small matrices. Compact models also can be directly designed either manually (requiring sufficient expertise) or automatically (*e.g.*, via NAS). The obtained compact models can be further represented with low-bits via quantization methods for efficient deployment on resource-limited devices.

4 Conclusions and Discussions

Transformer is becoming a hot topic in the field of computer vision due to its competitive performance and tremendous potential compared with CNNs. To discover and utilize the power of transformer, as summarized in this survey, a number of methods have been proposed in recent years. These methods show excellent performance on a wide range of visual tasks, including backbone, high/mid-level vision, low-level vision, and video processing. Nevertheless, the potential of transformer for computer vision has not yet been fully explored, meaning that several challenges still need to be resolved. In this section, we discuss these challenges and provide insights on the future prospects.

4.1 Challenges

Although researchers have proposed many transformer-based models to tackle computer vision tasks, these works are only the first steps in this field and still have much room for improvement. For example, the transformer architecture in ViT [15] follows the standard transformer for NLP [9], but an improved version specifically designed for CV remains to be explored. Moreover, it is necessary to apply transformer to more tasks other than those mentioned earlier.

The generalization and robustness of transformers for computer vision are also challenging. Compared with CNNs, pure transformers lack some inductive biases and rely heavily on massive datasets for large-scale training [15]. Consequently, the quality of data has a significant influence on the generalization and robustness of transformers. Although ViT shows exceptional performance on downstream image classification tasks such as CIFAR [229] and VTAB [230], directly applying the ViT backbone on object detection has failed to achieve better results than CNNs [113]. There is still a long way to go in order to better generalize pre-trained transformers on more generalized visual tasks. Practitioners concern the robustness of transformer (e.g. the vulnerability issue [231]). Although the robustness has been investigated in [232], [233], [234], it is still an open problem waiting to be solved.

Although numerous works have explained the use of transformers in NLP [235], [236], it remains a challenging subject to clearly explain why transformer works well on visual tasks. The inductive biases, including translation equivariance and locality, are attributed to CNN's success, but transformer lacks any inductive bias. The current literature usually analyzes the effect in an intuitive way [15], [237]. For example, Dosovitskiy *et al.* [15] claim that large-scale training can surpass inductive bias. Position embeddings are added into image patches to retain positional information, which is important in computer vision tasks. Inspired by the heavy parameter usage in transformers, over-parameterization [238], [239] may be a potential point to the interpretability of vision transformers.

Last but not least, developing efficient transformer models for CV remains an open problem. Transformer models are usually huge and computationally expensive. For example, the base ViT model [15] requires 18 billion FLOPs to process an image. In contrast, the lightweight CNN model GhostNet [240], [241] can achieve similar performance with only about 600 million FLOPs. Although several methods have been proposed to compress transformer, they remain highly complex. And these methods, which were originally designed for NLP, may not be suitable for CV. Consequently, efficient transformer models are urgently needed so that vision transformer can be deployed on resource-limited devices.

4.2 Future Prospects

In order to drive the development of vision transformers, we provide several potential directions for future study.

One direction is the effectiveness and the efficiency of transformers in computer vision. The goal is to develop highly effective and efficient vision transformers; specifically, transformers with high performance and low resource cost. The performance determines whether the model can be applied on real-world applications, while the resource cost influences the deployment on devices [242], [243]. The effectiveness is usually correlated with the efficiency, so determining how to achieve a better balance between them is a meaningful topic for future study.

Most of the existing vision transformer models are designed to handle only a single task. Many NLP models such as GPT-3 [11] have demonstrated how transformer can deal with multiple tasks in one model. IPT [19] in the CV field is also able to process multiple low-level vision tasks, such as super-resolution, image denoising, and deraining. Perceiver [244] and Perceiver IO [245] are the pioneering models that can work on several domains including images, audio, multimodal, point clouds. We believe that more tasks can be involved in only one model. Unifying all visual tasks and even other tasks in one transformer (i.e., a grand unified model) is an exciting topic.

There have been various types of neural networks, such as CNN, RNN, and transformer. In the CV field, CNNs used to be the mainstream choice [12], [93], but now transformer is becoming popular. CNNs can capture inductive biases such as translation equivariance and locality, whereas ViT uses large-scale training to surpass inductive bias [15]. From the evidence currently available [15], CNNs perform well on small datasets, whereas transformers perform better on large datasets. The question for the future is whether to use CNN or transformer.

By training with large datasets, transformers can achieve state-of-the-art performance on both NLP [11], [10] and CV benchmarks [15]. It is possible that neural networks need big data rather than inductive bias. In closing, we leave you with a question: Can transformer obtains satisfactory results with a very simple computational paradigm (*e.g.*, with only fully connected layers) and massive data training?

ACKNOWLEDGEMENT

This research is partially supported by MindSpore (https://mindspore.cn/) and CANN (Compute Architecture for Neural Networks).

APPENDIX

A1. General Formulation of Self-attention

The self-attention module [9] for machine translation computes the responses at each position in a sequence by estimating attention scores to all positions and gathering the corresponding embeddings based on the scores accordingly. This can be viewed as a form of non-local filtering operations [246], [247]. We follow the convention [246] to formulate the self-attention module. Given an input signal (*e.g.*, image, sequence, video and feature) $\mathbf{X} \in \mathbb{R}^{n \times d}$, where $n = h \times w$ (indicating the number of pixels in feature) and d is the number of channels, the output signal is generated as:

$$\mathbf{y}_i = \frac{1}{C(\mathbf{x}_i)} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j), \tag{14}$$

where $\mathbf{x}_i \in \mathbb{R}^{1 \times d}$ and $\mathbf{y}_i \in \mathbb{R}^{1 \times d}$ indicate the i^{th} position (*e.g.*, space, time and spacetime) of the input signal $\mathbf{X}$ and output signal $\mathbf{Y}$, respectively. Subscript j is the index that enumerates all positions, and a pairwise function $f(\cdot)$ computes a representing relationship (such as affinity) between i and all j. The function $g(\cdot)$ computes a representation of the input signal at position j, and the response is normalized by a factor $C(x_i)$.

