What is attention mechanism? A comprehensive survey of attention methods and transformer models

Farhad Mortezapour Shiri [1*], Fateme Memar [2], Maryam Parhizgar [3]

[1]*Faculty of Computer Science and Information Technology, University Putra Malaysia (UPM), Serdang, Malaysia*
gs63904@student.upm.edu.my

[2]*Department of electrical engineering and computer science, University of Kansas, Lawrence, Kansas, USA*

amemar@ku.edu

[3]*Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Qazvin, Iran*

Parhizgar.m92@gmail.com

Abstract

The attention mechanism is a fundamental component widely used in deep learning models across numerous domains and tasks. It enables models to selectively focus on the most relevant parts of the input, rather than processing all elements equally, by assigning weights according to their importance for the task. This paper presents a comprehensive overview of the attention mechanism, outlining its general framework and offering a taxonomy of attention models, including Hierarchical, Bidirectional, Multi-Head, Multi-query, Group-query, Graph, Channel, Spatial, Channel-Spatial, Temporal, Spatial-Temporal, Cross, Axial and Flash Attention. The computational trade-offs of these models are also analyzed, and new developments in efficient attention are highlighted. We also discuss the Transformer architecture, one of the most influential deep learning frameworks that makes effective use of attention mechanism and review its major variants including BERT, GPT, Transformer XL, XLNet, BART, Fast Transformer, T5, Longformer, BIGBIRD, Performer, Linformer, Reformer, RoFormer, ALiBi, Switch Transformer, LLaMA, ViT, DETR, DeepViT, DeiT, T2T ViT, CrossViT, PVT, Swin Transformer, TNT, MViT, ViViT, DAT, and Spiking Transformer. Additionally, we provide a comparative overview of these models, highlighting their key ideas, results, advantages, and limitations. Through this survey, we emphasized the pivotal role of attention-based models, especially those built on the Transformer architecture, in shaping diverse application domains such as natural language processing, computer vision, recommender systems, and sensor data analysis.

Keywords: Attention Mechanism, Transformer, Deep Learning, Computer Vision, Natural Language Processing (NLP).

1. Introduction

With advancements in hardware processing capabilities and the evolution of deep learning techniques, artificial intelligence (AI) is undergoing a significant transformation, enhancing performance across various domains such as image recognition, machine translation, language modeling, and time series forecasting [1]. Deep Learning (DL) refers to the method of acquiring hierarchical data representations through neural network architectures composed of multiple hidden layers, contributing to the network's overall depth [2, 3]. Among the most widely adopted DL models are Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), both of which have demonstrated considerable success in modeling, analyzing, and predicting complex systems [4]. RNNs are especially adept at handling sequential data due to their ability to account for the temporal order of inputs [5]. Nevertheless, traditional deep learning models are not inherently capable of sequential reasoning, a cognitive process that relies on perception guided by attention. In the human brain, attention mechanisms facilitate reasoning by enabling focus on specific parts of the input or memory (such as images or text), while de-emphasizing others [6].

Attention is a concept that has been the focus of extensive public discourse and rigorous academic research. It carries various definitions across different disciplines such as psychology, neuroscience, and more recently, machine learning [7]. Attention is inherently connected to the human cognitive system. Cognitive science suggests that the optic nerve delivers an overwhelming amount of sensory data, more than the brain can fully process. Therefore, the brain evaluates this input and selectively focuses on only the most relevant information [8]. This mechanism significantly enhances both the speed and precision of sensory data interpretation. A fundamental property of attention is its vital role in optimizing information processing, both in the brain and, more

* Corresponding author

recently, in artificial systems. The capability to dynamically redirect and modulate information flow offers significant advantages in making a system more adaptive [9].

The concept of replicating human attention was first explored in the domain of computer vision [10-14]. Drawing from the visual attention mechanism, researchers have aimed to develop models that simulate human visual attention, allowing them to predict how people allocate their focus across images and videos, and extend this understanding to various applications [15]. One such system, inspired by the foveal structure of the human retina, was designed for image classification [10]. It integrates a restricted Boltzmann machine (RBM) with a specialized attentional module known as the fixation controller. Another study introduced a novel visual attention framework in [12], which is implemented as a recurrent neural network that processes input via a glimpse window, and utilizes its internal state to determine subsequent focal points and produce control signals in changing environments. Moreover, a probabilistic graphical model for learning generative models through attention was proposed by [14]. By excluding irrelevant background elements, the generative model is able to dedicate its capacity to the primary object of focus.

Although the original development of attention mechanisms began in computer vision, the modern recognition of attention's impact is often attributed to the field of natural language processing, largely due to four influential studies [16].

The first of these was presented by Bahdanau et al. [17], who introduced a novel method for neural machine translation. Their approach incorporated attention into a machine translation framework to address some structural challenges of recurrent neural networks.

The second key study, by Luong et al. [18], proposed two straightforward yet effective attention mechanisms for neural machine translation: one global strategy that considers all source words simultaneously, and a local strategy that focuses on a limited subset of source words at any given time.

The third important advancement introduced a self-attention mechanism, sometimes called intra-attention, within a neural architecture known as the Long Short-Term Memory-Networks (LSTMN) [19].

The fourth and most transformative work was the Transformer model, introduced by Vaswani et al. [20] in 2017. Initially designed for machine translation, this architecture has since become a cornerstone of modern deep learning, particularly in the field of NLP. The Transformer relies on a self-attention-based encoder-decoder structure, where both the encoder and decoder are composed of a sequence of identical layers. Each layer includes two main components: the first is a multi-head self-attention mechanism, and the second is a position-wise fully connected feed-forward network [21].

Most attention-based architectures can be trained jointly with a base model such as a recurrent neural network or a convolutional neural network using standard backpropagation. However, the introduction of the Transformer marked a major milestone, as it showed that attention mechanisms alone could build top-performing models [16]. This innovation offered a way around limitations like the inherent difficulty of parallelizing recurrent networks. Similar to the initial application of attention in [17], the Transformer was designed for machine translation but quickly found utility in various other domains including image processing [22-24], video processing [25-27], recommender systems [28, 29], and others.

This survey covers attention mechanisms and Transformer models published between 2014 and 2025, tracing the evolution from the foundational additive attention [17] to recent advances such as Flash Attention 3 and LLaMA-based architectures. Models were selected for inclusion based on four criteria applied in combination: (1) Architectural novelty, the model introduces a new attention mechanism, positional encoding strategy, head configuration, or structural scope pattern not already covered by another entry in the survey. (2) Demonstrated impact, The model has achieved significant recognition in the research community, evidenced by substantial citation counts, wide adoption in downstream work, or deployment in large-scale practical systems. (3) Domain coverage, Collectively, the selected models span the four application domains examined in Section 5 (natural language processing, computer vision, recommender systems, and sensor data analysis), ensuring the survey is representative across domains rather than biased toward a single application area. (4) Publication and verifiability, each included model has sufficient published technical documentation to permit accurate, verifiable, and citation-supported description.

Models were excluded if they are purely proprietary with no public architectural details, or if they constitute minor implementation variants without substantive novelty relative to an already-included method. Applying these criteria, Section 3 covers 36 attention methods drawn from seven distinct types, while Section 4 reviews 29 Transformer architectures that represent the major branches of the NLP and computer vision Transformer literature.

This research aims to present a comprehensive framework for attention mechanisms and provide an extensive survey of various attention methods and Transformer models employed in deep learning. Although several existing surveys address attention mechanisms or Transformer architectures, none provides the integrated, up-to-date coverage offered by the present work. Most survey articles in this field focus on attention methods and Transformer models within specific domains, such as medical image analysis [1, 30], complex systems [6], intrusion detection systems [31] fault diagnosis of machinery [32], image captioning [33], speech recognition [34], and renewable energy forecasting [35]. While valuable within their respective areas, these domain-specific surveys do not offer a generalized, cross-disciplinary perspective on attention and Transformer architectures.

Other works adopt a narrower technical focus. For instance, [36] and [8] concentrate exclusively on visual attention in deep learning, thereby leaving NLP-centric Transformer families, vision-centric Transformers, and positional encoding strategies

unaddressed. Meanwhile, articles such as [37] and [16] limit themselves to the fundamental concepts and historical evolution of attention mechanisms and Transformers, without delving into recent architectural innovations.

A few surveys, including [38], and [9], attempt a more comprehensive, domain-agnostic review. However, their analyses are largely confined to early attention methods and older Transformer models, offering limited insight into more recent developments. Similarly, [16] present a useful but limited taxonomy of attention mechanisms, yet they provide minimal coverage of the latest Transformer families and efficient attention variants. Likewise, [39] deliver a thorough review at the mechanistic level but fall short of examining complete Transformer architectural families and their comparative trade-offs across different application domains.

Taken together, it is fair to conclude that the most comprehensive surveys in this field are becoming outdated and constrained in scope. Given the rapid growth of attention mechanisms and Transformer architectures in recent years, and their increasing necessity across various fields of artificial intelligence and machine learning, there is a clear and pressing need for a new, comprehensive survey. Such a survey should not only cover major concepts but also provide an analysis of nearly all common and useful attention methods and Transformer models for researchers and practitioners alike.

The present survey fills these gaps in three specific ways. First, it provides a unified, current treatment of both attention mechanisms and Transformer architectures within a single coherent framework. Coverage extends through developments up to and including 2025, encompassing models and variants that are absent from all prior surveys. This ensures that readers gain access to the most recent advances in the field, including state-of-the-art efficient attention mechanisms, and emergent Transformer variants. By treating attention and Transformers together rather than as separate topics, the survey reveals conceptual continuities and design patterns that are often lost when these subjects are reviewed in isolation.

Second, it covers and classifies useful articles that employ novel attention mechanisms and Transformer models, organizing them according to both the specific tasks they address and the methodological approaches they adopt across a wide range of applications. Importantly, this survey explicitly incorporates sensor data analysis as an application domain, a perspective that remains significantly underrepresented in existing attention surveys, alongside the more commonly covered areas of natural language processing (NLP), computer vision, and recommender systems. By doing so, it broadens the traditional scope of attention-related reviews and offers researchers from emerging fields such as robotic, human activity recognition (HAR), environmental monitoring, and predictive maintenance a valuable entry point into attention-based architectures.

Third, it provides a structured comparative synthesis spanning more than 35 Transformer and attention models. For each model, the synthesis distills key ideas, typical applications, strengths, limitations, and benchmark results, thereby offering readers practical, actionable guidance for model selection in real-world tasks. Together, these three contributions position this survey as a more comprehensive and current reference than any existing work for researchers and practitioners working across attention-based deep learning. It not only fills the gaps left by prior domain-specific or outdated surveys but also equips its audience with both a broad conceptual understanding and a practical tool for navigating the rapidly expanding landscape of attention and Transformer models.

The key contributions of this study are as follows:

1- Conceptual Overview: We discuss the concept of attention mechanisms, offering a brief historical perspective on the development of vanilla attention methods designed to enhance the performance of deep learning models.

2- Survey of Attention Methods: We review a wide range of attention techniques, including Hierarchical, Bidirectional, Multi-Head, Multi-query, Group-query, Graph, Channel, Spatial, Channel-Spatial, Temporal, Spatial-Temporal, Cross, Axial and Flash Attention, along with their respective sub-models.

3- Comprehensive Review of Transformer Variants: We provide an in-depth examination of major Transformer architectures, including BERT, GPT, Transformer XL, XLNet, BART, Fast Transformer, T5, Longformer, BIGBIRD, Performer, Linformer, Reformer, RoFormer, ALiBi, Switch Transformer, LLaMA, ViT, DETR, DeepViT, DeiT, T2T ViT, CrossViT, PVT, Swin Transformer, TNT, MViT, ViViT, DAT, and Spiking Transformer. This review highlights their architectural innovations, distinctive features, and applications across diverse domains.

4- Applications Across Domains: We illustrate the impact of attention mechanisms and Transformer-based models in advancing multiple fields, including natural language processing, computer vision, recommender systems, and sensor data analysis.

5- Comparative Analysis: We provide a comparative evaluation of various attention mechanisms and Transformer architectures, emphasizing their core principles, significant findings, unique strengths, and notable limitations.

The structure of the paper is as follows: Section 2 presents a general attention framework to provide foundational insights into its core principles and applications. Section 3 categorizes the different types of attention models. Section 4 is the classification of the various Transformer models. Applications of attention mechanisms and Transformer models in different tasks are investigated in Section 5. Section 6 provides a comparative analysis of diverse attention mechanisms and Transformer architectures. Research directions and future aspects are covered in Section 7. The article concludes with Section 8.

2. Attention mechanism

In the human brain, reasoning involves validating and forming conclusions by integrating attention with either new or pre-existing

information. Similarly, in deep learning models, attention mechanisms serve to highlight a particular portion of the input or memory (such as an image or text), thereby directing the reasoning process [6]. These mechanisms enable a single deep learning model to efficiently handle various tasks, including those with inputs that differ in length, size, or structure [7].

