A survey of transformers

Bobbili Sravani

Electrical and Electronics Engineering

Swarnandhra college of engineering and technology

Narsapur, INDIA

Sravanibobbili301@gmail.com

ABSTRACT

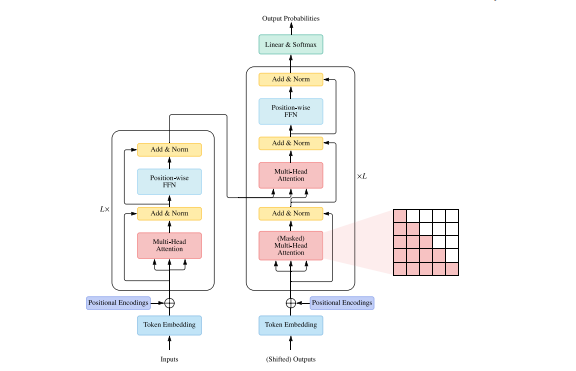
Transformers have achieved great success in many artificial intelligence fields, such as natural language processing, computer vision, and audio processing. Therefore, it is natural to attract lots of interest from academic and industry researchers. Up to the present, a great variety of Transformer variants (a.k.a. X-formers) have been proposed, however, a systematic and comprehensive literature review on these Transformer variants is still missing. In this survey, we provide a comprehensive review of various X-formers. We first briefly introduce the vanilla Transformer and then propose a new taxonomy of X-formers. Next, we introduce the various X-formers from three perspectives: architectural modification, pre-training, and applications. Finally, we outline some potential directions for future research.

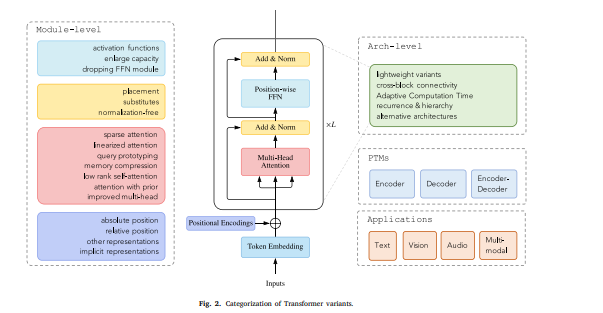
**INTRODUCTION**

Transformer (Vaswani et al., 2017) is a prominent deep learning model that has been widely adopted in various fields, such as natural language processing (NLP), computer vision (CV) and speech processing. Transformer was originally proposed as a sequence-to-sequence model (Sutskever et al., 2014) for machine translation. Later works show that Transformer-based pre-trained models (PTMs) (Qiu et al., 2020) can achieve state-of-the-art performances on various tasks. As a consequence, Transformer has become the go-to architecture in NLP, especially for PTMs. In addition to language related applications, Transformer has also been adopted in CV (Parmar et al., 2018; Carion et al., 2020; Dosovitskiy et al., 2020), audio processing (Dong et al., 2018; Gulati et al., 2020; Chen et al., 2021) and even other disciplines, such as chemistry (Schwaller et al., 2019) and life sciences (Rives et al., 2021). Due to the success, a variety of Transformer variants (a.k.a. Xformers) have been proposed over the past few years. These X-formers improve the vanilla Transformer from different perspectives. 1. Model Efficiency. A key challenge of applying Transformer is its inefficiency at processing long sequences mainly due to the computation and memory complexity of the self-attention module. The improvement methods include lightweight attention (e.g. sparse attention variants) and Divide-and-conquer methods (e.g., recurrent and hierarchical mechanism).

2. Model Generalization. Since the transformer is a flexible architecture and makes few assumptions on the structural bias of input data, it is hard to train on small-scale data. The improvementmethods include introducing structural bias or regularization, pre-training on large-sca sle unlabeled data, etc.