Note that there are many choices for the pairwise function $f(\cdot)$. For example, a simple extension of the Gaussian function could be used to compute the similarity in an embedding space. As such, the function $f(\cdot)$ can be formulated as:

$$f(\mathbf{x}_i, \mathbf{x}_j) = e^{\theta(\mathbf{x}_i)\phi(\mathbf{x}_j)^T} \tag{15}$$

where $\theta(\cdot)$ and $\phi(\cdot)$ can be any embedding layers. If we consider the $\theta(\cdot), \phi(\cdot), g(\cdot)$ in the form of linear embedding: $\theta(\mathbf{X}) = \mathbf{X}\mathbf{W}_\theta$, $\phi(\mathbf{X}) = \mathbf{X}\mathbf{W}_\phi, g(\mathbf{X}) = \mathbf{X}\mathbf{W}_g$ where $\mathbf{W}_\theta \in \mathbb{R}^{d \times d_k}, \mathbf{W}_\phi \in \mathbb{R}^{d \times d_k}, \mathbf{W}_g \in \mathbb{R}^{d \times d_v}$, and set the normalization factor as $C(\mathbf{x}_i) = \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j)$, the Eq. 14 can be rewritten as:

$$\mathbf{y}_i = \frac{e^{\mathbf{x}_i w_{\theta,i} \mathbf{w}_{\phi,j}^T \mathbf{x}_j^T}}{\sum_j e^{\mathbf{x}_i \mathbf{w}_{\theta,i} \mathbf{w}_{\phi,j}^T x_j^T}} \mathbf{x}_j \mathbf{w}_{g,j}, \tag{16}$$

where $\mathbf{w}_{\theta,i} \in \mathbb{R}^{d \times 1}$ is the i^{th} row of the weight matrix W_θ. For a given index i, $\frac{1}{C(\mathbf{x}_i)} f(\mathbf{x}_i, \mathbf{x}_j)$ becomes the softmax output along the dimension j. The formulation can be further rewritten as:

$$\mathbf{Y} = \mathrm{softmax}(\mathbf{X}\mathbf{W}_\theta \mathbf{W}_\phi^T \mathbf{X}) g(\mathbf{X}), \tag{17}$$

where $\mathbf{Y} \in \mathbb{R}^{n \times c}$ is the output signal of the same size as $\mathbf{X}$. Compared with the query, key and value representations $\mathbf{Q} = \mathbf{X}\mathbf{W}_q, \mathbf{K} = \mathbf{X}\mathbf{W}_k, \mathbf{V} = \mathbf{X}\mathbf{W}_v$ from the translation module, once $\mathbf{W}_q = \mathbf{W}_\theta, \mathbf{W}_k = \mathbf{W}_\phi, \mathbf{W}_v = \mathbf{W}_g$, Eq. 17 can be formulated as:

$$\mathbf{Y} = \mathrm{softmax}(\mathbf{Q}\mathbf{K}^T)\mathbf{V} = \mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}), \tag{18}$$

The self-attention module [9] proposed for machine translation is, to some extent, the same as the preceding non-local filtering operations proposed for computer vision.

Generally, the final output signal of the self-attention module for computer vision will be wrapped as:

$$\mathbf{Z} = \mathbf{Y}\mathbf{W}^o + \mathbf{X} \tag{19}$$

where $\mathbf{Y}$ is generated through Eq. 17. If $\mathbf{W}^o$ is initialized as zero, this self-attention module can be inserted into any existing model without breaking its initial behavior.

A2. Revisiting Transformers for NLP

Before transformer was developed, RNNs (*e.g.*, GRU [248] and LSTM [6]) with added attention [7] empowered most of the state-of-the-art language models. However, RNNs require the information flow to be processed sequentially from the previous hidden states to the next one. This rules out the possibility of using acceleration and parallelization during training, and consequently hinders the potential of RNNs to process longer sequences or build larger models. In 2017, Vaswani *et al.* [9] proposed transformer, a novel encoder-decoder architecture built solely on multi-head self-attention mechanisms and feed-forward neural networks. Its purpose was to solve seq-to-seq natural language tasks (*e.g.*, machine translation) easily by acquiring global dependencies. The subsequent success of transformer demonstrates that leveraging attention mechanisms alone can achieve performance comparable with attentive RNNs. Furthermore, the architecture of transformer lends itself to massively parallel computing, which enables training on larger datasets. This has given rise to the surge of large pre-trained models (PTMs) for natural language processing.

BERT [10] and its variants (*e.g.*, SpanBERT [249], RoBERTa [250]) are a series of PTMs built on the multi-layer transformer encoder architecture. Two tasks are conducted on BookCorpus [251] and English Wikipedia datasets at the pre-training stage of BERT: 1) Masked language modeling (MLM), which involves first randomly masking out some tokens in the input and then training the model to predict; 2) Next sentence prediction, which uses paired sentences as input and predicts whether the second sentence is the original one in the document. After pre-training, BERT can be fine-tuned by adding an output layer on a wide range of downstream tasks. More specifically, when performing sequence-level tasks (*e.g.*, sentiment analysis), BERT uses the representation of the first token for classification; for token-level tasks (*e.g.*, name entity recognition), all tokens are fed into the softmax layer for classification. At the time of its release, BERT achieved the state-of-the-art performance on 11 NLP tasks, setting a milestone in pre-trained language models. Generative Pre-trained Transformer models (*e.g.*, GPT [252], GPT-2 [109]) are another type of PTMs based on the transformer decoder architecture, which uses masked self-attention mechanisms. The main difference between the GPT series and BERT is the way in which pre-training is performed. Unlike BERT, GPT models are unidirectional language models pre-trained using Left-to-Right (LTR) language modeling. Furthermore, BERT learns the sentence separator ([SEP]) and classifier token ([CLS]) embeddings during pre-training, whereas these embeddings are involved in only the fine-tuning stage of GPT. Due to its unidirectional pre-training strategy, GPT achieves superior performance in many natural language generation tasks. More recently, a massive transformer-based model called GPT-3, which has an astonishing 175 billion parameters, was developed [11]. By pre-training on 45 TB of compressed plaintext data, GPT-3 can directly process different types of downstream natural language tasks without fine-tuning.

As a result, it achieves strong performance on many NLP datasets, including both natural language understanding and generation. Since the introduction of transformer, many other models have been proposed in addition to the transformer-based PTMs mentioned earlier. We list a few representative models in Table 5 for interested readers, but this is not the focus of our study.

TABLE 5: List of representative language models built on transformer. Transformer is the standard encoder-decoder architecture. Transformer Enc. and Dec. represent the encoder and decoder, respectively. Decoder uses mask self-attention to prevent attending to the future tokens. The data of the Table is from [197].