In a general sense, attention in machine learning is structured as a sequential process, where the learning task is steered by selected elements from the input source or memory. This is accomplished by embedding the attention value into the task. Attention mechanisms represent and will continue to represent a significant paradigm shift in machine learning. In particular, this shift moves away from the conventional use of large-scale vector transformations and toward more deliberate processes that selectively attend to subsets of elements, such as breaking down a complex task into a sequence of reasoning steps guided by attention [6].

As mentioned earlier, the earliest successful implementation of attention mechanisms was the additive attention proposed by [17]. Their approach incorporated attention into a machine translation model to resolve certain limitations associated with the architecture of recurrent neural networks.

Neural machine translation models are typically structured as encoder-decoder architectures, where each language is assigned its own encoder and decoder [40]. However, a key limitation of the basic encoder-decoder approach lies in the requirement to compress the entire content of a source sentence into a single fixed-length vector, which often hinders the model's ability to fully capture the semantic richness of long sentences [37].

Bahdanau et al. [17] enhanced the basic encoder–decoder model by allowing the decoder to perform a (soft-) search through a set of input words or their encoder-derived representations when generating each word in the target language. This adjustment eliminates the need to encode the entire source sentence into one fixed vector and instead permits the model to selectively focus on information that is directly relevant to producing the next word. This significantly improves the model's performance on longer sequences. Their machine translation architecture includes a bidirectional recurrent neural network (BiRNN) as the encoder, while the decoder is composed of an attention module alongside a recurrent neural network (RNN). A visual representation of this model is provided in Fig. 1.

The encoder calculates annotations $(\mathbf{h_1}, \ldots., \mathbf{h_T})$ that are the hidden state of the BiRNN based on input sequence $(x_1, \ldots., x_T)$:

$$(\boldsymbol{h_1}, \ldots., \boldsymbol{h_T}) = BiRNN(x_1, \ldots., x_T) \quad (1)$$

In particular, shown in Fig. 1, hidden states $\{\overrightarrow{\boldsymbol{h_1}}, \ldots. \overrightarrow{\boldsymbol{h_T}}\}$ and $\{\overleftarrow{\boldsymbol{h_1}}, \ldots. \overleftarrow{\boldsymbol{h_T}}\}$ are the output of the forward RNN and the backward RNN respectively. The encoder produces an annotation for one word x_i by simply concatenating the forward hidden state $\overrightarrow{\boldsymbol{h_i}}$ and the backward hidden state $\overleftarrow{\boldsymbol{h_i}}$.

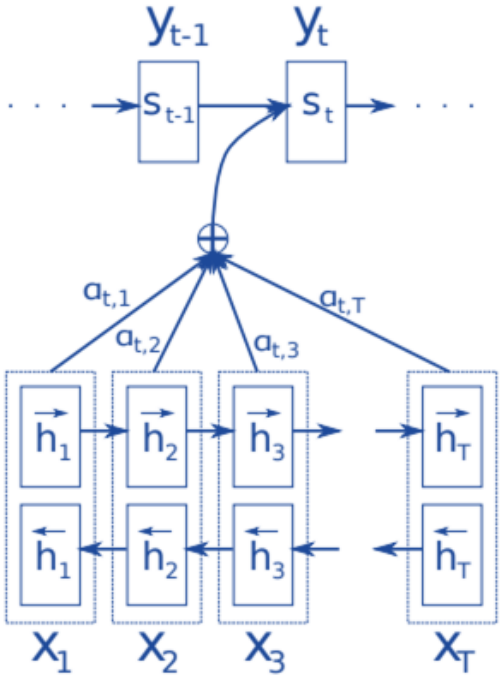

Fig. 1. A depiction of a single step of decoding in attention-based neural machine translation [17].

The decoder is composed of an attention block and recurrent neural network (RNN). The function of the attention block is to compute a context vector c that encodes the context relationships between the current output symbol and each term of the input sequence. The context vector c_t is computed as a weighted sum of these annotations $\boldsymbol{h_j}$:

$$c_t = \sum_{j=1}^{T} \alpha_{tj} h_j \tag{2}$$

The weight α_{tj} of each annotation h_j is derived using

$$\begin{cases} \alpha_{tj} = \dfrac{\exp(e_{tj})}{\sum_{k=1}^{T} \exp(e_{tk})} \\ where \\ e_{tj} = a(s_{t-1}, h_j) \end{cases} \tag{3}$$

Here a is a learnable function and represents the relevance of $\boldsymbol{h_j}$ to the next hidden state $\boldsymbol{s_i}$ given the previous state $\boldsymbol{s_{i-1}}$.

After [17] applied attention models to machine translation problems, multiple variations of attention models for different application areas have sprang up rapidly.

Luong et al. [18] proposed two simple and effective attentional schemes for use in neural machine translation: a global approach that attends all source positions at all times, and a local approach that attends to only a subset of source positions at one time. What is common to both of these models is that at each decoding step t, both models take the hidden state h_t at the top layer of a stacked LSTM (Long Short-Term Memory) [41]. The objective is then to build a context vector c_t, which extracts the relevant source-side information to assist in predicting the current target word y_t. Although their approaches differ in how the context vector c_t, is developed, their subsequent usage is the same. Specifically, to create an attentional hidden state, it combines the hidden state h_t from the decoder with the source-side context vector c_t, a simple concatenation layer is employed to combine the information from both vectors to generate the attentional hidden state:

$$\tilde{h}_t = tanh(W_c[c_t; h_t]) \tag{4}$$

The attentional vector $\tilde{h}_t$ is then passed through the softmax layer to give the predictive distribution defined by:

$$p(y_t \mid y_{\{<t\}}, x = softmax(W_s \tilde{h}_t) \tag{5}$$

We now describe how each model type computes the context vector on the source side c_t.

The main idea of a global attentional model is that c_t is computed through consideration of all the hidden states of the encoder. Fig. 2 shows the Global attentional model.

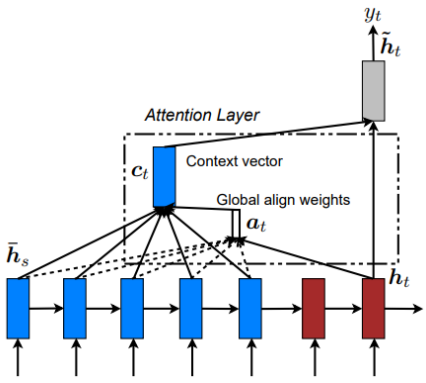

Fig. 2. Global attention model [18].

This model type consists of an alignment vector a_t, of variable length that is equal to the number of time steps on the source side. The alignment vector a_t is obtained from comparing the current hidden state on the target side h_t to each hidden state on the source side, $\bar{h}_s$:

$$a_{t(s)} = align(h_t, \bar{h}_s) = \frac{exp\left(score(h_t, \bar{h}_s)\right)}{\sum_s' exp\left(score(h_t, \bar{h}_s')\right)} \tag{6}$$

Luong et al. [18] use three different content-based functions as alternatives, and a location-based function was used for scoring.

$$score(h_t, \bar{h}) = \begin{cases} h_t^T \bar{h}_s & dot \\ h_t^T W_a \bar{h}_s & general \\ v_a^T tanh(W_a[h_t; \bar{h}_s]) & concat \end{cases} \tag{7}$$

$$a_t = softmax(W_a h_t) \tag{8}$$

Global attention has the disadvantage of having to attend to every word on the source side for each target word, which is expensive and may make it implausible to translate longer sequences, e.g., paragraphs or documents. To overcome this limitation, a local attentional mechanism that chooses to focus only on a small subset of the source positions per target word is proposed. The local attentional model is shown in Fig. 3.

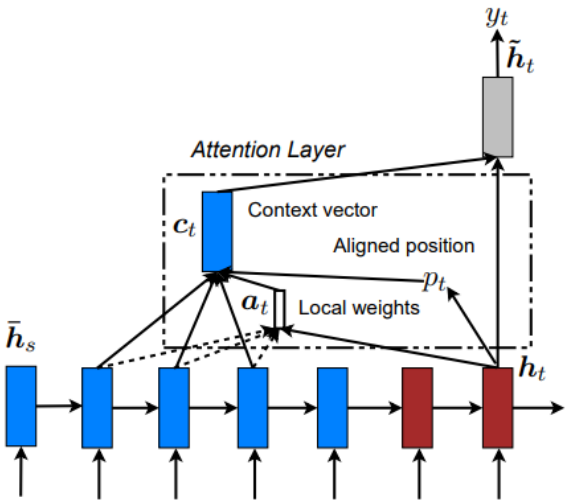

Fig. 3. Local attention model [18].

This model initially predicts a unique aligned position p_t for the target word at the current time, which in turn defines a window centered on the source position p_t to calculate a context vector c_t a weighted average of the source hidden states within the window. The weights at are functions of the target state h_t and of the source states $\bar{h}_s$ within the window. The two forms of the local model are outlined below.

Monotonic alignment (local-m): it is the same as the global model except the vector at is fixed-length and shorter. It sets $p_t = t$, given that the source and target sequences are approximately monotonically aligned. The alignment vector at is set as per Eq. (6).

Predictive alignment (local-p): rather than monotonic alignments being assumed, this model predicts an aligned position as follows:

$$p_t = S.sigmoid\left(v_p^T tanh(W_p h_t)\right) \tag{9}$$

w_p and v_p are the model parameters that will be learned to predict positions. S is the length of the source sentence. As the

result of sigmoid, $p_t \in [0, S]$. To favor alignment points near p_t, a Gaussian distribution centered around p_t is placed. Specifically, alignment weights are now defined as:

$$a_t(s) = align(h_t, \bar{h}_s) exp\left(-\frac{(s-p_t)^2}{2\sigma^2}\right) \quad (10)$$

Align function is the same as in Eq. (6) and the standard deviation is set empirically as $\sigma = \frac{D}{2}$. Note that p_t is a real number; while s is an integer inside the window centered at p_t. Using p_t to calculate a_t, we can backpropagate gradients for W_p and v_p. This model is differentiable almost everywhere.

Cheng et al. [19] propose a self-attention with a modified LSTM unit, namely the Long Short-Term Memory-Network (LSTMN). The LSTMN substitutes the memory cell with a memory network to enable the storage of the contextual representation of every input token with a distinct memory slot, and the memory size increases over time until an upper limit of the memory span. The design allows the LSTM to reason over relations between tokens using a neural attention layer and then conduct non-Markov state updates. The architecture of the LSTMN is shown in Fig. 4 and the formal definition is provided as follows.

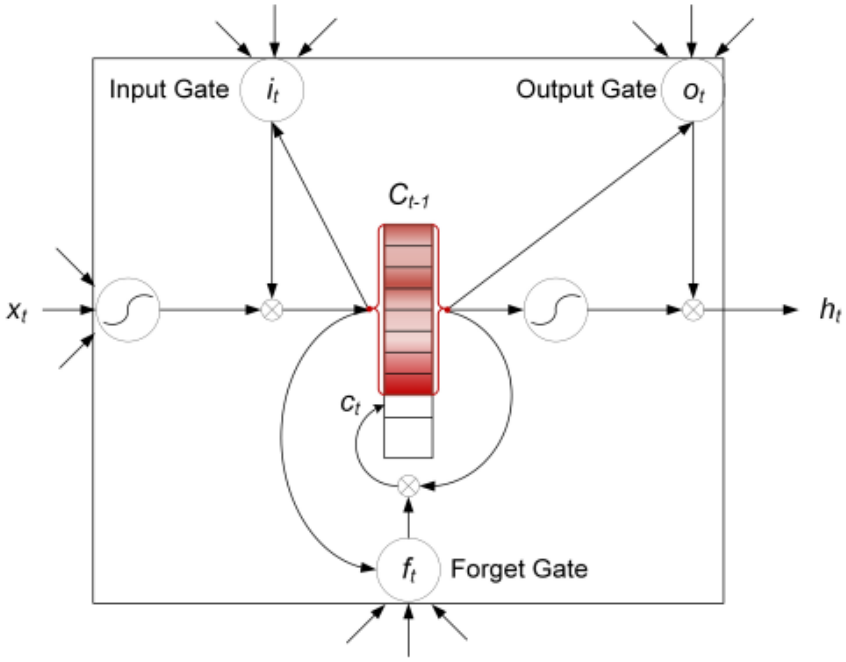

Fig. 4. Long Short-Term Memory-Network. (Color indicates degree of memory activation) [19].