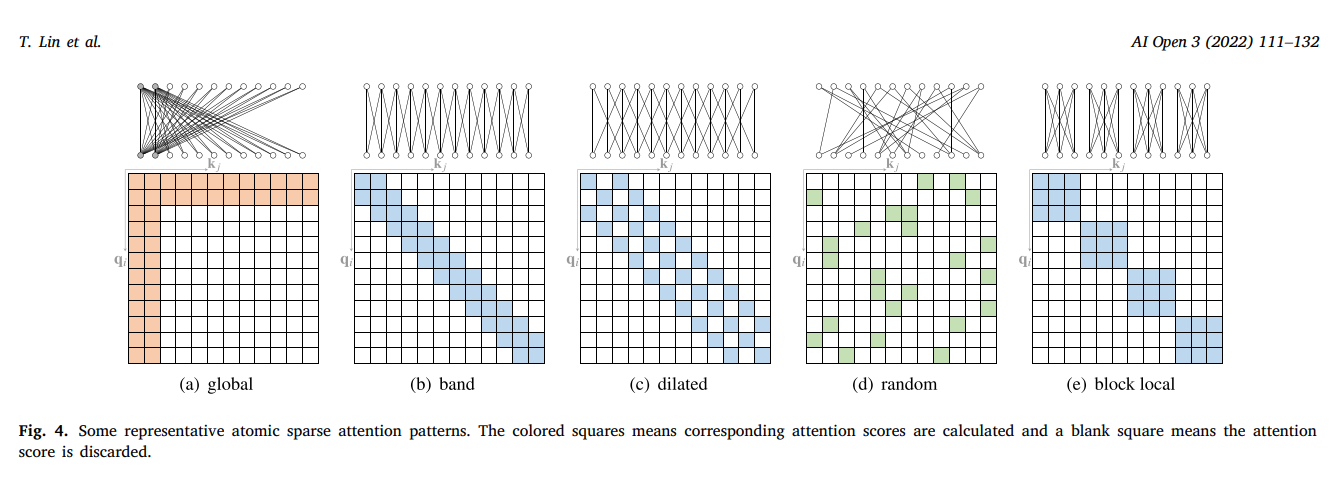
3. Model Adaptation. This line of work aims to adapt the Transformer to to specific downstream tasks and applications.

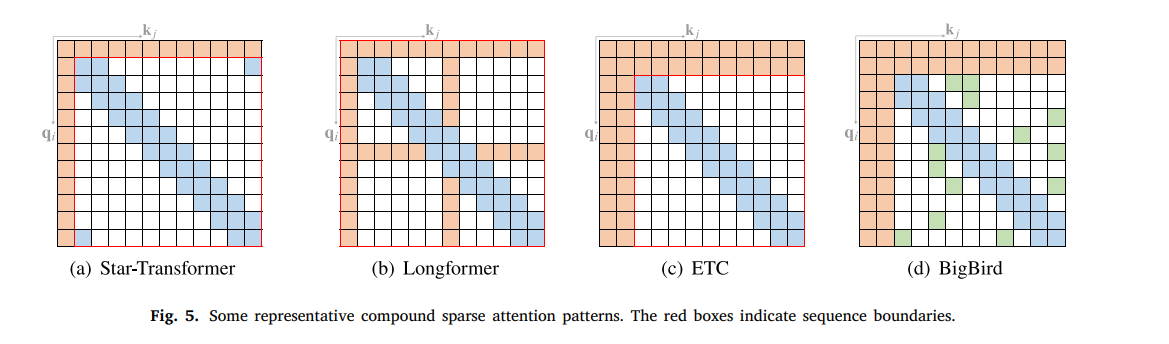
In this survey, we aim to provide a comprehensive review of the Transformer and its variants. Although we can organize X-formers on the basis of the perspectives mentioned above, many existing X-formers may address one or several issues. For example, sparse attention variants not only reduce the computational complexity but also introduce structural prior on input data to alleviate the overfitting problem on small datasets. Therefore, it is more methodical to categorize the various existing X-formers and propose a new taxonomy mainly according to their ways to improve the vanilla Transformer: architecture modification, pre-training, and applications. Considering the audience of this survey may be from different domains, we mainly focus on the general architecture variants and just briefly discuss the specific variants on pre-training and applications. The rest of the survey is organized as follows. Section 2 introduces the architecture and the key components of Transformer. Section 3 clarifies the categorization of Transformer variants. Section 4∼5 review the module-level modifications, including attention module, position encoding, layer normalization and feed-forward layer. Section 6 reviews the architecture-level variants. Section 7 introduces some of the representative Transformer-based PTMs. Section 8 introduces the application of Transformer to various different fields. Section 9 discusses some aspects of Transformer that researchers might find intriguing and summarizes the paper**.**