Models	Architecture	# of Params	Fine-tuning
GPT [252]	Transformer Dec.	117M	Yes
GPT-2 [109]	Transformer Dec.	117M-1542M	No
GPT-3 [11]	Transformer Dec.	125M-175B	No
BERT [10]	Transformer Enc.	110M-340M	Yes
RoBERTa [250]	Transformer Enc.	355M	Yes
XLNet [253]	Two-Stream Transformer Enc.	$\approx$ BERT	Yes
ELECTRA [254]	Transformer Enc.	335M	Yes
UniLM [255]	Transformer Enc.	340M	Yes
BART [256]	Transformer	110% of BERT	Yes
T5 [152]	Transfomer	220M-11B	Yes
ERNIE (THU) [257]	Transformer Enc.	114M	Yes
KnowBERT [258]	Transformer Enc.	253M-523M	Yes

Apart from the PTMs trained on large corpora for general NLP tasks, transformer-based models have also been applied in many other NLP-related domains and to multi-modal tasks.

BioNLP Domain. Transformer-based models have outperformed many traditional biomedical methods. Some examples of such models include BioBERT [259], which uses a transformer architecture for biomedical text mining tasks, and SciBERT [260], which is developed by training transformer on 114M scientific articles (covering biomedical and computer science fields) with the aim of executing NLP tasks in the scientific domain more precisely. Another example is ClinicalBERT, proposed by Huang *et al.* [261]. It utilizes transformer to develop and evaluate continuous representations of clinical notes. One of the side effects of this is that the attention map of ClinicalBERT can be used to explain predictions, thereby allowing high-quality connections between different medical contents to be discovered.

The rapid development of transformer-based models on a variety of NLP-related tasks demonstrates its structural superiority and versatility, opening up the possibility that it will become a universal module applied in many AI fields other than just NLP. The following part of this survey focuses on the applications of transformer in a wide range of computer vision tasks that have emerged over the past two years.

A3. Self-attention for Computer Vision

The preceding sections reviewed methods that use a transformer architecture for vision tasks. We can conclude that self-attention plays a pivotal role in transformer. The self-attention module can also be considered a building block of CNN architectures, which have low scaling properties concerning the large receptive fields. This building block is widely used on top of the networks to capture long-range interactions and enhance high-level semantic features for vision tasks. In this section, we delve deeply into the models based on self-attention designed for challenging tasks in computer vision. Such tasks include semantic segmentation, instance segmentation, object detection, keypoint detection, and depth estimation. Here we briefly summarize the existing applications using self-attention for computer vision.

Image Classification. Trainable attention for classification consists of two main streams: hard attention [262], [263], [264] regarding the use of an image region, and soft attention [265], [266], [267], [268] generating non-rigid feature maps. Ba *et al.* [262] first proposed the term "visual attention" for image classification tasks, and used attention to select relevant regions and locations within the input image. This can also reduce the computational complexity of the proposed model regarding the size of the input image. For medical image classification, AG-CNN [269] was proposed to crop a sub-region from a global image by the attention heat map. And instead of using hard attention and recalibrating the crop of feature maps, SENet [270] was proposed to reweight the channel-wise responses of the convolutional features using soft self-attention. Jetley *et al.* [266] used attention maps generated by corresponding estimators to reweight intermediate features in DNNs. In addition, Han *et al.* [267] utilized the attribute-aware attention to enhance the representation of CNNs.

Semantic Segmentation. PSANet [271], OCNet [272], DANet [273] and CFNet [274] are the pioneering works to propose using the self-attention module in semantic segmentation tasks. These works consider and augment the relationship and similarity [275], [276], [277], [278], [279], [280] between the contextual pixels. DANet [273] simultaneously leverages the self-attention module on spatial and channel dimensions, whereas A^2Net [281] groups the pixels into a set of regions, and then augments the pixel representations by aggregating the region representations with the generated attention weights. DGCNet [282] employs a dual graph CNN to model coordinate space similarity and feature space similarity in a single framework. To improve the efficiency of the self-attention module for semantic segmentation, several works [283], [284], [285], [286], [287] have been proposed, aiming to alleviate the huge amount of parameters brought by calculating pixel similarities. For example, CGNL [283] applies the Taylor series of the RBF kernel function to approximate the pixel similarities. CCNet [284] approximates the original self-attention scheme via two consecutive criss-cross attention modules. In addition, ISSA [285] factorizes the dense affinity matrix as the product of two sparse affinity matrices. There are other related works using attention based graph reasoning modules [288], [289], [286] to enhance both the local and global representations.

Object Detection. Ramachandran *et al.* [268] proposes an attention-based layer and swapped the conventional convolution layers to build a fully attentional detector that outperforms the typical RetinaNet [127] on COCO benchmark [290]. GCNet [291] assumes that the global contexts modeled by non-local operations are almost the same for different query positions within an image, and unifies the simplified formulation and SENet [270] into a general framework for global context modeling [292], [293], [294], [295]. Vo *et al.* [296] designs a bidirectional operation to gather and distribute information from a query position to all possible positions. Zhang *et al.* [118] suggests that previous methods fail to interact with cross-scale features, and proposes Feature Pyramid Transformer, based on the self-attention module, to fully exploit interactions across both space and scales.

Conventional detection methods usually exploit a single visual representation (*e.g.*, bounding box and corner point) for predicting the final results. Hu *et al.* [297] proposes a relation module based on self-attention to process a set of objects simultaneously through interaction between their appearance features. Cheng *et al.* [119] proposes RelationNet++ with the bridging visual representations (BVR) module to combine different heterogeneous representations into a single one similar to that in the self-attention module. Specifically, the master representation is treated as the query input and the auxiliary representations are regarded as the key input. The enhanced feature can therefore bridge the information from auxiliary representations and benefit final detection results.

Other Vision Tasks. Zhang *et al.* [298] proposes a resolution-wise attention module to learn enhanced feature maps when training multi-resolution networks to obtain accurate human key-point locations for pose estimation task. Furthermore, Chang *et al.* [299] uses an attention-mechanism based feature fusion block to improve the accuracy of the human keypoint detection model.

To explore more generalized contextual information for improving the self-supervised monocular trained depth estimation, Johnston *et al.* [300] directly leverages self-attention module. Chen *et al.* [301] also proposes an attention-based aggregation network to capture context information that differs in diverse scenes for depth estimation. And Aich *et al.* [302] proposes bidirectional attention modules that utilize the forward and backward attention operations for better results of monocular depth estimation.