The model has two sets of vectors retained in a hidden state tape employed to interact with the environment (e.g., calculating attention), and a memory tape to represent what is really stored in memory. Thus, every token has a hidden vector and a memory vector. Let x_t be the input at the current time; $C_{t-1} = (c_1, \cdots, c_{t-1})$ be the current memory tape, and $H_{t-1} = (h_t, \cdots, h_{t-1})$ the preceding hidden tape. At time t, the model calculates the relationship between x_t and $x_1 \cdots x_{t-1}$ through $h_1 \cdots h_{t-1}$ with an attention layer:

$$a_i^t = v^T tanh\left(W_h h_i + W_x x_t + W_{\tilde{h}} \tilde{h}_{t-1}\right) \quad (11)$$
$$s_i^t = softmax(a_i^t) \quad (12)$$

This gives us a probability distribution over the hidden state vectors of the past tokens. We can then calculate an adaptive summary vector for the past hidden tape and memory tape represented by $\tilde{c}_t$ and $\tilde{h}_t$, respectively:

$$\begin{bmatrix} \tilde{h}_t \\ \tilde{c}_t \end{bmatrix} = \sum_{i=1}^{t-1} S_i^t . \begin{bmatrix} h_i \\ c_i \end{bmatrix} \quad (13)$$

and utilize them for calculating the values of c_t and h_t in the recurrent update as:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ \hat{c}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} W.\left[\tilde{h}_t, x_t\right] \tag{14}$$

$$c_t = f_t \odot \tilde{c}_t + i_t \odot \hat{c}_t \tag{15}$$

$$h_t = o_t \odot tanh(c_t) \tag{16}$$

where v, W_h, W_x and $W_{\tilde{h}}$ are the new weight terms of the network.

One of the central notions of the LSTMN is to utilize attention for imposing relations among tokens. Such relations are soft and differentiable, and parts of a larger representation learning network.

In another study, Vaswani et al. [20] introduced the Transformer model based on the attention mechanism, specifically the scaled dot-product attention, which has since become fundamental for many subsequent attention-based models in deep learning. Dot-product attention is a fast and efficient form of content-based attention. Scaled Dot-product attention consists of three parts: a learned key matrix, a value matrix, and a query vector where the similarity between queries and keys is computed using a dot product. The attention function is computed on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. Matrix of outputs computed as:

$$Attention\ (Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{17}$$

The first step in scaled dot-product attention is to transform the input data into an embedding vector and the three vectors of query vector (Q), key vector (K), and value vector (V) are then extracted from the embedding vectors. A score is then calculated for each vector: score $=Q \cdot K$. Score normalization (dividing by $\sqrt{d_k}$) is applied for gradient stability. Then the score is passed through the softmax activation function. The weighted score v for each input vector is derived by taking the softmax dot product value v. The output is generated after summation. Scaled dot-product attention is illustrated in Fig. 5.

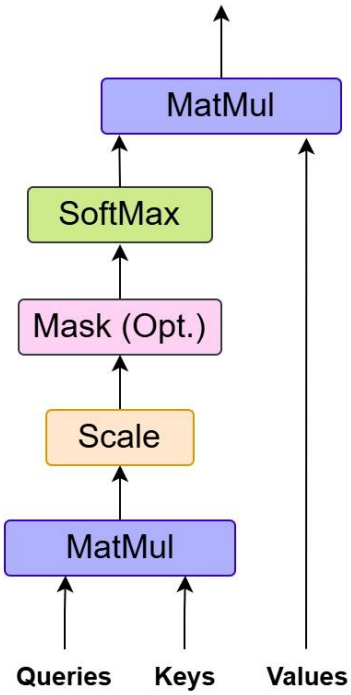

Fig. 5. Scaled Dot-Product Attention.

3. Various Attention Methods

Attention mechanism, as a technique to enhance the information processing capacity of neural networks, can be integrated into a wide range of models across different areas of deep learning. While the underlying principle of attention remains consistent, researchers have introduced various adjustments and enhancements to tailor these mechanisms more effectively to the requirements of specific tasks. There is no standard or unified classification for attention methods, as these techniques are numerous and can be grouped in various ways. In this article, we aim to investigate the most common and useful among them. Additionally, we propose a reasonable taxonomy for the methods used in this survey. Fig. 6 presents a visual taxonomy of the attention methods covered in this article.

We first examine foundational methods, including hierarchical attention and bidirectional attention. Next, we analyze attention methods extended for use in the Transformer core. These include multi-head attention (MHA), introduced in the standard Transformer in 2017, as well as multi-query attention (MQA) and group-query attention (GQA), introduced in 2019 and 2023, respectively. Following that, we discuss graph attention methods, beginning with the Graph Attention Network (GAT) introduced in 2017. In the continue, we explore the most common channel and spatial attention methods, which are widely used in computer vision tasks such as image classification and segmentation. Subsequently, we investigate temporal attention methods, which are commonly applied in time-series tasks. These include methods that rely solely on temporal attention, as well as those that incorporate both spatial and temporal attention, the latter being more frequently used in video analysis. Finally, we examine attention approaches designed to reduce memory consumption, following our discussion of cross-attention methods.

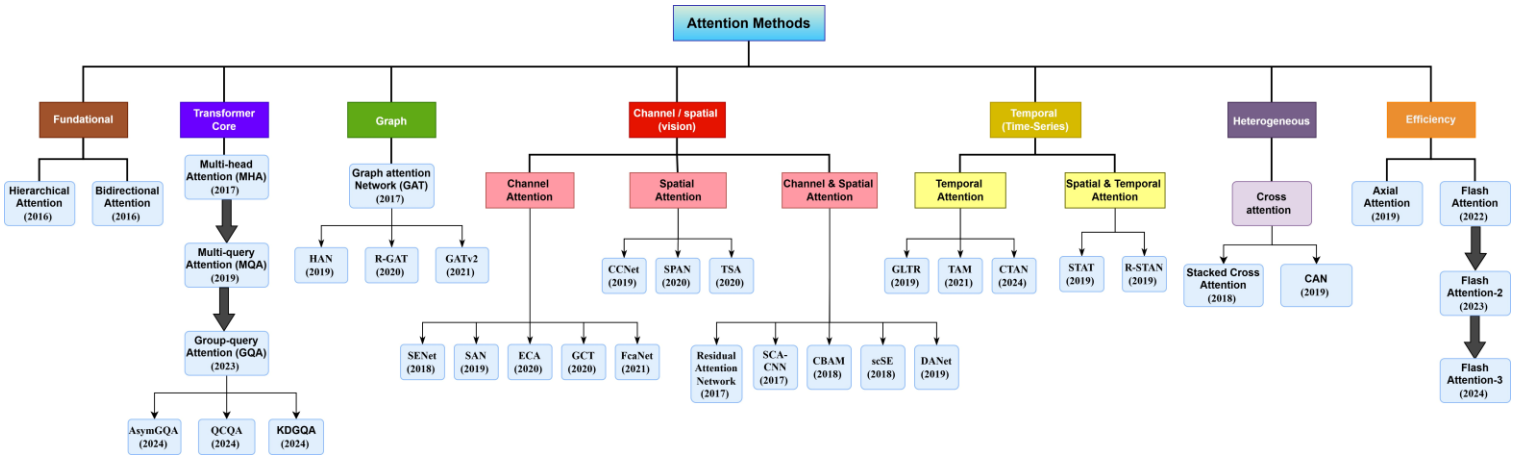

Fig. 6. Visual taxonomy of attention methods.

3.1. Hierarchical Attention

The Hierarchical Attention Network was initially introduced by Yang et al. [42] in 2016 for the task of document classification. The complete structure of the Hierarchical Attention Network (HAN) is depicted in Fig. 7. This architecture is composed of multiple components: a word sequence encoder, a word-level attention layer, a sentence encoder, and a sentence-level attention layer.

They consider document-level classification in this paper. Suppose that a document has L sentences s_i and each sentence has T_t words. W_{it} with $t \in [1,T]$ denotes the words in the ith sentence. The model maps the raw document into a vector representation, upon which a classifier for document classification is constructed. In what follows, each component of the network is explained.

Word Encoder: For a sentence with words W_{it}, $t \in [0,T]$, the words initially embed to vectors via an embedding matrix W_e, $x_{ij} = W_{it}w_{ij}$. A bidirectional GRU is employed to obtain annotations of words by capturing information from both directions for words and thus includes the contextual information in the annotation. The bidirectional GRU comprises the forward GRU which reads the sentence s_i from W_{i1} to W_{iT} and a backward GRU which reads from W_{iT} to W_{i1} [43].

$$x_{it} = W_e w_{it}, t \in [1,T] \tag{18}$$

$$\vec{h}_{it} = \overrightarrow{GRU}(x_{it}), t \in [1,T] \tag{19}$$

$$\overleftarrow{h}_{it} = \overleftarrow{GRU}(x_{it}), t \in [T,1] \tag{20}$$

An annotation is obtained for a given word$W_{it}$by concatenating the forward hidden state $\vec{h}_{it}$ and backward hidden state $\overleftarrow{h}_{it}$, i.e.,

h_{it}=$[\vec{h}_{it}, \overleftarrow{h}_{it}]$, which summarizes the information of the whole sentence centered around W_{it}.

Word Attention: Since not every word has an equal impact on the overall meaning of a sentence, an attention mechanism is employed to identify and emphasize the words that are most relevant. This mechanism selectively focuses on informative words and combines their representations to construct a sentence-level vector. Specifically,

$$u_{it} = tanh(W_w h_{it} + b_w) \tag{21}$$

$$\alpha_{it} = \frac{exp(u_{it}^T u_w)}{\sum_t exp(u_{it}^T u_w)} \tag{22}$$

$$s_i = \sum_t \alpha_{it} hit \tag{23}$$

That is to say, word annotation h_{it} is first run through a one-layer MLP to produce u_{it} as a hidden representation of h_{it}, then the importance of the word as the similarity of u_{it} is measured against a word-level context vector u_w and ultimately produced a normalized importance weight $\propto_{it}$ through a softmax function. Then, the sentence vector s_i is determined as a weighted sum of the word annotations by the weights. The context vector u_w can be thought of as a high-level representation of a fixed query "what is the informative word" over the words (as used in a memory network). The word context vector u_w is initialized randomly and learned jointly during the training process.

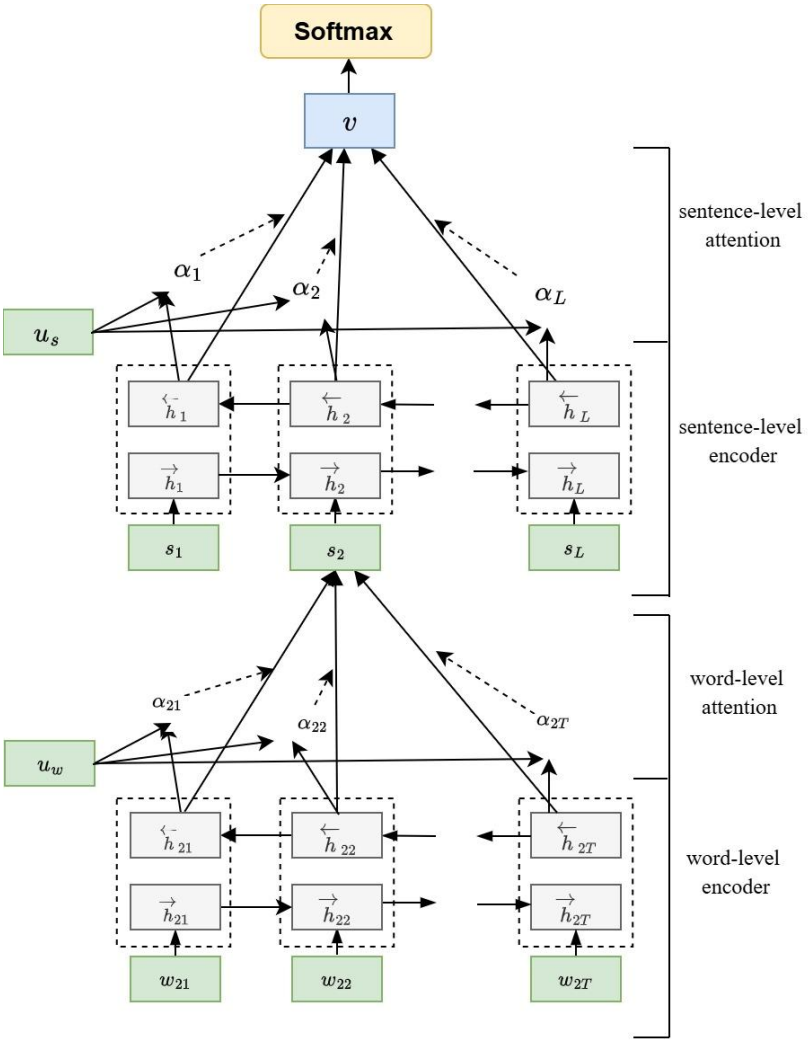

Fig. 7. Hierarchical Attention Network.