****

Taxonomy of Transformers A wide variety of models have been proposed so far based on the vanilla Transformer from three perspectives: types of architecture modification, pre-training methods, and applications. Fig. 2 gives an Illustrations of our categorization of Transformer variants. Fig. 3 illustrates our taxonomy and some representative models. In this survey, we focus on reviewing the works on architecture modifications. Since the attention module is the key component of Transformer, we solely describe the attention-related variants in Section 4 and introduce the other module-level variants in Section 5. Then Section 6 describes the other architecture-level variants. Finally, we briefly review the works on pre-training in Section 7 and applications in Section 8. There are some comprehensive surveys on the latter two categories of work, such as pre-trained models (PTMs) (Qiu et al., 2020) and visual Transformers (Han et al., 2021a; Khan et al., 2021). 4. Attention Self-attention plays an important role in Transformer, but there are two challenges in practical applications. 1. Complexity. As discussion in Section 2.3, the complexity of selfattention is (𝑇 2 ⋅ 𝐷). Therefore, the attention module becomes a bottleneck when dealing with long sequences. 2. Structural prior. Self-attention does no assume any structural bias over inputs. Even the order information is also needed to be learned from training data. Therefore, Transformer (w/o pretraining) is usually easy to overfit on small or moderate-size data.

The improvements on attention mechanism can be divided into several directions: 1. Sparse Attention. This line of work introduces sparsity bias into the attention mechanism, leading to reduced complexity. 2. Linearized Attention. This line of work disentangles the attention matrix with kernel feature maps. The attention is then computed in reversed order to achieve linear complexity. 3. Prototype and Memory Compression. This class of methods reduces the number of queries or key–value memory pairs to reduce the size of the attention matrix. 4. Low-rank Self-Attention. This line of work capture. the low-rank property of self-attention. 5. Attention with Prior. The line of research explores supplementing or substituting standard attention with prior attention distributions. 6. Improved Multi-Head Mechanism. The line of studies explores different alternative multi-head mechanisms. We will describe these attention variants at length in the rest of this section.

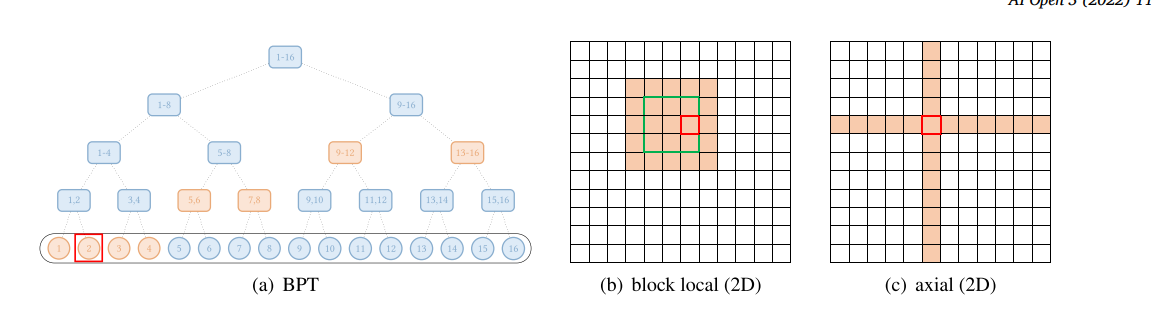


****Dilated Attention. Analogous to dilated CNNs (van den Oord et al., 2016), one can potentially increase the receptive field of the band attention without increasing computation complexity by using a dilated window with gaps of dilation 𝑤𝑑 ≥ 1, as depicted in Fig. 4(c). This can be easily extended to strided attention, where the window size is not limited but the dilation 𝑤𝑑 is set to a large value.

Random Attention. To increase the ability of non-local interactions, a few edges are randomly sampled for each query, as illustrated in Fig. 4(d). This is based on the observation that random graphs (e.g., Erdős–Rényi random graph) can have similar spectral properties with complete graphs that leads to a fast mixing time for random walking on graphs.

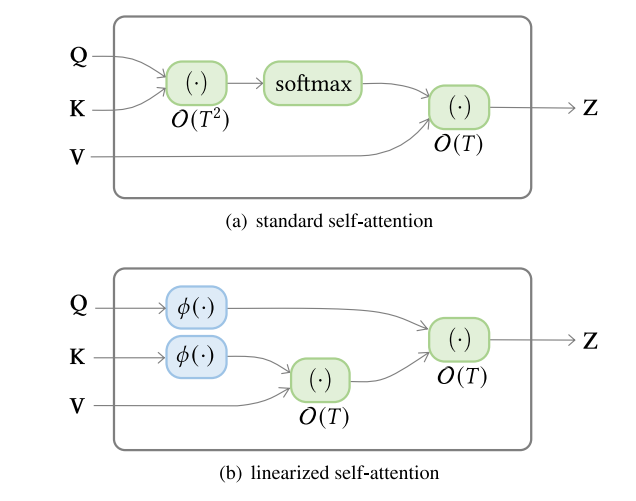
Block Local Attention. This class of attention segments input sequence into several non-overlapping query blocks, each of which is associated with a local memory block. All the queries in a query block attend to only the keys in the corresponding memory block. Fig. 4(e) depicts a commonly used case where the memory blocks are identical to their corresponding query block.