References

[1] F. Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.
[2] F. ROSENBLATT. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, 1961.
[3] Y. LeCun et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
[4] A. Krizhevsky et al. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pp. 1097–1105, 2012.
[5] D. E. Rumelhart et al. Learning internal representations by error propagation. Technical report, 1985.
[6] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
[7] D. Bahdanau et al. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.
[8] A. Parikh et al. A decomposable attention model for natural language inference. In *EMNLP*, 2016.
[9] A. Vaswani et al. Attention is all you need. In *NeurIPS*, 2017.
[10] J. Devlin et al. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.
[11] T. B. Brown et al. Language models are few-shot learners. In *NeurIPS*, 2020.
[12] K. He et al. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.
[13] S. Ren et al. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.
[14] M. Chen et al. Generative pretraining from pixels. In *ICML*, 2020.
[15] A. Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.
[16] N. Carion et al. End-to-end object detection with transformers. In *ECCV*, 2020.
[17] X. Zhu et al. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2021.
[18] S. Zheng et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021.
[19] H. Chen et al. Pre-trained image processing transformer. In *CVPR*, 2021.
[20] L. Zhou et al. End-to-end dense video captioning with masked transformer. In *CVPR*, pp. 8739–8748, 2018.
[21] S. Ullman et al. *High-level vision: Object recognition and visual cognition*, volume 2. MIT press Cambridge, MA, 1996.
[22] R. Kimchi et al. *Perceptual organization in vision: Behavioral and neural perspectives*. Psychology Press, 2003.
[23] J. Zhu et al. Top-down saliency detection via contextual pooling. *Journal of Signal Processing Systems*, 74(1):33–46, 2014.
[24] J. Long et al. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
[25] H. Wang et al. Max-deeplab: End-to-end panoptic segmentation with mask transformers. In *CVPR*, pp. 5463–5474, 2021.
[26] R. B. Fisher. Cvonline: The evolving, distributed, non-proprietary, on-line compendium of computer vision. *Retrieved January 28, 2006 from http://homepages. inf. ed. ac. uk/rbf/CVonline*, 2008.
[27] N. Parmar et al. Image transformer. In *ICML*, 2018.
[28] Y. Zeng et al. Learning joint spatial-temporal transformations for video inpainting. In *ECCV*, pp. 528–543. Springer, 2020.
[29] K. Han et al. Transformer in transformer. In *NeurIPS*, 2021.
[30] H. Cao et al. Swin-unet: Unet-like pure transformer for medical image segmentation. *arXiv:2105.05537*, 2021.
[31] X. Chen et al. An empirical study of training self-supervised vision transformers. In *ICCV*, 2021.
[32] Z. Dai et al. UP-DETR: unsupervised pre-training for object detection with transformers. In *CVPR*, 2021.
[33] Y. Wang et al. End-to-end video instance segmentation with transformers. In *CVPR*, 2021.
[34] L. Huang et al. Hand-transformer: Non-autoregressive structured modeling for 3d hand pose estimation. In *ECCV*, pp. 17–33, 2020.
[35] L. Huang et al. Hot-net: Non-autoregressive transformer for 3d hand-object pose estimation. In *ACM MM*, pp. 3136–3145, 2020.
[36] K. Lin et al. End-to-end human pose and mesh reconstruction with transformers. In *CVPR*, 2021.
[37] P. Esser et al. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.
[38] Y. Jiang et al. Transgan: Two transformers can make one strong gan. In *NeurIPS*, 2021.
[39] F. Yang et al. Learning texture transformer network for image super-resolution. In *CVPR*, pp. 5791–5800, 2020.
[40] A. Radford et al. Learning transferable visual models from natural language supervision. *arXiv:2103.00020*, 2021.
[41] A. Ramesh et al. Zero-shot text-to-image generation. In *ICML*, 2021.
[42] M. Ding et al. Cogview: Mastering text-to-image generation via transformers. In *NeurIPS*, 2021.
[43] R. Hu and A. Singh. Unit: Multimodal multitask learning with a unified transformer. In *ICCV*, 2021.
[44] P. Michel et al. Are sixteen heads really better than one? In *NeurIPS*, pp. 14014–14024, 2019.
[45] X. Jiao et al. TinyBERT: Distilling BERT for natural language understanding. In *Findings of EMNLP*, pp. 4163–4174, 2020.
[46] G. Prato et al. Fully quantized transformer for machine translation. In *Findings of EMNLP*, 2020.
[47] Z.-H. Jiang et al. Convbert: Improving bert with span-based dynamic convolution. *NeurIPS*, 33, 2020.
[48] J. Gehring et al. Convolutional sequence to sequence learning. In *ICML*, pp. 1243–1252. PMLR, 2017.
[49] P. Shaw et al. Self-attention with relative position representations. In *NAACL*, pp. 464–468, 2018.
[50] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv:1606.08415*, 2016.
[51] J. L. Ba et al. Layer normalization. *arXiv:1607.06450*, 2016.
[52] A. Baevski and M. Auli. Adaptive input representations for neural language modeling. In *ICLR*, 2019.
[53] Q. Wang et al. Learning deep transformer models for machine translation. In *ACL*, pp. 1810–1822, 2019.
[54] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.
[55] S. Shen et al. Powernorm: Rethinking batch normalization in transformers. In *ICML*, 2020.
[56] J. Xu et al. Understanding and improving layer normalization. In *NeurIPS*, 2019.
[57] T. Bachlechner et al. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence*, pp. 1352–1361. PMLR, 2021.
[58] B. Wu et al. Visual transformers: Token-based image representation and processing for computer vision. *arXiv:2006.03677*, 2020.
[59] H. Touvron et al. Training data-efficient image transformers & distillation through attention. In *ICML*, 2020.
[60] Z. Liu et al. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.
[61] C.-F. Chen et al. Regionvit: Regional-to-local attention for vision transformers. *arXiv:2106.02689*, 2021.
[62] X. Chu et al. Twins: Revisiting the design of spatial attention in vision transformers. *arXiv:2104.13840*, 2021.
[63] H. Lin et al. Cat: Cross attention in vision transformer. *arXiv*, 2021.