Sentence Encoder: Provided the sentence vectors s_i, we can similarly obtain a document vector. A bidirectional GRU is also used to encode the sentences:

$$\vec{h}_i = \overrightarrow{GRU}(s_i), i \in [1, L] \quad (24)$$

$$\overleftarrow{h}_i = \overleftarrow{GRU}(s_i), i \in [L, 1] \quad (25)$$

To get an annotation of sentence i, $\vec{h}_i$ and $\overleftarrow{h}_i$ is concatenated i.e., $h_i = [\vec{h}_i, \overleftarrow{h}_i]$. h_i summarizes the neighbor sentences around sentence i but still focus on sentence i.

Sentence Attention: The attention method is once more employed to reward sentences that provide hints for accurately classifying a document. Additionally, a sentence level context vector, u_s, is introduced to gauge the significance of the phrases. This yields:

$$u_{it} = tanh(W_s h_i + b_s) \quad (26)$$

$$\alpha_{it} = \frac{exp(u_i^T u_s)}{\sum_t exp(u_i^T u_s)} \quad (27)$$

$$s_i = \sum_t \alpha_i hi \quad (28)$$

In this framework, V represents the document vector that encapsulates the comprehensive information of all sentences within a document. Similarly, the context vector at the sentence level can be initialized randomly and optimized jointly during the training phase.

Hierarchical Attention Networks have since been adapted for various applications. For example, Ying et al. [44] proposed a hierarchical attention network tailored for the next-item recommendation task. In their approach, both users and items are first embedded into low-dimensional latent spaces. A two-layer hierarchical attention network is then applied to capture users' evolving long-term preferences and sequential behavioral patterns. Their model accounts not only for the dynamic aspects of users' long- and short-term interests, but also for intricate higher-level interactions between user and item features, as well as between different items. Fig. 8 illustrates their proposed hierarchical attention framework.

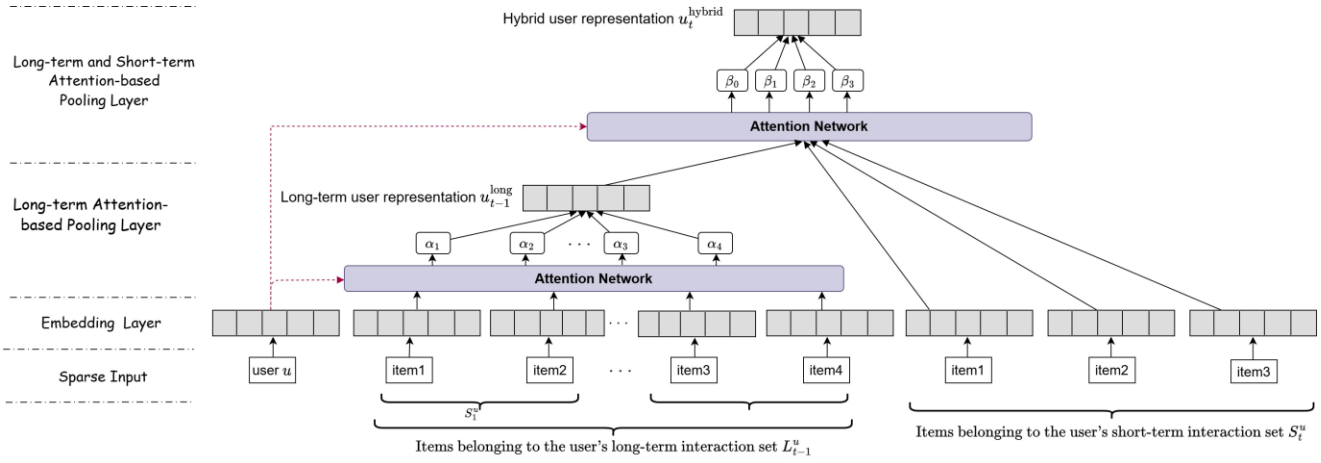

Fig. 8. A sequential hierarchical attention network.

To capture the user's long-term preference, prior to time step t, they learn the long-term user representation as a weighted sum over the embeddings of the items in L_{t-1}^u, with the weights being inferred based on an attention-based pooling layer which is guided by the user embedding. Furthermore, they combine the two representations to get the final hybrid user representation capturing short-term preference along with the long-term preference, which was learned by another attention-based pooling layer over the embeddings of the items in S_t^u.

Some research efforts have also applied Hierarchical Attention Networks to the task of image captioning [45-48]. This form of attention is capable of capturing various levels of abstraction across its layers. Lower-level concepts are integrated into higher-level

representations, and these lower-level features are also propagated upward to assist in word prediction [49].

Wang et al. [47] introduced an innovative Hierarchical Attention Network (HAN) specifically designed for image captioning, where attention is computed across a pyramidal hierarchy of features simultaneously. This pyramidal structure comprises features spanning multiple semantic levels, enabling the generation of different words based on varying feature representations. Given the distinct modalities present in the features, the authors also proposed a Multivariate Residual Module (MRM) to learn unified representations across these modalities. The MRM effectively models feature projections and captures meaningful relationships among them. Additionally, a context gate is incorporated to regulate the relative influence of the different features.

In a separate study, Liu et al. [50] applied a hierarchical attention mechanism within the context of an intrusion detection system (IDS). This mechanism integrates two levels of attention: feature-based attention and slice-based attention. The feature-based attention is employed to strengthen the representational capacity of the traffic features, while the slice-based attention is subsequently applied to multiple segments of traffic data.

3.2. Bidirectional attention

Seo et al. [51] proposed the Bi-Directional Attention Flow (BIDAF) network, a hierarchical multi-stage architecture designed to represent context at multiple levels of granularity. This model employs a bidirectional attention flow mechanism to generate a query-aware context representation without performing early summarization. The bidirectional attention mechanism operates in two directions: query-to-context and context-to-query, offering complementary information from both perspectives. Fig. 9 illustrates the Bi-Directional Attention Flow (BIDAF) network, which models the contextual paragraph through a layered, hierarchical structure.

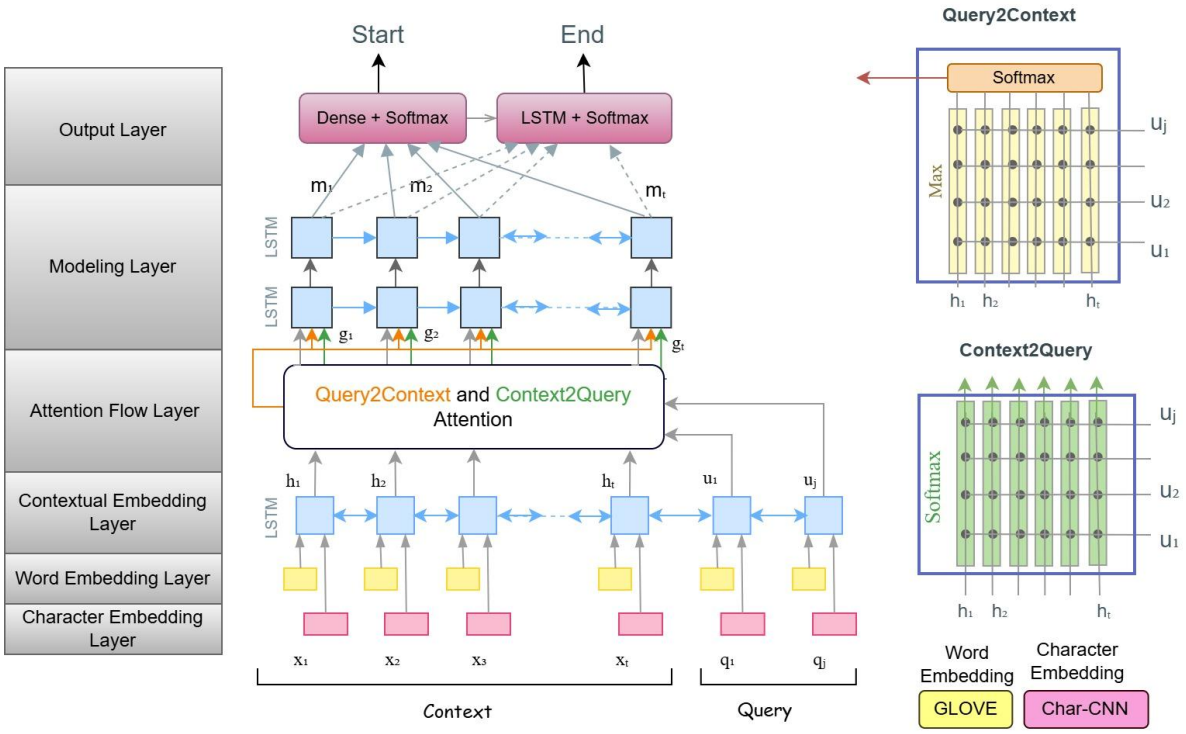

Fig. 9. Bi-Directional Attention Flow Model.

The layer receives two inputs: contextual vector representations of the context $\mathbf{H}$ and the query $\mathbf{U}$. In fact, $\mathbf{H} \in \mathbb{R}^{2d \times T}$ from the context word vectors X, and $\mathbf{U} \in \mathbb{R}^{2d \times J}$ from query word vectors Q. It outputs the query-aware vector representations of the context words, denoted $\mathbf{G}$, alongside the contextual embeddings received from the previous layer.

The attention flow layer serves to connect and disambiguate the information from the context and query words. To do this, attention is computed in both directions: from context to query and from query to context. Both attention methods are computed from the same similarity matrix, $S \in \mathbb{R}^{T \times J}$, where T is the number of context words, J is the number of query words, S_{tj} denotes the similarity between the t-th context word and j-th query word as derived from the similarity matrix.

The similarity matrix is computed by

$$S_{tj} = \alpha(H_{:t}, U_{:j}) \in \mathbb{R} \tag{29}$$

where α is a trainable scalar function encoding the similarity between its two input vectors; $\mathbf{H}_{:t}$ is t-th column vector of $\mathbf{H}$, and $\mathbf{U}_{:j}$ is j-th column vector of $\mathbf{U}$.

Context-to-query (C2Q): This attention mechanism identifies which query words are most relevant to each word in the context. Let $a_t \in \mathbb{R}^J$ represent the attention weights on the query words by $t-$th context word, $\sum a_{tj} = 1$ for all t. The attention weight is calculated by $a_t = softmax(S_{t:}) \in \mathbb{R}^J$, and then subsequently each attended query vector is $\widetilde{\boldsymbol{U}}_{:t} = \sum_j \boldsymbol{a_{tj}} \boldsymbol{U_{:j}}$. Thus $\widetilde{U}$ is a $2d - by - \boldsymbol{T}$ matrix comprising the attended query vectors for the whole context.

Query-to-context (Q2C): This attention mechanism determines which context words exhibit the highest similarity to any of the query words, making them particularly important for accurately answering the query. The attention weights for the contextual words are obtained by $\boldsymbol{b} = \text{softmax}(\max_{\boldsymbol{col}}(\boldsymbol{S})) \in \mathbb{R}^{\boldsymbol{T}}$, where the max operation ($\max_{\boldsymbol{col}}$) is taken over the columns. Thus, attended context vector is given by $\widetilde{\boldsymbol{h}} = \sum_t \boldsymbol{b_t} \boldsymbol{H_{:t}} \in \mathbb{R}^{\boldsymbol{2d}}$. This vector signifies the weighted sum of the most relevant words in the context with respect to the query. $\widetilde{\boldsymbol{h}}$ is tiled T times along the column to get $\widetilde{\mathbf{H}} \in \mathbb{R}^{2d \times T}$. Finally, the contextual embeddings are combined with the attention vectors to generate $\boldsymbol{G}$, where each column vector is the query-aware representation of each corresponding context word. $\boldsymbol{G}$ is defined by

$$\boldsymbol{G_{:t}} = \beta(\boldsymbol{H_{:t}}, \widetilde{\boldsymbol{U}}_{:t}, \widetilde{\boldsymbol{H}}_{:t}) \in \mathbb{R}^{\boldsymbol{d_G}} \tag{30}$$

where $G_{:t}$ is the $t-th$ column vector (corresponding to $t-th$ context word), β is a trainable vector function that combines its (three) input vectors, and d_G is the output dimension of the β function. Although the β function can be any arbitrary trainable neural network, e.g., Multi-Layer Perceptron (MLP), a straightforward concatenation as below still performs well in experiments:

$$\beta(\boldsymbol{h}, \widetilde{\boldsymbol{u}}, \widetilde{\boldsymbol{h}}) = [\boldsymbol{h}; \widetilde{\boldsymbol{u}}; \boldsymbol{h} \circ \widetilde{\boldsymbol{u}}; \boldsymbol{h} \circ \widetilde{\boldsymbol{h}}] \in \mathbb{R}^{8d \times T} (i.e., d_G = 8d) \tag{31}$$

In contrast to the original attention mechanism, which summarizes the query and context into single feature vectors, bi-directional attention allows the attention vector at each time step, as well as the embeddings from earlier layers, to pass through to the following modeling layer. This design helps minimize information loss that typically results from early summarization.