Compound sparse attention. Existing sparse attentions are often composed of more than one of the above atomic patterns. Fig. 5 illustrates some representative compound sparse attention patterns. Star-Transformer (Guo et al., 2019a) uses a combination of band attention and global attention. Specifically, Star-Transformer just includes only a global node and a band attention with the width of 3, in which any pair of non-adjacent nodes are connected through a shared global node and adjacent nodes are connected directly with each other. This kind of sparse pattern forms a star-shaped graph among nodes. Longformer (Beltagy et al., 2020) uses a combination of band attention and internal global-node attention.



. Extended sparse attention. Apart from the above patterns, some existing studies have explored extended sparse patterns for specific data types. For text data, BP-Transformer (Ye et al., 2019) constructs a binary tree where all tokens are leaf nodes and the internal nodes are span nodes containing many tokens. The edges in this graph are constructed so that each leaf node is connected to its neighbor leaf nodes and higher-level span nodes containing tokens from a longer distance. This approach can be seen as an extension of global attention, where global nodes are hierarchically organized and any pair of tokens are connected with paths in the binary tree. An abstract view of this method is illustrated in Fig. 6(a).

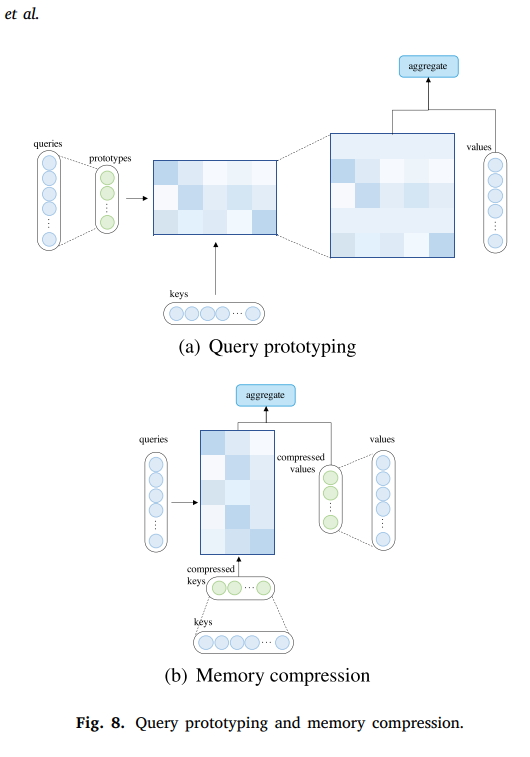
. Content-based sparse attention Another line of work creates a sparse graph based on input content, i.e., the sparse connections are conditioned on inputs. A straightforward way of constructing a content-based sparse graph is to select those keys that are likely to have large similarity scores with the given query. To efficiently construct the sparse graph, we can recur to Maximum Inner Product Search (MIPS) problem, where one tries to find the keys with maximum dot product with a query without computing all dot product terms. Routing Transformer (Roy et al., 2021) uses k-means clustering to cluster both queries {𝐪𝑖 } 𝑇 𝑖=1 and keys {𝐤𝑖 } 𝑇 𝑖= on the same set of centroid vectors {𝝁𝑖 } 𝑘 𝑖=1. Each query only attends to the keys that belong to the same cluster. During training, the cluster centroid vectors are updated using the exponentially moving average of vectors assigned to it, divided by the exponentially moving average of cluster counts: ̃



This is adopted by several studies (Katharopoulos et al., 2020; Choromanski et al., 2020a,b). However, it could be more beneficial for the network to selectively drop associations as new associations are added to the memory matrix. RFA (Peng et al., 2021) introduces a gating mechanism to the summation to model local dependency in sequence data. Specifically, when adding a new association to the memory matrix 𝐒, at a particular time step, they weigh 𝐒 by a learnable, input-dependent scalar 𝑔, and the new association by (1−𝑔) (and a similar mechanism to 𝐮). With this modification, history associations are exponentially decayed and recent context is favored in each timestep.