[64] X. Dong et al. Cswin transformer: A general vision transformer backbone with cross-shaped windows. *arXiv:2107.00652*, 2021.
[65] Z. Huang et al. Shuffle transformer: Rethinking spatial shuffle for vision transformer. *arXiv:2106.03650*, 2021.
[66] J. Fang et al. Msg-transformer: Exchanging local spatial information by manipulating messenger tokens. *arXiv:2105.15168*, 2021.
[67] L. Yuan et al. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*, 2021.
[68] D. Zhou et al. Deepvit: Towards deeper vision transformer. *arXiv*, 2021.
[69] P. Wang et al. Kvt: k-nn attention for boosting vision transformers. *arXiv:2106.00515*, 2021.
[70] D. Zhou et al. Refiner: Refining self-attention for vision transformers. *arXiv:2106.03714*, 2021.
[71] A. El-Nouby et al. Xcit: Cross-covariance image transformers. *arXiv:2106.09681*, 2021.
[72] W. Wang et al. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, 2021.
[73] S. Sun* et al. Visual parser: Representing part-whole hierarchies with transformers. *arXiv:2107.05790*, 2021.
[74] H. Fan et al. Multiscale vision transformers. *arXiv:2104.11227*, 2021.
[75] Z. Zhang et al. Nested hierarchical transformer: Towards accurate, data-efficient and interpretable visual understanding. In *AAAI*, 2022.
[76] Z. Pan et al. Less is more: Pay less attention in vision transformers. In *AAAI*, 2022.
[77] Z. Pan et al. Scalable visual transformers with hierarchical pooling. In *ICCV*, 2021.
[78] B. Heo et al. Rethinking spatial dimensions of vision transformers. In *ICCV*, 2021.
[79] C.-F. Chen et al. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *ICCV*, 2021.
[80] Z. Wang et al. Uformer: A general u-shaped transformer for image restoration. *arXiv:2106.03106*, 2021.
[81] X. Zhai et al. Scaling vision transformers. *arXiv:2106.04560*, 2021.
[82] X. Su et al. Vision transformer architecture search. *arXiv*, 2021.
[83] M. Chen et al. Autoformer: Searching transformers for visual recognition. In *ICCV*, pp. 12270–12280, 2021.
[84] B. Chen et al. Glit: Neural architecture search for global and local image transformer. In *ICCV*, pp. 12–21, 2021.
[85] X. Chu et al. Conditional positional encodings for vision transformers. *arXiv:2102.10882*, 2021.
[86] K. Wu et al. Rethinking and improving relative position encoding for vision transformer. In *ICCV*, 2021.
[87] H. Touvron et al. Going deeper with image transformers. *arXiv:2103.17239*, 2021.
[88] Y. Tang et al. Augmented shortcuts for vision transformers. In *NeurIPS*, 2021.
[89] I. Tolstikhin et al. Mlp-mixer: An all-mlp architecture for vision. *arXiv:2105.01601*, 2021.
[90] L. Melas-Kyriazi. Do you even need attention? a stack of feed-forward layers does surprisingly well on imagenet. *arXiv:2105.02723*, 2021.
[91] M.-H. Guo et al. Beyond self-attention: External attention using two linear layers for visual tasks. *arXiv:2105.02358*, 2021.
[92] H. Touvron et al. Resmlp: Feedforward networks for image classification with data-efficient training. *arXiv:2105.03404*, 2021.
[93] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.
[94] J. Guo et al. Cmt: Convolutional neural networks meet vision transformers. *arXiv:2107.06263*, 2021.
[95] L. Yuan et al. Volo: Vision outlooker for visual recognition. *arXiv:2106.13112*, 2021.
[96] H. Wu et al. Cvt: Introducing convolutions to vision transformers. *arXiv:2103.15808*, 2021.
[97] K. Yuan et al. Incorporating convolution designs into visual transformers. *arXiv:2103.11816*, 2021.
[98] Y. Li et al. Localvit: Bringing locality to vision transformers. *arXiv:2104.05707*, 2021.
[99] B. Graham et al. Levit: a vision transformer in convnet's clothing for faster inference. In *ICCV*, 2021.
[100] A. Srinivas et al. Bottleneck transformers for visual recognition. In *CVPR*, 2021.
[101] Z. Chen et al. Visformer: The vision-friendly transformer. *arXiv*, 2021.
[102] T. Xiao et al. Early convolutions help transformers see better. In *NeurIPS*, volume 34, 2021.
[103] G. E. Hinton and R. S. Zemel. Autoencoders, minimum description length, and helmholtz free energy. *NIPS*, 6:3–10, 1994.
[104] P. Vincent et al. Extracting and composing robust features with denoising autoencoders. In *ICML*, pp. 1096–1103, 2008.
[105] A. v. d. Oord et al. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016.
[106] D. Pathak et al. Context encoders: Feature learning by inpainting. In *CVPR*, pp. 2536–2544, 2016.
[107] Z. Li et al. Mst: Masked self-supervised transformer for visual representation. In *NeurIPS*, 2021.
[108] H. Bao et al. Beit: Bert pre-training of image transformers. *arXiv:2106.08254*, 2021.
[109] A. Radford et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
[110] Z. Xie et al. Self-supervised learning with swin transformers. *arXiv:2105.04553*, 2021.
[111] C. Li et al. Efficient self-supervised vision transformers for representation learning. *arXiv:2106.09785*, 2021.
[112] K. He et al. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.
[113] J. Beal et al. Toward transformer-based object detection. *arXiv:2012.09958*, 2020.
[114] Z. Yuan et al. Temporal-channel transformer for 3d lidar-based video object detection for autonomous driving. *IEEE TCSVT*, 2021.
[115] X. Pan et al. 3d object detection with pointformer. In *CVPR*, 2021.
[116] R. Liu et al. End-to-end lane shape prediction with transformers. In *WACV*, 2021.
[117] S. Yang et al. Transpose: Keypoint localization via transformer. In *ICCV*, 2021.
[118] D. Zhang et al. Feature pyramid transformer. In *ECCV*, 2020.
[119] C. Chi et al. Relationnet++: Bridging visual representations for object detection via transformer decoder. *NeurIPS*, 2020.
[120] Z. Sun et al. Rethinking transformer-based set prediction for object detection. In *ICCV*, pp. 3611–3620, 2021.
[121] M. Zheng et al. End-to-end object detection with adaptive clustering transformer. In *BMVC*, 2021.
[122] T. Ma et al. Oriented object detection with transformer. *arXiv:2106.03146*, 2021.
[123] P. Gao et al. Fast convergence of detr with spatially modulated co-attention. In *ICCV*, 2021.
[124] Z. Yao et al. Efficient detr: Improving end-to-end object detector with dense prior. *arXiv:2104.01318*, 2021.
[125] Z. Tian et al. Fcos: Fully convolutional one-stage object detection. In *ICCV*, pp. 9627–9636, 2019.
[126] Y. Fang et al. You only look at one sequence: Rethinking transformer in vision through object detection. In *NeurIPS*, 2021.
[127] T.-Y. Lin et al. Focal loss for dense object detection. In *ICCV*, 2017.
[128] Z. Cai and N. Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018.
[129] A. Bar et al. Detreg: Unsupervised pretraining with region priors for object detection. *arXiv:2106.04550*, 2021.
[130] J. Hu et al. Istr: End-to-end instance segmentation with transformers. *arXiv:2105.00637*, 2021.
[131] Z. Yang et al. Associating objects with transformers for video object segmentation. In *NeurIPS*, 2021.
[132] S. Wu et al. Fully transformer networks for semantic image segmentation. *arXiv:2106.04108*, 2021.
[133] B. Dong et al. Solq: Segmenting objects by learning queries. In *NeurIPS*, 2021.
[134] R. Strudel et al. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021.
[135] E. Xie et al. Segformer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, 2021.
[136] J. M. J. Valanarasu et al. Medical transformer: Gated axial-attention for medical image segmentation. In *MICCAI*, 2021.
[137] T. Prangemeier et al. Attention-based transformers for instance segmentation of cells in microstructures. In *International Conference on Bioinformatics and Biomedicine*, pp. 700–707. IEEE, 2020.
[138] C. R. Qi et al. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pp. 652–660, 2017.
[139] C. R. Qi et al. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *NeurIPS*, 30:5099–5108, 2017.
[140] S. Hampali et al. Handsformer: Keypoint transformer for monocular 3d pose estimation ofhands and object in interaction. *arXiv*, 2021.
[141] Y. Li et al. Tokenpose: Learning keypoint tokens for human pose estimation. In *ICCV*, 2021.
[142] W. Mao et al. Tfpose: Direct human pose estimation with transformers. *arXiv:2103.15320*, 2021.
[143] T. Jiang et al. Skeletor: Skeletal transformers for robust body-pose estimation. In *CVPR*, 2021.