3.3. Multi-Head attention

Vaswani et al. [20] introduced the Transformer architecture for machine translation in 2017, and it has since become a foundational model in the field of deep learning, particularly within natural language processing (NLP).

The Transformer architecture relies entirely on the attention mechanism, enabling it to model dependencies between input tokens without the need for recurrent connections. It incorporates a multi-head attention module, which enhances its ability to capture complex relationships among input elements. This module is widely regarded as a key factor behind the Transformer's remarkable success [52].

The fundamental idea behind the attention mechanism is that each token in the sequence can gather information from all other tokens, enabling the model to better comprehend contextual relationships. This is accomplished by defining an attention function that maps a query, a collection of key-value pairs, and an output, with all components represented as vectors. The resulting output is calculated as a weighted sum of the values, where the weights are derived from a compatibility function that measures the relationship between the query and its associated key [21].

Multi-head attention works by combining n separate scaled dot-product attention mechanisms (self-attention). The scaled dot-product attention is described earlier at the end of section 2. This approach allows simultaneous processing of the three input vectors Q, K, and V, to compute a final result more effectively. The concept of multi-head attention is illustrated in Fig. 10, and its mathematical representation is provided in Eq. 32:

$$\begin{cases} MultiHead\ (Q,K,V) = Concat(head_1, \dots, head_2) W^O \\ where\ head_i = Attention\ (QW_i^Q, KW_i^K, VW_i^V) \end{cases} \tag{32}$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}, W_i^K \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_V}, and\ W^O \in \mathbb{R}^{hd_v \times d_{model}}$.

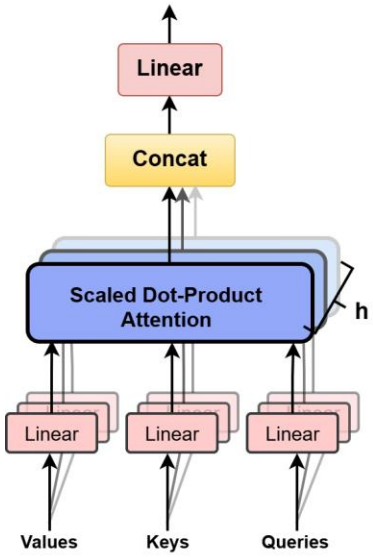

Fig. 10. Structure of the Multi-Head Attention mechanism.

Numerous studies have applied and extended the multi-head attention mechanism across a variety of tasks. Li et al. [53], for instance, proposed a disagreement regularization approach aimed at explicitly promoting diversity among different attention heads. As is well established, multi-head attention enables the model to simultaneously focus on information from different representation subspaces. However, the authors of [53] increase the separation among attention heads by introducing a disagreement regularization term, thereby promoting greater diversity. This approach is applied within the context of machine translation tasks. Given a source sentence x and its translation y, the neural machine translation model is trained to maximize the conditional probability of the translation based on a parallel training corpus. To enhance diversity among attention heads, an auxiliary regularization term is incorporated. Formally, the training objective is revised as follows:

$$J(\theta) = \arg\max_{\theta} \{ \underbrace{L(y|x;\theta)}_{likelihood} + \lambda * \underbrace{D(a|x,y;\theta)}_{disagreement} \} \tag{33}$$

In Equation (33), a denotes the attention matrices, and λ is a hyperparameter, which is empirically set to 1.0 in their study. The auxiliary regularization term $D(\cdot)$ encourages each attention component to capture distinct features from its corresponding projected subspace.

To achieve this, the authors introduce three types of disagreement regularization, each targeting a different aspect: the subspace, the attended positions, and the output representations. These regularizations are designed to make the outputs of each attention head differ from one another.

Disagreement on Subspaces (Sub.): This form of regularization is designed to maximize the cosine distance between projected value vectors. The process begins by computing the cosine similarity $cos(\cdot)$ between vector pairs V^{i} and V^j from different value subspaces, using the dot product of the normalized vectors. This similarity reflects the cosine of the angle between V^i and V^j. Accordingly, the cosine distance is defined as the negative of this similarity, i.e., $-cos(\cdot)$. The goal is to increase the average cosine distance among all attention head pairs. The corresponding regularization term is formally defined as follows:

$$D_{subpace} = -\frac{1}{H^2}\sum_{i=1}^{H}\sum_{j=1}^{H}\frac{V^i \cdot V^j}{||V^i||||V^j||} \tag{34}$$

Disagreement on Attended Positions (Pos.): This strategy aims to increase the diversity of attended positions predicted by different heads. While agreement regularization [54, 55] promotes similar alignment patterns across heads, this work introduces a modified approach called alignment disagreement regularization, which encourages variation instead. Specifically, the similarity between the attended positions of two heads is measured by the sum of the element-wise product between their attention matrices $A^i$and A^j.

$$D_{position} = -\frac{1}{H^2}\sum_{i=1}^{H}\sum_{j=1}^{H}\left|A^i \odot A^j\right| \tag{35}$$

Disagreement on Outputs (Out.): This approach applies regularization directly to the outputs produced by individual attention heads, aiming to maximize the differences between them. In a manner similar to the subspace-based method, the negative cosine similarity is utilized to quantify the divergence among these outputs:

$$D_{output} = -\frac{1}{H^2}\sum_{i=1}^{H}\sum_{j=1}^{H}\frac{O^i \cdot O^j}{||O^i||||O^j||} \tag{36}$$

In another study, Cordonnier et al. [56] introduced a collaborative multi-head attention mechanism that allows attention heads to share learned projections. Their findings revealed that the concatenated heads in conventional multi-head attention (MHA) architectures often acquire redundant query and key representations. To address this redundancy, they propose substituting the standard concatenation-based MHA with a collaborative variant within Transformer models. The primary distinction lies in the handling of key and query matrices: unlike the standard approach where each head has its own set of projections, the collaborative model assigns each head a mixing vector m_i. This design reduces the total number of parameters within an attention layer and can be seamlessly integrated into existing Transformer architectures. Notably, collaborative multi-head attention achieves a fourfold reduction in the size of key and query projections without compromising accuracy or inference speed.

3.4. Multi-Query Attention

Multi-head attention layers, as implemented in the Transformer sequence model, serve as a powerful mechanism for transferring information both within and between sequences. Although these layers are typically quick and efficient to train, thanks to their ability to parallelize across the sequence length, one significant limitation of the Transformer lies in the speed of incremental inference. This performance bottleneck is largely due to the memory bandwidth required to repeatedly access the large key and value tensors that store the attention states.

To overcome this challenge, Shazeer et al. [57] introduced a modified attention mechanism known as multi-query attention, in which all attention heads share a single set of keys and values. This design significantly reduces tensor size and, consequently, the memory bandwidth demands during incremental decoding. In conventional multi-head attention, each of the H heads has its own separate query, key, and value projections, allowing them to process different aspects of the input. In contrast, multi-query attention maintains distinct query projections for each head but utilizes shared keys and values across all heads. Fig. 11 illustrates a comparison between standard multi-head attention and the proposed multi-query attention mechanism.

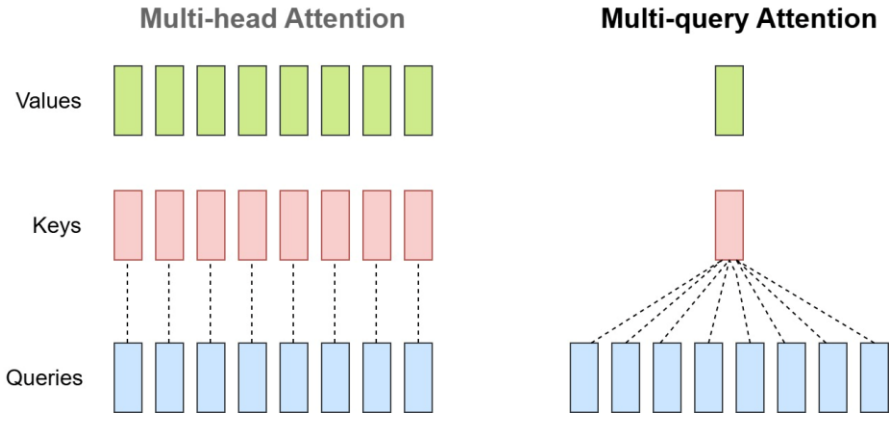

Fig. 11. Overview of multi-query attention methods.

3.5. Group-Query Attention

Ainslie et al. [58] introduced grouped-query attention (GQA), a mechanism that bridges multi-query attention (MQA) and multi-head attention (MHA), achieving performance comparable to MHA while preserving MQA's efficiency. In GQA, attention heads are divided into G groups, with each group sharing a single set of key and value projections. The notation GQA-G denotes the

variant using G groups: for example, GQA-1 corresponds to MQA (since all heads share the same key and value), while GQA-H, where H is the total number of heads, recovers standard MHA.

Fig. 12 compares GQA with MQA and MHA. In GQA, the key and value vectors for each group are computed by mean-pooling the corresponding heads. This design achieves a balance between speed and accuracy.

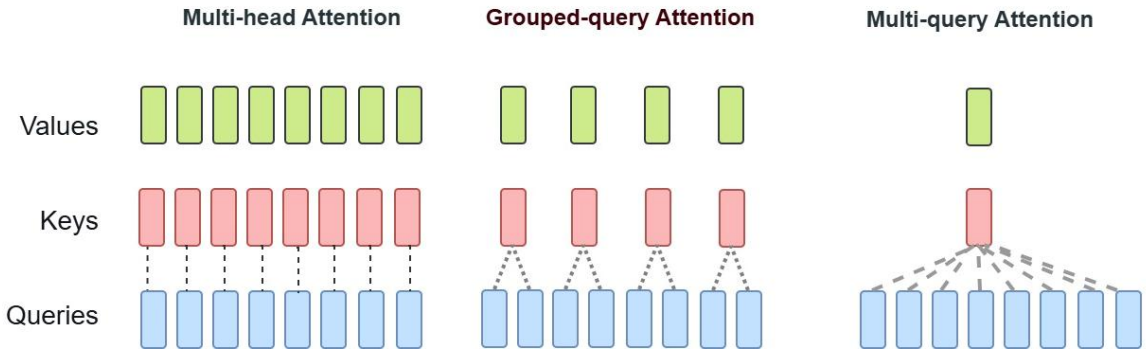

Fig. 12. Overview of grouped-query attention, multi-query attention, and multi-query attention methods.

Transitioning from MHA to MQA reduces the number of key and value heads from H to one, significantly lowering the size of the key-value cache and decreasing memory bandwidth needs by a factor of H. However, in large models, where the number of heads increases with scale, this transition becomes more extreme and leads to underutilization of capacity. GQA mitigates this issue by preserving a proportional balance between bandwidth and model capacity as the model size increases.

Furthermore, standard sharding methods often involve partitioning keys and values by the number of model parallel units, which introduces inefficiencies [59]. GQA avoids such redundancy, making it particularly suitable for large-scale architectures by achieving a practical compromise between speed and performance.

Grouped-query attention (GQA) has been widely integrated into large language models (LLMs) as a means to reduce the computational overhead associated with multi-head attention (MHA). Converting an MHA to a GQA typically involves dividing adjacent query heads in MHA into equal-sized groups, where each group shares the same key and value projections. Nonetheless, several studies have proposed alternative variants of GQA. For instance, Chen et al. [60] introduced AsymGQA, an activation-aware method that asymmetrically converts MHA into GQA to enhance model performance. As illustrated in Fig. 13, AsymGQA utilizes activation-induced similarities across layers to inform the grouping of attention heads, thereby achieving improved model accuracy.

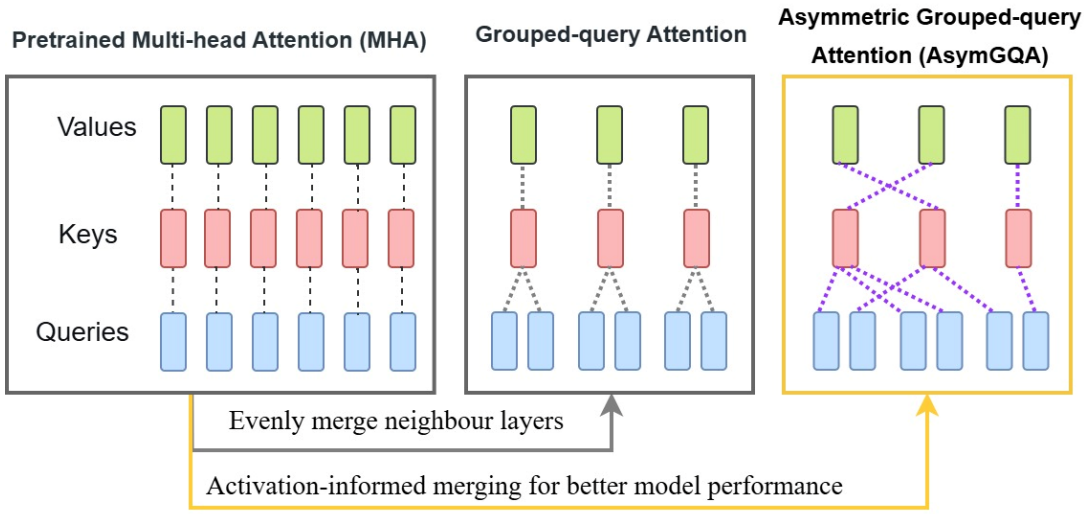

Fig. 13. Comparison of GQA and AsymGQA.