. Query prototyping and memory compression Apart from using sparse attention or kernel-based linearized attention, one could also reduce the complexity of attention by reducing the number of queries or key–value pairs, which leads to query prototyping and memory compression7 methods, respectively.

Attention with prototype queries In query prototyping, several prototypes of queries serve as the main source to compute attention distributions. The model either copies the distributions to the positions of represented queries or filling those positions with discrete uniform distributions. Fig. 8(a) illustrates the computing flow of query prototyping. Clustered Attention (Vyas et al., 2020) groups queries into several clusters and then computes attention distributions for cluster centroids. All queries in a cluster share the attention distribution calculated with the corresponding centroid. Informer (Zhou et al., 2021) selects prototypes from queries using explicit query sparsity measurement, which is derived from an approximation of the Kullback–Leibler divergence between the query’s attention distribution and the discrete uniform distribution. Attention distributions are then only calculated for the top-𝑢 queries under query sparsity measurement. The rest of the queries are assigned with discrete uniform distributions.



A blue and white squares

Description automatically generated

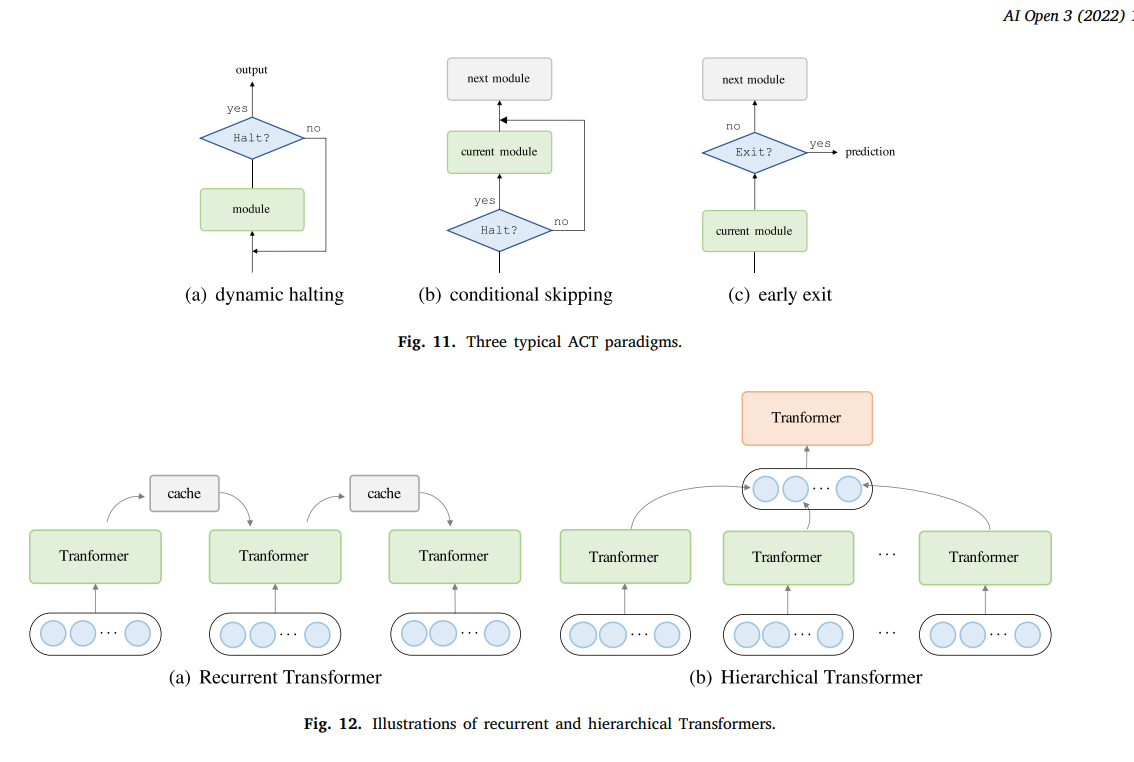
Low-rank parameterization The fact that the rank of the attention matrix is less than sequence length implies that, for scenarios where the inputs are typically short, setting 𝐷𝑘 > 𝑇 would be more than an over-parameterization and lead to overfitting. It is thus reasonable to limit the dimension of 𝐷𝑘 to explicitly model the low-rank property as an inductive bias. Guo et al. (2019b) decompose self-attention matrix into a low-rank attention module with small 𝐷𝑘 that captures long-range non-local interactions, and a band attention module that captures local dependencies.