[144] Y. Li et al. Test-time personalization with a transformer for human pose estimation. *Advances in Neural Information Processing Systems*, 34, 2021.

[145] M. Lin et al. Detr for pedestrian detection. *arXiv:2012.06785*, 2020.

[146] L. Tabelini et al. Polylanenet: Lane estimation via deep polynomial regression. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 6150–6156. IEEE, 2021.

[147] L. Liu et al. Condlanenet: a top-to-down lane detection framework based on conditional convolution. *arXiv:2105.05003*, 2021.

[148] P. Xu et al. A survey of scene graph: Generation and application. *IEEE Trans. Neural Netw. Learn. Syst*, 2020.

[149] J. Yang et al. Graph r-cnn for scene graph generation. In *ECCV*, 2018.

[150] S. Sharifzadeh et al. Classification by attention: Scene graph classification with prior knowledge. In *AAAI*, 2021.

[151] S. Sharifzadeh et al. Improving Visual Reasoning by Exploiting The Knowledge in Texts. *arXiv:2102.04760*, 2021.

[152] C. Raffel et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21(140):1–67, 2020.

[153] N. Wang et al. Transformer meets tracker: Exploiting temporal context for robust visual tracking. In *CVPR*, 2021.

[154] M. Zhao et al. TrTr: Visual Tracking with Transformer. *arXiv:2105.03817 [cs]*, May 2021. arXiv: 2105.03817.

[155] X. Chen et al. Transformer tracking. In *CVPR*, 2021.

[156] P. Sun et al. TransTrack: Multiple Object Tracking with Transformer. *arXiv:2012.15460 [cs]*, May 2021. arXiv: 2012.15460.

[157] S. He et al. TransReID: Transformer-based object re-identification. In *ICCV*, 2021.

[158] X. Liu et al. A video is worth three views: Trigeminal transformers for video-based person re-identification. *arXiv:2104.01745*, 2021.

[159] T. Zhang et al. Spatiotemporal transformer for video-based person re-identification. *arXiv:2103.16469*, 2021.

[160] N. Engel et al. Point transformer. *IEEE Access*, 9:134826–134840, 2021.

[161] M.-H. Guo et al. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021.

[162] H. Zhao et al. Point transformer. In *ICCV*, pp. 16259–16268, 2021.

[163] K. Lee et al. Vitgan: Training gans with vision transformers. *arXiv preprint arXiv:2107.04589*, 2021.

[164] A. v. d. Oord et al. Neural discrete representation learning. *arXiv*, 2017.

[165] X. Wang et al. Sceneformer: Indoor scene generation with transformers. In *3DV*, pp. 106–115. IEEE, 2021.

[166] Z. Liu et al. Convtransformer: A convolutional transformer network for video frame synthesis. *arXiv:2011.10185*, 2020.

[167] R. Girdhar et al. Video action transformer network. In *CVPR*, 2019.

[168] H. Liu et al. Two-stream transformer networks for video-based face alignment. *T-PAMI*, 40(11):2546–2554, 2017.

[169] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[170] S. Lohit et al. Temporal transformer networks: Joint learning of invariant and discriminative time warping. In *CVPR*, 2019.

[171] M. Fayyaz and J. Gall. Sct: Set constrained temporal transformer for set supervised action segmentation. In *2020 CVPR*, pp. 501–510, 2020.

[172] W. Choi et al. What are they doing?: Collective activity classification using spatio-temporal relationship among people. In *ICCVW*, 2009.

[173] K. Gavrilyuk et al. Actor-transformers for group activity recognition. In *CVPR*, pp. 839–848, 2020.

[174] J. Shao et al. Temporal context aggregation for video retrieval with contrastive learning. In *WACV*, 2021.

[175] V. Gabeur et al. Multi-modal transformer for video retrieval. In *ECCV*, pp. 214–229, 2020.

[176] Y. Chen et al. Memory enhanced global-local aggregation for video object detection. In *CVPR*, pp. 10337–10346, 2020.

[177] J. Yin et al. Lidar-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention. In *2020 CVPR*, pp. 11495–11504, 2020.

[178] H. Seong et al. Video multitask transformer network. In *ICCVW*, 2019.

[179] K. M. Schatz et al. A recurrent transformer network for novel view action synthesis. In *ECCV (27)*, pp. 410–426, 2020.

[180] C. Sun et al. Videobert: A joint model for video and language representation learning. In *ICCV*, pp. 7464–7473, 2019.

[181] L. H. Li et al. Visualbert: A simple and performant baseline for vision and language. *arXiv:1908.03557*, 2019.

[182] W. Su et al. Vl-bert: Pre-training of generic visual-linguistic representations. In *ICLR*, 2020.

[183] Y.-S. Chuang et al. Speechbert: Cross-modal pre-trained language model for end-to-end spoken question answering. In *Interspeech*, 2020.

[184] S. Prasanna et al. When bert plays the lottery, all tickets are winning. In *EMNLP*, 2020.

[185] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2018.

[186] Y. Tang et al. Patch slimming for efficient vision transformers. *arXiv:2106.02852*, 2021.