Instead of the neighbor-based grouping strategy used in standard GQA, the authors adopted activation-informed asymmetric grouping to organize attention heads. Initially, they applied activation-informed symmetric grouping, which sequentially groups heads from the first MHA layer to the last. In this approach, the key and value projections are grouped independently at each layer.

Their proposed method utilizes a search-based strategy to identify the optimal grouping of key (and value) layers by evaluating the similarity among them within the MHA framework.

The similarity $sim(A,B)$ between two activation matrices $A \in \mathbb{R}^{n\times m}$ and $B \in \mathbb{R}^{n\times m}$ using row vectors is as follows.

$$sim(A,B) = \frac{1}{2}\left(\sum_{i=0}^{n-1} \underset{j=0}{\overset{n-1}{max}}\ cosim\left(A_{i,*}, B_{i,*}\right) + \sum_{i=0}^{n-1} \underset{j=0}{\overset{n-1}{max}}\ cosim\left(B_{i,*}, A_{i,*}\right)\right) \tag{37}$$

where $cosim(\cdot)$ calculates the cosine similarity between two vectors u and v:

$$cosim(u,v) = \frac{u.v}{\|u\|\|v\|} \tag{38}$$

In contrast to symmetric grouping, asymmetric grouping permits variable group sizes, making it especially advantageous in situations where information relevance is unevenly distributed across the input space. By expanding the search space beyond equal-sized groupings to allow arbitrary configurations, asymmetric grouping increases the potential to discover more optimal grouping strategies than those achieved through symmetric grouping.

Fig. 14 illustrates the differences between activation-informed asymmetric grouping (AsymGQA) and the standard neighbor-based grouping approach. Enabling support for asymmetric grouping requires only minimal modifications to the existing search algorithm. In symmetric grouping, preserving uniform group sizes necessitates that whenever a key or value head is reassigned to a new group, another element from that group must be swapped back. To accommodate unequal group sizes, the method introduces an additional parameter p_{preserve}, which denotes the likelihood of maintaining group sizes during the swap operation.

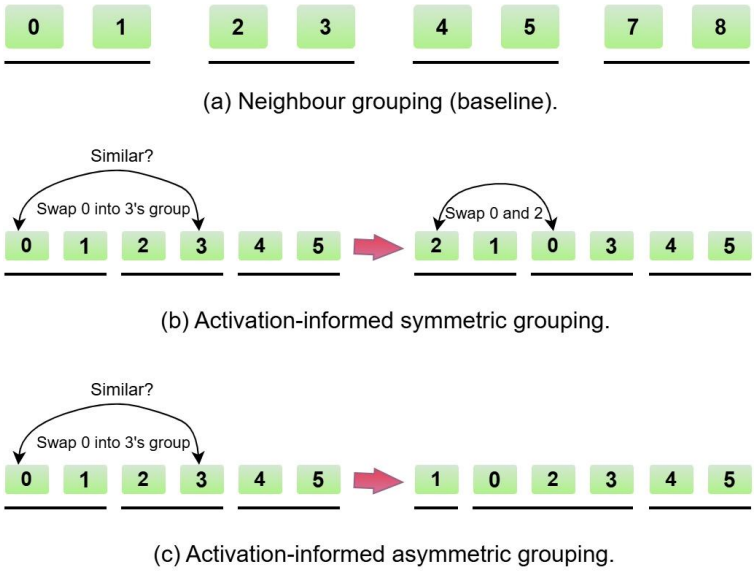

Fig. 14. Naive neighbor grouping vs AsymGQA.

Similarly, Joshi et al. [61] introduced Quality and Capacity-aware Grouped-Query Attention (QCQA), which aims to reduce the accuracy degradation caused by query grouping through a loss function called Weight-Sharing Error, a measure of the distance between the key and value head distributions. Their method adopts a two-stage search strategy. In the first stage, query groups Q are formed independently for each layer. The second stage evaluates these groupings using Weight-Sharing Error (WSE) and the KV cache as fitness criteria, allowing the method to identify and exclude layers where grouping would most negatively impact accuracy. Layers that cause significant accuracy loss are retained in their original multi-head attention (MHA) form, while others are grouped, thus determining which layers maintain groupings and which remain as MHA. Fig. 15 Illustrates the grouping approach employed in grouped-query attention (GQA).

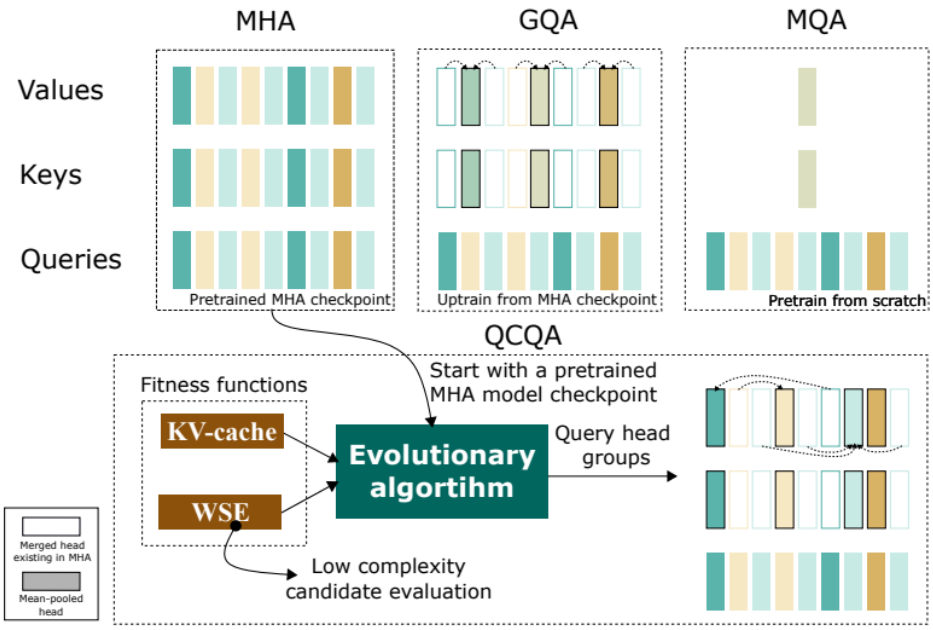

Fig. 15. Grouping approach in QCQA [61].

Additionally, Chinnakonduru et al. [62] proposed a variant called Weighted Grouped-Query Attention (WGQA). Instead of using the heuristic element-wise mean to aggregate key and value heads, they introduced a parametric aggregation method. They explored various aggregation strategies to assess whether incorporating a small number of additional parameters during training could lead to improved performance. Their analysis confirmed that scaling laws remain valid, showing that the performance gap between standard GQA and WGQA grows as model size increases.

Khan et al. [63] proposed advancements to grouped-query attention (GQA) by introducing two innovative strategies that move beyond the static nature of traditional grouping: Key-Distributed GQA (KDGQA) and Dynamic Key-Distributed GQA (DGQA). Both approaches utilize the norms of key heads to guide the assignment of queries to groups. Specifically, KDGQA determines query grouping during each forward pass based on the ratio of key head norms, whereas DGQA tracks how these norm ratios evolve throughout training to inform dynamic allocation. The authors also introduce Perturbed GQA (PGQA) as a case study, which injects variability into otherwise static groupings by subtracting noise from attention maps during the grouping process. Fig. 16 illustrates these mechanisms. In standard multi-head attention (MHA), each query corresponds to its own key-value head. In contrast, GQA assigns a single key-value head to fixed subgroups of queries, maintaining constant group sizes through uniform and static grouping. KDGQA and DGQA depart from this approach by allowing non-uniform groupings determined by the magnitudes (norms) of the key heads.

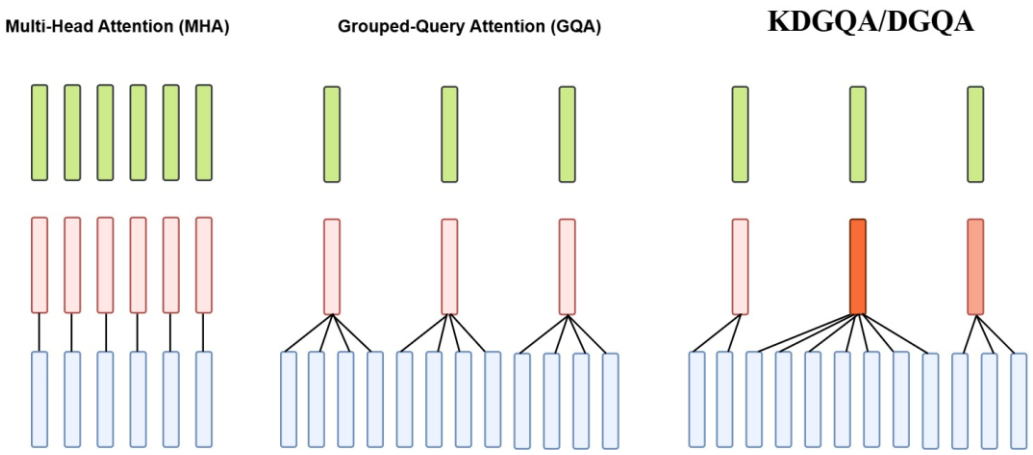

Fig. 16. Overview of Key-Distributed GQA (KDGQA) / Dynamic Key-Distributed GQA (DGQA) approaches.

3.6. Graph Attention

Many real-world problems involve data that do not conform to regular, grid-like structures but instead exist in irregular domains. Examples include 3D meshes, social and telecommunication networks, biological systems, and brain connectomes. These types of data are typically modeled as graphs [64]. In response, various efforts have been made to adapt neural networks to handle such arbitrarily structured data. Graph Neural Networks (GNNs) [65, 66] emerged as a generalized form of recursive neural networks, capable of processing a wider range of graph types, including cyclic, directed, and undirected graphs. GNNs operate through an iterative mechanism that updates node representations until a stable state is reached. Once equilibrium is achieved, a neural network generates node-specific outputs based on their final states. This foundational approach was refined in [67], where the propagation phase was enhanced through the integration of gated recurrent units (GRUs) [68, 69].

Attention mechanisms are particularly powerful in handling inputs of variable sizes by enabling the model to selectively focus on the most informative parts of the input during decision-making. Building on these strengths, Veličković et al. [70] proposed an attention-based framework for node classification in graph-structured data. Their approach involves computing the hidden representations of each node by attending to its neighbors using a self-attention mechanism. This model, known as the Graph Attention Network (GAT), introduces a neural architecture specifically designed for graphs, utilizing masked self-attention layers to overcome limitations seen in earlier graph convolutional methods. By stacking multiple attention layers where each node adaptively weighs features from its neighbors, the model implicitly assigns varying levels of importance to different neighbors without relying on expensive matrix operations like inversion or requiring prior knowledge of the entire graph structure. This attention-based setup brings multiple advantages: (1) it is computationally efficient, as it can be parallelized over node-neighbor pairs; (2) it naturally handles nodes with varying degrees by learning flexible weighting schemes; and (3) it is well-suited for inductive learning scenarios, where the model needs to generalize to entirely unseen graphs.

In Graph Attention Networks (GATs), the graph attention layer executes a sequence of operations tailored for processing graph-structured data [71]. As a first step, each node is subjected to a shared linear transformation, governed by a learnable weight matrix W, which prepares the node features for subsequent attention-based computations and representation learning.

After the initial linear transformation in Graph Attention Networks (GATs), the next step involves computing attention coefficients, which quantify the raw (non-normalized) attention scores between neighboring nodes. Specifically, the transformed embeddings z of each pair of adjacent nodes are concatenated to form a joint feature vector. This concatenated representation is then processed via a dot product with a trainable weight vector, allowing the model to integrate node-specific information into the attention computation.

To incorporate nonlinearity into the model, the result of the dot product operation is passed through the LeakyReLU activation function. This activation helps the network capture complex patterns by introducing a nonlinear transformation. Following this, the attention coefficients are normalized to ensure consistency and comparability across different nodes. This normalization is accomplished using the Softmax function, which scales the coefficients into a probability distribution, enabling effective weighting of neighboring node features.