. Low-rank approximation Another implication of the low-rank property of the attention matrix is that one can use a low-rank matrix approximation to reduce the complexity of self-attention. A closely related methodology is the lowrank approximation of kernel matrices. We believe some existing works are inspired by kernel approximation. Some of the aforementioned linearized attention methods in Section 4.2 are inspired from kernel approximation with random feature maps. For example, Performer (Choromanski et al., 2020a) follows the Random Fourier feature map originally proposed to approximate Gaussian kernels. The method first decomposes the attention distribution matrix 𝐀 into 𝐂𝑄𝐆𝐂𝐾 where 𝐆 is a Gaussian kernel matrix and the random feature map is used to approximate 𝐆. Another line of work follow the idea of Nyström method. These Nyström-based methods (Chen et al., 2020a; Xiong et al., 2021) first select 𝑚 landmark nodes from the 𝑇 inputs with down-sampling methods (e.g., strided average pooling). Let 𝐐̃ , 𝐊̃ be the selected landmark queries and keys, then the follow approximation is used in the attention computation.

Attention with prior Attention mechanism generally outputs an expected attended value as a weighted sum of vectors, where the weights are an attention distribution over the values. Traditionally, the distribution is generated from inputs (e.g., softmax(𝐐𝐊⊤) in vanilla Transformer). As a generalized case, attention distribution can also come from other sources, which we refer to as prior. Prior attention distribution can be a supplement or substitute for distribution generated from inputs. We abstract this formulation of attention as attention with prior, as depicted in Fig. 9. In most cases, the fusion of two attention distribution can be done by computing a weighted sum of the scores corresponding to the prior and generated attention before applying softmax.

Encoding structural information Attention mechanism is often interpreted as aggregating information from inputs based on weights generated from content similarity measures. However, it might be insufficient to base the weights solely on the content of input elements. Since the attention matrix encodes relations between inputs, it is natural to encode structural information using prior attention. For instance, one can use prior attention to encode positional relations between input elements. The prior attention can be formulated as a trainable attention prior 𝐁 that is added directly to the unnormalized attention matrix (Raffel et al., 2020) or generated from position embeddings (Ke et al., 2020). One can also use prior attention to model more complex structural information (e.g., edges in graph data). Graphormer (Ying et al., 2021a) utilizes a spatial encoding 𝐁 that encodes connectivity between nodes, as well as an edge encoding 𝐂 that encodes edge features.

Attention with only prior Some works have explored using an attention distribution that is independent of pair-wise interaction between inputs. In other words, their models exploit only a prior attention distribution. Zhang et al. (2018) design an efficient Transformer decoder variant called average attention network that uses a discrete uniform distribution as the sole source of attention distribution. The values are thus aggregated as a cumulative-average of all values. To improve the expressiveness of the network, they further adds a feed-forward gating layer on top of the average attention module. The advantage of this approach is that the adapted Transformer decoder can train in a parallel manner as usual Transformers do and decode like an RNN, thus avoiding the (𝑇 2 ) complexity in decoding.



Recurrent Transformers In recurrent Transformers, a cache memory is maintained to incorporate the history information. While processing a segment of text, the network reads from the cache as an additional input. After the processing is done, the network writes to the memory by simply copying hidden states or using more complex mechanisms. The abstract process is illustrated in Fig. 12(a).

Hierarchical Transformers Hierarchical Transformer decomposes inputs hierarchically into elements of finer granularity. Low-level features are first fed to a Transformer encoder, producing output representations that are then aggregated (using pooling or other operations) to form a high-level feature, which is then processed by a high-level Transformer. This class of methods can be understood as a process of hierarchical abstraction. The overview of this approach is depicted in Fig. 12(b). The advantages of this approach are twofold: (1) Hierarchical modeling allows the model to handle long inputs with limited resources; (2) It has the potential to generate richer representations that are beneficial to tasks.

Hierarchical for long sequence inputs. For tasks with inherently long input length, one can use hierarchical Transformers for effective modeling of long-range dependencies. For document-level machine translation tasks, Miculicich et al. (2018) introduce dependencies on the previous sentences from both the source and target sides when translating a sentence. They use an attention mechanism as the aggregation operation to summarize low-level information. For document summarization, HIBERT (Zhang et al., 2019) encodes a document of text by first learn sentence representations for all sentences and then use these sentence representations to encode document-level representations that are then used to generate the summary. The model uses the last hidden representation.