[187] M. Zhu et al. Vision transformer pruning. *arXiv:2104.08500*, 2021.

[188] Z. Liu et al. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017.

[189] Z. Lan et al. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*, 2020.

[190] C. Xu et al. Bert-of-theseus: Compressing bert by progressive module replacing. In *EMNLP*, pp. 7859–7869, 2020.

[191] S. Shen et al. Q-bert: Hessian based ultra low precision quantization of bert. In *AAAI*, pp. 8815–8821, 2020.

[192] O. Zafrir et al. Q8bert: Quantized 8bit bert. *arXiv:1910.06188*, 2019.

[193] V. Sanh et al. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv:1910.01108*, 2019.

[194] S. Sun et al. Patient knowledge distillation for bert model compression. In *EMNLP-IJCNLP*, pp. 4323–4332, 2019.

[195] Z. Sun et al. Mobilebert: a compact task-agnostic bert for resource-limited devices. In *ACL*, pp. 2158–2170, 2020.

[196] I. Turc et al. Well-read students learn better: The impact of student initialization on knowledge distillation. *arXiv:1908.08962*, 2019.

[197] X. Qiu et al. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pp. 1–26, 2020.

[198] A. Fan et al. Reducing transformer depth on demand with structured dropout. In *ICLR*, 2020.

[199] L. Hou et al. Dynabert: Dynamic bert with adaptive width and depth. *NeurIPS*, 33, 2020.

[200] Z. Wang et al. Structured pruning of large language models. In *EMNLP*, pp. 6151–6162, 2020.

[201] G. Hinton et al. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015.

[202] C. Buciluǎ et al. Model compression. In *SIGKDD*, pp. 535–541, 2006.

[203] J. Ba and R. Caruana. Do deep nets really need to be deep? *NIPS*, 2014.

[204] S. Mukherjee and A. H. Awadallah. Xtremedistil: Multi-stage distillation for massive multilingual models. In *ACL*, pp. 2221–2234, 2020.

[205] W. Wang et al. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *arXiv:2002.10957*, 2020.

[206] S. I. Mirzadeh et al. Improved knowledge distillation via teacher assistant. In *AAAI*, 2020.

[207] D. Jia et al. Efficient vision transformers via fine-grained manifold distillation. *arXiv:2107.01378*, 2021.

[208] V. Vanhoucke et al. Improving the speed of neural networks on cpus. In *NIPS Workshop*, 2011.

[209] Z. Yang et al. Searching for low-bit weights in quantized neural networks. In *NeurIPS*, 2020.

[210] E. Park and S. Yoo. Profit: A novel training method for sub-4-bit mobilenet models. In *ECCV*, pp. 430–446. Springer, 2020.

[211] J. Fromm et al. Riptide: Fast end-to-end binarized neural networks. *Proceedings of Machine Learning and Systems*, 2:379–389, 2020.

[212] Y. Bai et al. Proxquant: Quantized neural networks via proximal operators. In *ICLR*, 2019.

[213] A. Bhandare et al. Efficient 8-bit quantization of transformer neural machine language translation model. *arXiv:1906.00532*, 2019.

[214] C. Fan. Quantized transformer. Technical report, Stanford Univ., 2019.

[215] K. Shridhar et al. End to end binarized neural networks for text classification. In *SustaiNLP*, 2020.

[216] R. Cheong and R. Daniel. transformers. zip: Compressing transformers with pruning and quantization. Technical report, 2019.

[217] Z. Zhao et al. An investigation on different underlying quantization schemes for pre-trained language models. In *NLPCC*, 2020.

[218] Z. Liu et al. Post-training quantization for vision transformer. In *NeurIPS*, 2021.

[219] Z. Wu et al. Lite transformer with long-short range attention. In *ICLR*, 2020.

[220] Z. Geng et al. Is attention better than matrix decomposition? In *ICLR*, 2020.

[221] Y. Guo et al. Nat: Neural architecture transformer for accurate and compact architectures. In *NeurIPS*, pp. 737–748, 2019.

[222] D. So et al. The evolved transformer. In *ICML*, pp. 5877–5886, 2019.

[223] C. Li et al. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. In *ICCV*, 2021.

[224] A. Katharopoulos et al. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, 2020.

[225] C. Yun et al. $o(n)$ connections are expressive enough: Universal approximability of sparse transformers. In *NeurIPS*, 2020.

[226] M. Zaheer et al. Big bird: Transformers for longer sequences. In *NeurIPS*, 2020.

[227] D. A. Spielman and S.-H. Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4), 2011.

[228] F. Chung and L. Lu. The average distances in random graphs with given expected degrees. *PNAS*, 99(25):15879–15882, 2002.

[229] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[230] X. Zhai et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv:1910.04867*, 2019.

[231] Y. Cheng et al. Robust neural machine translation with doubly adversarial inputs. In *ACL*, 2019.

[232] W. E. Zhang et al. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM TIST*, 11(3):1–41, 2020.

[233] K. Mahmood et al. On the robustness of vision transformers to adversarial examples. *arXiv:2104.02610*, 2021.

[234] X. Mao et al. Towards robust vision transformer. *arXiv*, 2021.

[235] S. Serrano and N. A. Smith. Is attention interpretable? In *ACL*, 2019.

[236] S. Wiegreffe and Y. Pinter. Attention is not not explanation. In *EMNLP-IJCNLP*, 2019.

[237] H. Chefer et al. Transformer interpretability beyond attention visualization. In *CVPR*, pp. 782–791, 2021.

[238] R. Livni et al. On the computational efficiency of training neural networks. In *NeurIPS*, 2014.

[239] B. Neyshabur et al. Towards understanding the role of over-parametrization in generalization of neural networks. In *ICLR*, 2019.

[240] K. Han et al. Ghostnet: More features from cheap operations. In *CVPR*, pp. 1580–1589, 2020.

[241] K. Han et al. Model rubik's cube: Twisting resolution, depth and width for tinynets. *NeurIPS*, 33, 2020.

[242] T. Chen et al. Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning. In *ASPLOS*, pp. 269–284, 2014.

[243] H. Liao et al. Davinci: A scalable architecture for neural network computing. In *2019 IEEE Hot Chips 31 Symposium (HCS)*, 2019.

[244] A. Jaegle et al. Perceiver: General perception with iterative attention. In *ICML*, volume 139, pp. 4651–4664. PMLR, 18–24 Jul 2021.

[245] A. Jaegle et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

[246] X. Wang et al. Non-local neural networks. In *CVPR*, pp. 7794–7803, 2018.