During the aggregation phase, the model integrates the embeddings of neighboring nodes, guided by the attention weights computed earlier. A key challenge in Graph Attention Networks (GATs) is the potential instability introduced by the self-attention mechanism. To mitigate this, GATs adopt the strategy of multi-head attention. This technique involves employing several parallel attention mechanisms, referred to as “heads”, each equipped with its own set of parameters. These multiple heads allow the model to capture a richer set of relationships while improving training stability. Each head independently generates an output, which is then combined to form the overall output of the layer. Typically, the intermediate layers concatenate the outputs of all attention heads, whereas the final layer averages them to synthesize insights from different attention perspectives. In GATs, attentional weights are implicitly learned by comparing node features through self-attention, as illustrated in Fig. 17. The mathematical formulation describing how node embeddings are computed is accordingly transformed in this context:

$$h_i = \sum_{j \in N_i} a_{ij} W x_j \tag{39}$$

where a_{ij} are the network's dynamically determined attention weights.

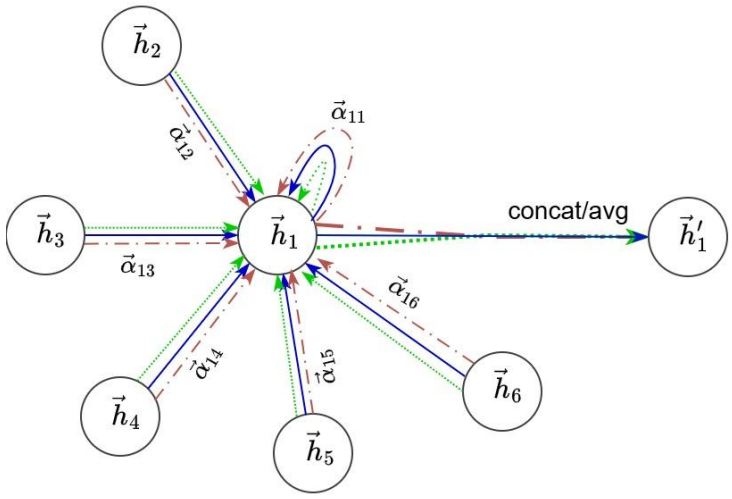

Fig. 17. An illustration of the multi-head attention mechanism, where node 1 attends to its neighboring nodes using K=3 distinct attention heads.

Drawing inspiration from attention mechanisms, the Graph Attention Network (GAT) was introduced to learn the relative importance between a node and its neighbors, enabling effective fusion of neighbor information for node classification. Despite its strengths, traditional graph neural networks are limited in that they can only be applied to homogeneous graphs, those with a single type of node and edge. In contrast, real-world graphs are often heterogeneous, containing multiple types of nodes and edges, which results in richer semantics and more comprehensive information. This inherent complexity renders standard homogeneous graph approaches unsuitable for direct application to heterogeneous settings [72]. To address this limitation, Wang et al. [73] proposed the Heterogeneous Graph Attention Network (HAN), a model designed to handle the challenges of heterogeneous graphs by incorporating both node-level and semantic-level attentions. The architecture adopts a hierarchical attention mechanism, progressing from node-level attention to semantic-level attention. As illustrated in Fig. 18, the process begins with node-level attention, which learns to assign weights to meta-path–based neighbors and aggregates them to generate semantic-specific node embeddings. Subsequently, semantic-level attention differentiates among various meta-paths, learning an optimal weighted combination of the semantic-specific embeddings tailored to the task at hand. Through this two-tiered attention mechanism, HAN effectively identifies the most informative neighbors and meta-paths, enabling the model to construct node representations that capture both structural complexity and semantic richness.

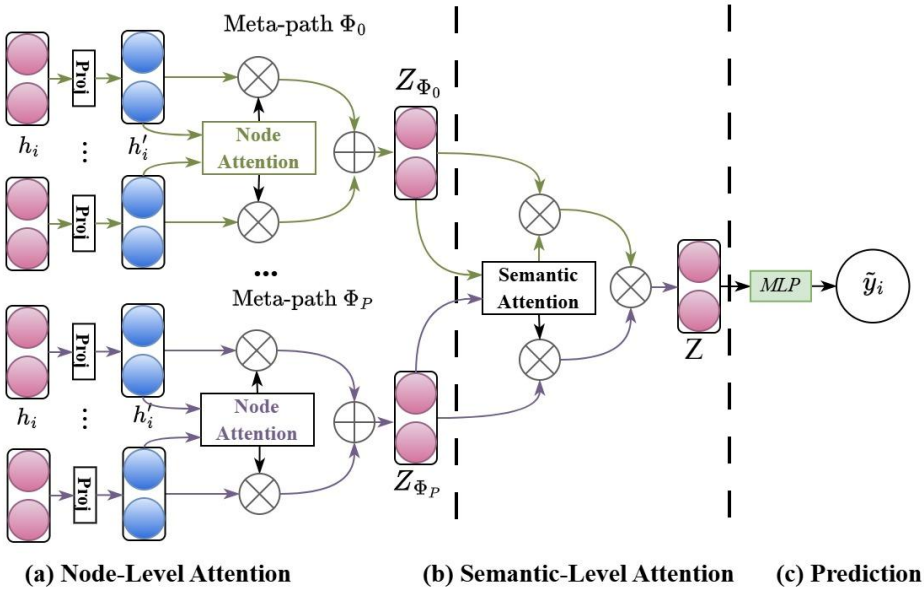

Fig. 18. Architecture of the Heterogeneous Graph Attention Network (HAN).

Building upon the foundational GAT framework, Wang et al. [74] propose an extension that incorporates additional relational heads. These relational heads act as relation-specific gates, regulating the flow of information from neighboring nodes based on the nature of their relationships. The overall architecture of this enhanced model is depicted in Fig. 19, which integrates two types of multi-head attention mechanisms: standard attention heads and the newly introduced relational heads.

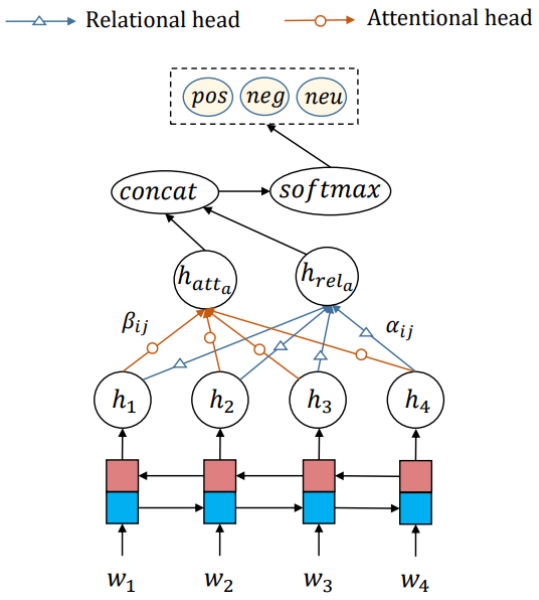

Fig. 19. The Structure of the relational graph attention network (R-GAT) [74].

Brody et al. [75] highlight a key limitation in the widely adopted Graph Attention Network (GAT), noting that its attention mechanism is inherently static. In its standard formulation and implementation, GAT assigns neighbor scores in a monotonic fashion based on individual node features, regardless of the specific query node. Consequently, GAT fails to capture even basic alignment tasks, as it lacks the capacity for dynamic attention computation. To address this limitation, the authors propose a modification to the internal operation sequence of GAT, resulting in GATv2, a straightforward yet more expressive variant of the original model. The improvement lies in the revised calculation of attention coefficients. In GATv2, the embeddings of a node pair are first concatenated and passed through a LeakyReLU activation function. The activated output is then projected via a dot product with a learnable weight vector. This revised formulation enables GATv2 to dynamically adjust attention weights based on the interaction between node pairs, thereby enhancing its ability to model complex structures and capture nuanced relationships within the graph.

3.7. Channel Attention

Channel attention mechanisms aim to highlight significant feature channels while reducing the impact of less relevant ones. This targeted focus allows deep learning models to prioritize informative features, enhancing performance in tasks such as image classification and segmentation. By enabling neural networks to selectively attend to individual feature map channels, channel attention facilitates more efficient information flow and improves the quality of reconstructed images. Dynamically adjusting each channel's weight based on its significance, these mechanisms lead to more accurate input representations and better outcomes in high-resolution image generation [76].

Channel attention mechanisms operate on the premise that different feature channels contribute unequally to a model's performance on a specific task. By computing an attention score for each channel, the network can dynamically increase or decrease the influence of individual features during the forward pass. This process of recalibrating feature importance generally involves global feature aggregation methods, such as average pooling or max pooling, which are applied across the spatial dimensions of the feature maps. The resulting channel-wise descriptors are then passed through a gating mechanism, commonly implemented as a feed-forward neural network, to produce the corresponding attention weights [77]. In this approach, attention scores are computed separately for each channel, as each feature map is responsible for capturing distinct regions or aspects of the input data [8].

A variety of techniques have been developed to implement channel attention mechanisms, each employing a distinct strategy. Notable examples include Squeeze and Excitation Network [78], Channel Attention in Convolutional Block Attention Module (CBAM) [79], Residual Channel Attention Network (RCAN) [80], Double Attention Networks [81], Second Order Channel Attention (SOCA) [82], Global Second-Order Pooling (GSoP) [83], Dual attention network [84], Gated Channel Transformation

(GCT) [85], Style-based Recalibration Module (SRM) [86], Efficient Channel Attention (ECA) [87], Holistic Channel Attention Network (HAN) [88], Frequency Channel Attention (Fcanet) [89], Split-Attention Networks [90]. In the following sections, we examine several of the more widely adopted models in greater detail.

3.7.1. Squeeze and Excitation Network (SENet)

The Squeeze-and-Excitation Network (SENet) [78] represents a foundational model in the development of modern attention mechanisms. A SENet is constructed by integrating multiple Squeeze-and-Excitation (SE) blocks, which are designed to apply dynamic, channel-wise attention. Each SE block operates through two primary stages: Squeeze and Excitation, as depicted in Fig. 20.

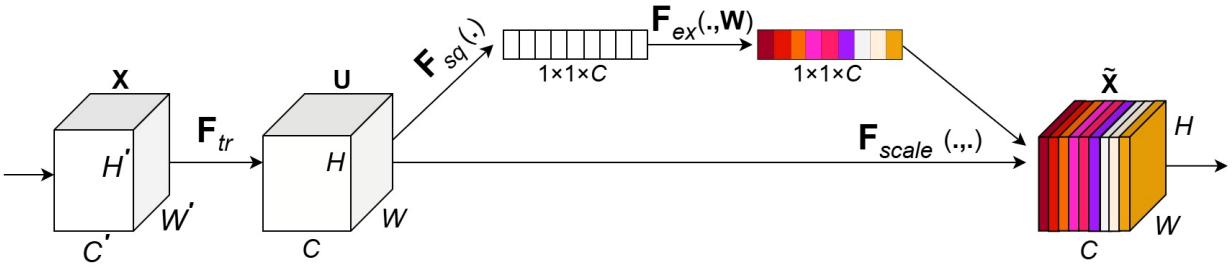

Fig. 20. Squeeze-and-Excitation block.

In the Squeeze phase, the output from a convolutional block is compressed by applying global average pooling to each feature map, reducing each channel to a single scalar value. This results in a compact representation called a channel descriptor, which captures the global contextual information for each channel and emphasizes significant features. In the subsequent Excitation step, the channel descriptors are passed through two fully connected (dense) layers to generate a set of attention weights. These weights allow the network to selectively enhance or suppress individual channels based on their importance. By recalibrating the influence of each channel in this way, the model can effectively emphasize the most informative features and suppress less relevant ones. The overall process can be summarized as follows:

$$f_s = \sigma\left(FC\left(Relu\left(FC(f_g)\right)\right)\right) \tag{40}$$

where FC denotes the fully connected layer, f_g represents global average pooling, and σ refers to the sigmoid activation function. The core idea is to identify the most informative representation of each channel to generate effective attention scores.

3.7.2. Channel Attention in CBAM

The Convolutional Block Attention Module (CBAM) [79] is a lightweight, yet powerful attention mechanism designed for feed-forward convolutional neural networks. It enhances feature representations through sequential attention refinement using two complementary modules: channel attention and spatial attention, delivering notable performance gains with minimal computational overhead. In CBAM, the channel attention sub-module generates an attention map by leveraging inter-channel dependencies within the feature representations. Since each channel in a feature map can be interpreted as a specific feature detector [91], the channel attention mechanism identifies what aspects of the input image are most informative. To compute channel attention, the spatial dimensions of the input feature map are compressed using both average pooling and max pooling, producing two separate descriptors. These descriptors are then fed into a shared three-layer multi-layer perceptron (MLP) to generate attention weights. The outputs of the two MLP branches are combined through element-wise addition and passed through a sigmoid activation function to produce the final channel attention map. Fig. 21 illustrates the structure of the channel attention sub-module used in CBAM. In summary, the channel attention is calculated as follows:

$$f_{ch} = \sigma\left(MLP\left(MaxPool\left(f\right)\right) + MLP\left(AvgPool\left(f\right)\right)\right) \tag{41}$$

where σ denotes the sigmoid activation function, and f refers to the input features. The multi-layer perceptron (MLP) employs the ReLU activation function following each convolutional layer.