On the one hand, this effectively makes Transformer a very universal architecture that has the potential of capturing dependencies of different ranges. On the other hand, this makes Transformer prone to overfitting when the data is limited. One way to alleviate this issue is to introduce inductive bias into the model.

Recent studies suggest that Transformer models that are pre-trained on large corpora can learn universal language representations that are beneficial for downstream tasks (Qiu et al., 2020). The models are pre-trained using various self-supervised objectives, e.g., predicting a masked word given its context. After pre-training a model, one can simply fine-tune it on downstream datasets, instead of training a model from scratch. To illustrate typical ways of using Transformers in pre-training, we identify some of the pre-trained Transformers and categorize them as follows.

• Encoder only. A line of work uses the Transformer encoder as its backbone architecture. BERT (Devlin et al., 2019) is a representative PTM that is typically used for natural language understanding tasks. It utilizes Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) as the self-supervised training objective. RoBERTa (Liu et al., 2019a) further adapts the training of BERT and removes the NSP objective as it is found to hurt performance on downstream tasks. • Decoder only. Several studies focus on pre-training Transformer decoders on language modeling. For example, the Generative Pre-trained Transformer (GPT) series (i.e., GPT (Radford et al., 2018), GPT-2 (Radford et al., 2019), and GPT-3 (Brown et al., 2020)) is dedicated to scaling pre-trained Transformer decoders and has recently illustrated that a large-scale PTM can achieve impressive few-shot performance with the task and examples fed to the model as constructed prompts (Brown et al., 2020).

**Conclusion and future directions**

In this survey, we conduct a comprehensive overview of X-formers and propose a new taxonomy. Most of the existing works improve Transformer from different perspectives, such as efficiency, generalization, and applications. The improvements include incorporating structural prior, designing lightweight architecture, pre-training, and so on. Although X-formers have proven their power for various tasks, challenges still exist. Besides the current concerns (e.g. efficiency and generalization), the further improvements of Transformer may lie in the following directions: (1) Theoretical Analysis. The architecture of Transformer has been demonstrated to be capable of supporting large-scale training datasets with enough parameters. Many works show that Transformer has a larger capacity than CNNs and RNNs and hence has the ability to handle a huge amount of training data. When Transformer is trained on sufficient data, it usually has better performances than CNNs or RNNs. An intuitive explanation is that Transformer has few prior assumptions on the data structure and therefore is more flexible than CNNs and RNNs. However, the theoretical reason is unclear and we need some theoretical analysis of Transformer ability. (2) Better Global Interaction Mechanism beyond Attention. A main advantage of Transformer is the use of the attention mechanism to model the global dependencies among nodes within input data. However, many studies have shown that full attention is unnecessary for most nodes. It is, to some degree, inefficient to indistinguishably calculate attention for all nodes. Therefore, there is still plenty of room for improvements in efficiently modeling global interactions. On the one hand, the self-attention module can be regarded as a fully-connected neural network with dynamical connection weights, which aggregates non-local information with dynamic routing. Therefore, other dynamic routing mechanisms are alternative approaches worth exploring. On the other hand, the global interaction can also be modeled by other types of neural networks, such as memory-enhanced models. (3) Unified Framework for Multimodal Data. In many application scenarios, integrating multimodal data is useful and necessary to boost the task performance. Moreover, the general AI also needs the ability to capture the semantic relations across different modalities. Since Transformer achieves great success on text, image, video, and audio, we have a chance to build a unified framework and better capture the inherent connections among multimodal data. However, the design of the intra-modal and cross-modal attention still remains to be improved. Finally, we wish this survey to be a hands-on reference for better understanding the current research progress on Transformers and help readers to further improve Transformers for various applications.