[247] A. Buades et al. A non-local algorithm for image denoising. In *CVPR*, pp. 60–65, 2005.

[248] J. Chung et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555*, 2014.

[249] M. Joshi et al. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

[250] Y. Liu et al. Roberta: A robustly optimized bert pretraining approach. *arXiv:1907.11692*, 2019.

[251] Y. Zhu et al. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*, pp. 19–27, 2015.

[252] A. Radford et al. Improving language understanding by generative pre-training, 2018.

[253] Z. Yang et al. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pp. 5753–5763, 2019.

[254] K. Clark et al. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv:2003.10555*, 2020.

[255] L. Dong et al. Unified language model pre-training for natural language understanding and generation. In *NeurIPS*, pp. 13063–13075, 2019.

[256] M. Lewis et al. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv:1910.13461*, 2019.

[257] Z. Zhang et al. Ernie: Enhanced language representation with informative entities. *arXiv:1905.07129*, 2019.

[258] M. E. Peters et al. Knowledge enhanced contextual word representations. *arXiv:1909.04164*, 2019.

[259] J. Lee et al. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[260] I. Beltagy et al. Scibert: A pretrained language model for scientific text. *arXiv:1903.10676*, 2019.

[261] K. Huang et al. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv:1904.05342*, 2019.

[262] J. Ba et al. Multiple object recognition with visual attention. In *ICLR*, 2014.

[263] V. Mnih et al. Recurrent models of visual attention. *NeurIPS*, pp. 2204–2212, 2014.

[264] K. Xu et al. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pp. 2048–2057, 2015.

[265] F. Wang et al. Residual attention network for image classification. In *CVPR*, pp. 3156–3164, 2017.

[266] S. Jetley et al. Learn to pay attention. In *ICLR*, 2018.

[267] K. Han et al. Attribute-aware attention model for fine-grained representation learning. In *ACM MM*, pp. 2040–2048, 2018.

[268] P. Ramachandran et al. Stand-alone self-attention in vision models. In *NeurIPS*, 2019.

[269] Q. Guan et al. Diagnose like a radiologist: Attention guided convolutional neural network for thorax disease classification. In *arXiv:1801.09927*, 2018.

[270] J. Hu et al. Squeeze-and-excitation networks. In *CVPR*, pp. 7132–7141, 2018.

[271] H. Zhao et al. Psanet: Point-wise spatial attention network for scene parsing. In *ECCV*, pp. 267–283, 2018.

[272] Y. Yuan et al. Ocnet: Object context for semantic segmentation. *International Journal of Computer Vision*, pp. 1–24, 2021.

[273] J. Fu et al. Dual attention network for scene segmentation. In *CVPR*, pp. 3146–3154, 2019.

[274] H. Zhang et al. Co-occurrent features in semantic segmentation. In *CVPR*, pp. 548–557, 2019.

[275] F. Zhang et al. Acfnet: Attentional class feature network for semantic segmentation. In *ICCV*, pp. 6798–6807, 2019.

[276] X. Li et al. Expectation-maximization attention networks for semantic segmentation. In *ICCV*, pp. 9167–9176, 2019.

[277] J. He et al. Adaptive pyramid context network for semantic segmentation. In *CVPR*, pp. 7519–7528, 2019.

[278] O. Oktay et al. Attention u-net: Learning where to look for the pancreas. 2018.

[279] Y. Wang et al. Self-supervised equivariant attention mechanism for weakly supervised semantic segmentation. In *CVPR*, pp. 12275–12284, 2020.

[280] X. Li et al. Global aggregation then local distribution in fully convolutional networks. In *BMVC*, 2019.

[281] Y. Chen et al. Aˆ 2-nets: Double attention networks. *NeurIPS*, pp. 352–361, 2018.

[282] L. Zhang et al. Dual graph convolutional network for semantic segmentation. In *BMVC*, 2019.

[283] K. Yue et al. Compact generalized non-local network. In *NeurIPS*, pp. 6510–6519, 2018.

[284] Z. Huang et al. Ccnet: Criss-cross attention for semantic segmentation. In *ICCV*, pp. 603–612, 2019.

[285] L. Huang et al. Interlaced sparse self-attention for semantic segmentation. *arXiv:1907.12273*, 2019.

[286] Y. Li and A. Gupta. Beyond grids: Learning graph representations for visual recognition. *NeurIPS*, pp. 9225–9235, 2018.

[287] S. Kumaar et al. Cabinet: Efficient context aggregation network for low-latency semantic segmentation. *arXiv:2011.00993*, 2020.

[288] X. Liang et al. Symbolic graph reasoning meets convolutions. *NeurIPS*, pp. 1853–1863, 2018.

[289] Y. Chen et al. Graph-based global reasoning networks. In *CVPR*, pp. 433–442, 2019.

[290] T.-Y. Lin et al. Microsoft coco: Common objects in context. In *ECCV*, pp. 740–755, 2014.

[291] Y. Cao et al. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *ICCV Workshops*, 2019.

[292] W. Li et al. Object detection based on an adaptive attention mechanism. *Scientific Reports*, pp. 1–13, 2020.

[293] T.-I. Hsieh et al. One-shot object detection with co-attention and co-excitation. In *NeurIPS*, pp. 2725–2734, 2019.

[294] Q. Fan et al. Few-shot object detection with attention-rpn and multi-relation detector. In *CVPR*, pp. 4013–4022, 2020.

[295] H. Perreault et al. Spotnet: Self-attention multi-task network for object detection. In *2020 17th Conference on Computer and Robot Vision (CRV)*, pp. 230–237, 2020.

[296] X.-T. Vo et al. Bidirectional non-local networks for object detection. In *International Conference on Computational Collective Intelligence*, pp. 491–501, 2020.

[297] H. Hu et al. Relation networks for object detection. In *CVPR*, pp. 3588–3597, 2018.

[298] K. Zhang et al. Learning enhanced resolution-wise features for human pose estimation. In *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 2256–2260, 2020.
[299] Y. Chang et al. The same size dilated attention network for keypoint detection. In *International Conference on Artificial Neural Networks*, pp. 471–483, 2019.
[300] A. Johnston and G. Carneiro. Self-supervised monocular trained depth estimation using self-attention and discrete disparity volume. In *CVPR*, pp. 4756–4765, 2020.
[301] Y. Chen et al. Attention-based context aggregation network for monocular depth estimation. *International Journal of Machine Learning and Cybernetics*, pp. 1583–1596, 2021.
[302] S. Aich et al. Bidirectional attention network for monocular depth estimation. In *ICRA*, 2021.