**References**

Ainslie, Joshua, Ontanon, Santiago, Alberti, Chris, Cvicek, Vaclav, Fisher, Zachary, Pham, Philip, Ravula, Anirudh, Sanghai, Sumit, Wang, Qifan, Yang, Li, 2020. ETC: Encoding long and structured inputs in transformers. In: Proceedings of EMNLP. pp. 268–284. http://dx.doi.org/10.18653/v1/2020.emnlp-main.19, Online. Al-Rfou, Rami, Choe, Dokook, Constant, Noah, Guo, Mandy, Jones, Llion, 2019. Character-level language modeling with deeper self-attention. In: Proceedings of AAAI. pp. 3159–3166. http://dx.doi.org/10.1609/aaai.v33i01.33013159. Arnab, Anurag, Dehghani, Mostafa, Heigold, Georg, Sun, Chen, Lučić, Mario, Schmid, Cordelia, 2021. Vivit: A video vision transformer. arXiv:2103.15691. Ba, Lei Jimmy, Kiros, Jamie Ryan, Hinton, Geoffrey E., 2016. Layer normalization. CoRR arXiv:1607.06450. Bachlechner, Thomas, Majumder, Bodhisattwa Prasad, Mao, Huanru Henry, Cottrell, Garrison W., McAuley, Julian J., 2020. Rezero is all you need: Fast convergence at large depth. CoRR arXiv:2003.04887. Baevski, Alexei, Auli, Michael, 2019. Adaptive input representations for neural language modeling. In: Proceedings of ICLR. URL https://openreview.net/forum?id= ByxZX20qFQ. Bapna, Ankur, Arivazhagan, Naveen, Firat, Orhan, 2020. Controlling computation versus quality for neural sequence models. arXiv:2002.07106, [cs.LG]. Bapna, Ankur, Chen, Mia, Firat, Orhan, Cao, Yuan, Wu, Yonghui, 2018. Training deeper neural machine translation models with transparent attention. In: Proceedings of EMNLP. Brussels, Belgium, pp. 3028–3033. http://dx.doi.org/10.18653/v1/D18- 1338. Battaglia, Peter W., Hamrick, Jessica B., Bapst, Victor, Sanchez-Gonzalez, Alvaro, Zambaldi, Vinicius, Malinowski, Mateusz, Tacchetti, Andrea, Raposo, David, Santoro, Adam, Faulkner, Ryan, Gulcehre, Caglar, Song, Francis, Ballard, Andrew, Gilmer, Justin, Dahl, George, Vaswani, Ashish, Allen, Kelsey, Nash, Charles, Langston, Victoria, Dyer, Chris, Heess, Nicolas, Wierstra, Daan, Kohli, Pushmeet, Botvinick, Matt, Vinyals, Oriol, Li, Yujia, Pascanu, Razvan, 2018. Relational inductive biases, deep learning, and graph networks. arXiv:1806.01261, [cs.LG]. Beltagy, Iz, Peters, Matthew E., Cohan, Arman, 2020. Longformer: The long-document transformer. arXiv:2004.05150, [cs.CL]. Bhojanapalli, Srinadh, Yun, Chulhee, Rawat, Ankit Singh, Reddi, Sashank J., Kumar, Sanjiv, 2020. Low-rank bottleneck in multi-head attention models. In: Proceedings of ICML. pp. 864–873, URL http://proceedings.mlr.press/v119/ bhojanapalli20a.html. Brown, Tom, Mann, Benjamin, Ryder, Nick, Subbiah, Melanie, Kaplan, Jared D, Dhariwal, Prafulla, Neelakantan, Arvind, Shyam, Pranav, Sastry, Girish, Askell, Amanda, Agarwal, Sandhini, Herbert-Voss, Ariel, Krueger, Gretchen, Henighan, Tom, Child, Rewon, Ramesh, Aditya, Ziegler, Daniel, Wu, Jeffrey, Winter, Clemens, Hesse, Chris, Chen, Mark, Sigler, Eric, Litwin, Mateusz, Gray, Scott, Chess, Benjamin, Clark, Jack, Berner, Christopher, McCandlish, Sam, Radford, Alec, Sutskever, Ilya, Amodei, Dario, 2020. Language models are few-shot learners. In: Proceedings of NeurIPS. pp. 1877–1901, URL https://proceedings.neurips.cc/paper/ 2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf. Carion, Nicolas, Massa, Francisco, Synnaeve, Gabriel, Usunier, Nicolas, Kirillov, Alexander, Zagoruyko, Sergey, 2020. End-to-end object detection with transformers. In: Proceedings of ECCV. pp. 213–229. http://dx.doi.org/10.1007/978-3-030-58452- 8\_13. Chen, Ziye, Gong, Mingming, Ge, Lingjuan, Du, Bo, 2020a. Compressed self-attention for deep metric learning with low-rank approximation. In: Proceedings of IJCAI. pp. 2058–2064. http://dx.doi.org/10.24963/ijcai.2020/285. Chen, Mark, Radford, Alec, Child, Rewon, Wu, Jeffrey, Jun, Heewoo, Luan, David, Sutskever, Ilya, 2020b. Generative pretraining from pixels. In: Proceedings of ICML. pp. 1691–1703, URL http://proceedings.mlr.press/v119/chen20s.html.