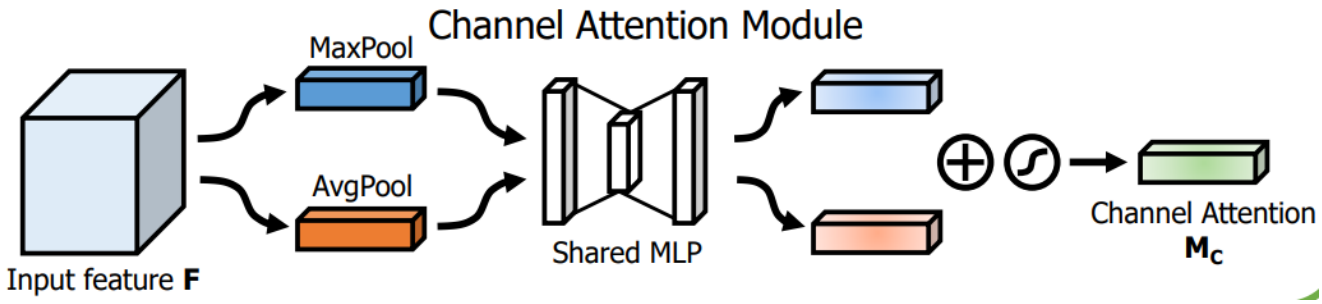

Fig. 21. Channel attention module of CBAM model [79].

3.7.3. Second-Order Channel Attention (SAN)

While SENet relies solely on first-order feature statistics through global average pooling, it overlooks higher-order information, which can limit the network's discriminative power. In contrast, prior studies [92, 93] have demonstrated that incorporating second-order statistics in deep convolutional neural networks (CNNs) can yield more informative and discriminative feature representations compared to first-order methods. Building on these insights, Dai et al. [82] explored the use of second-order feature statistics and introduced the Second-order Attention Network (SAN) for the task of Single Image Super-Resolution (SISR). A key innovation in this network is the integration of non-local enhanced residual groups (NLRG) with the Second-Order Channel Attention (SOCA) mechanism. The architecture is illustrated in Fig. 22. The NLRG module combines non-local operations to extract global contextual cues and integrates shared residual groups to capture deep hierarchical features. In contrast to SENet's first-order approach, SOCA leverages global covariance pooling to compute second-order statistics, capturing inter-feature correlations more effectively. This enables the network to selectively attend to more fine-grained and relevant features, thereby enhancing its ability to learn discriminative representations. As a result, SAN, empowered by the SOCA mechanism achieves superior performance, especially when dealing with images containing complex textures and higher-order structural information.

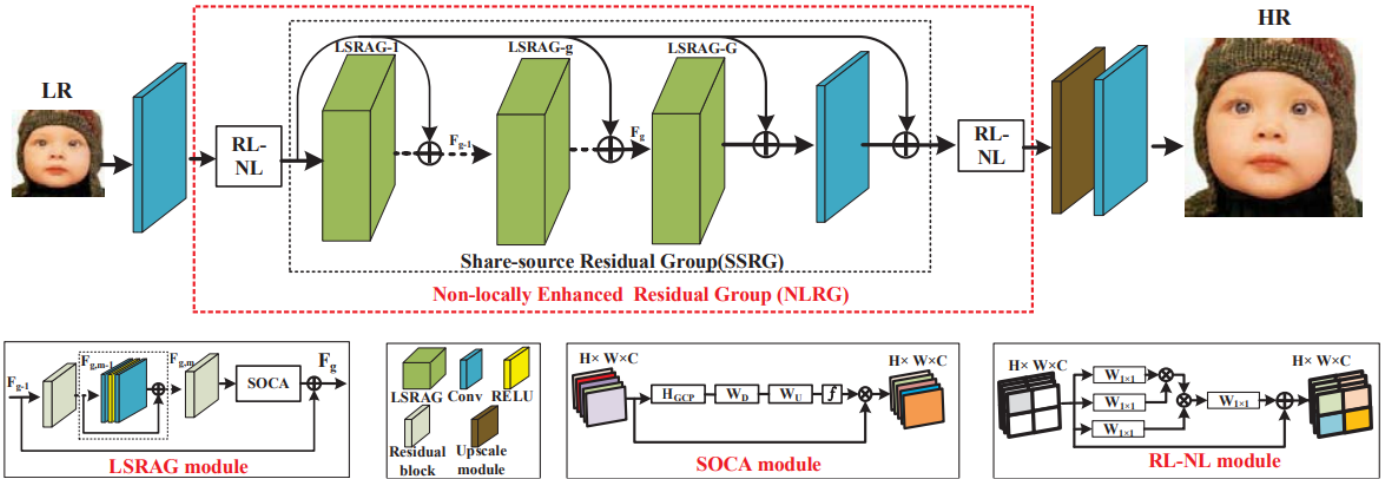

Fig. 22. Framework of the second-order attention network (SAN) and its sub-modules [82].

3.7.4. Efficient Channel Attention (ECA)

To limit model complexity, SENet reduces the number of channels during the attention process. However, this dimensionality reduction limits the model's ability to explicitly capture the relationship between input features and their corresponding weight vectors, which can degrade performance quality [36]. To address this limitation, the Efficient Channel Attention (ECA) block was introduced by [87]. Instead of reducing dimensionality, ECA generates channel attention using a lightweight 1D convolution operation, where the kernel size is adaptively determined through a non-linear function of the channel dimension. The architecture of an ECA block is illustrated in Fig. 23.

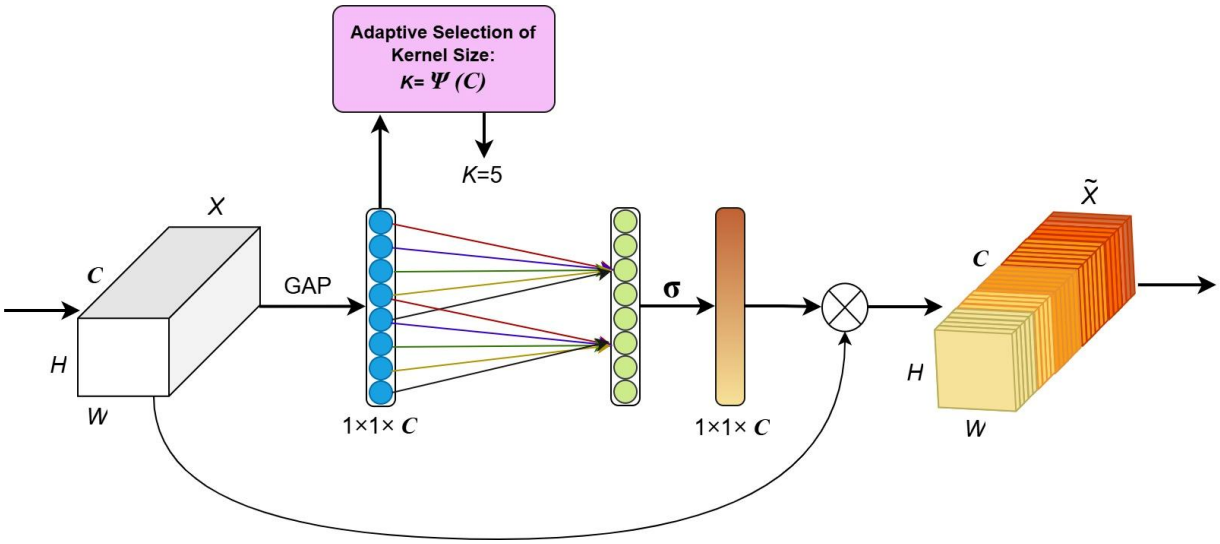

Fig. 23. Diagram of efficient channel attention (ECA) model.

After applying global average pooling across channels, ECA captures local cross-channel dependencies by evaluating each channel alongside its k neighboring channels without performing dimensionality reduction. The formal definition of the ECA block is as follows:

$$\begin{cases} s = F_{\text{eca}}(X, \theta) = \sigma\left(\text{Conv1D}\left(\text{GAP}(X)\right)\right) \\ Y = sX \end{cases} \tag{42}$$

where Conv1D(·) denotes 1D convolution with a kernel of shape k across the channel domain, to model local cross-channel interaction. The parameter k decides the coverage of interaction. In ECA, the kernel size k is adaptively determined from the channel dimensionality C, instead of by manual tuning, using cross-validation:

$$k = \psi(C) = \left|\frac{\log_2(C)}{\gamma} + \frac{b}{\gamma}\right|_{odd} \tag{43}$$

where γ and b are hyperparameters, and $|x|_{odd}$ denotes the function that rounds to the nearest odd integer.

3.7.5. Gated Channel Transformation (GCT)

Gated Channel Transformation (GCT) [85] is an innovative module designed to enhance the discriminative power of deep convolutional neural networks (CNNs) by modeling inter-channel relationships. GCT introduces a normalization mechanism that facilitates either competition or cooperation among channels; notably, this normalization step is parameter-free. To make the transformation learnable, GCT incorporates two key components: a global context embedding operator, which captures global contextual information and determines the importance of each channel prior to normalization, and a gating adaptation operator, which modulates the input features on a per-channel basis according to the normalization output. Despite being lightweight, the trainable parameters associated with each channel are highly effective, allowing GCT to be widely adopted without introducing significant parameter overhead. Furthermore, the gating adaptation parameters are straightforward to interpret and visualize, providing insights into GCT's internal behavior. The overall architecture of GCT is depicted in Fig. 24. In this structure, the embedding weight α governs each channel's influence before normalization, while the gating weight γ and bias β are responsible for adjusting the scale of the input feature x channel-wise.

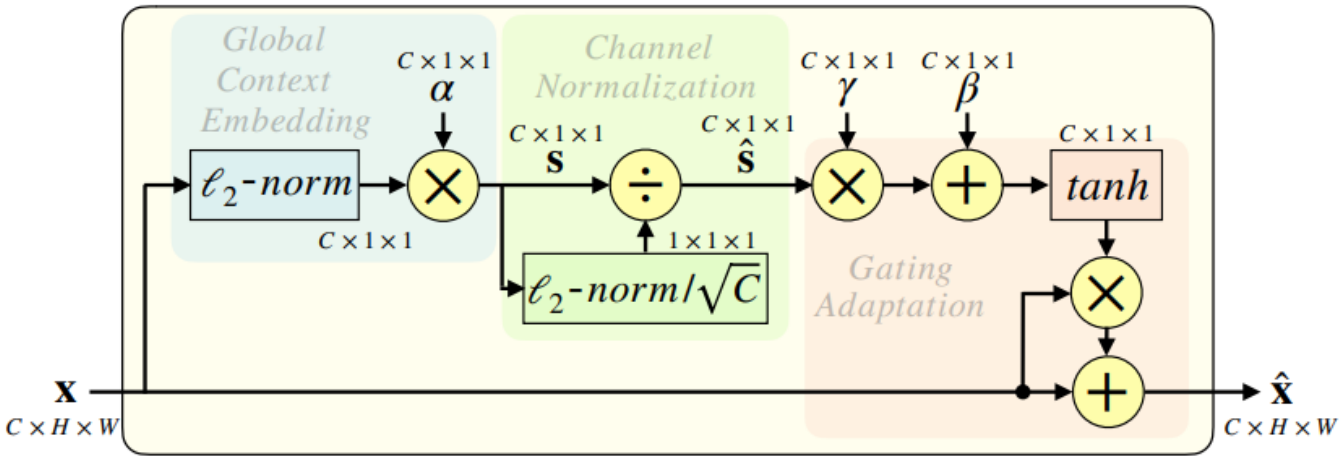

Fig. 24. The structure of Gated Channel Transformation [85].

3.7.6. Frequency Channel Attention (FcaNet)

Relying solely on global average pooling in channel attention mechanisms may constrain a model's representational capacity. To address this, Qin et al. [89] proposed replacing the traditional global average pooling operation with frequency components. Through frequency-domain analysis, they mathematically demonstrated that global average pooling can be interpreted as a special case of the Discrete Cosine Transform (DCT). This insight allowed them to generalize channel compression within the attention mechanism using a frequency-based approach. Building on this foundation, they introduced a method called multi-spectral channel attention, implemented in their proposed model FcaNet. An overview of the multi-spectral channel attention structure is presented on Fig. 25.

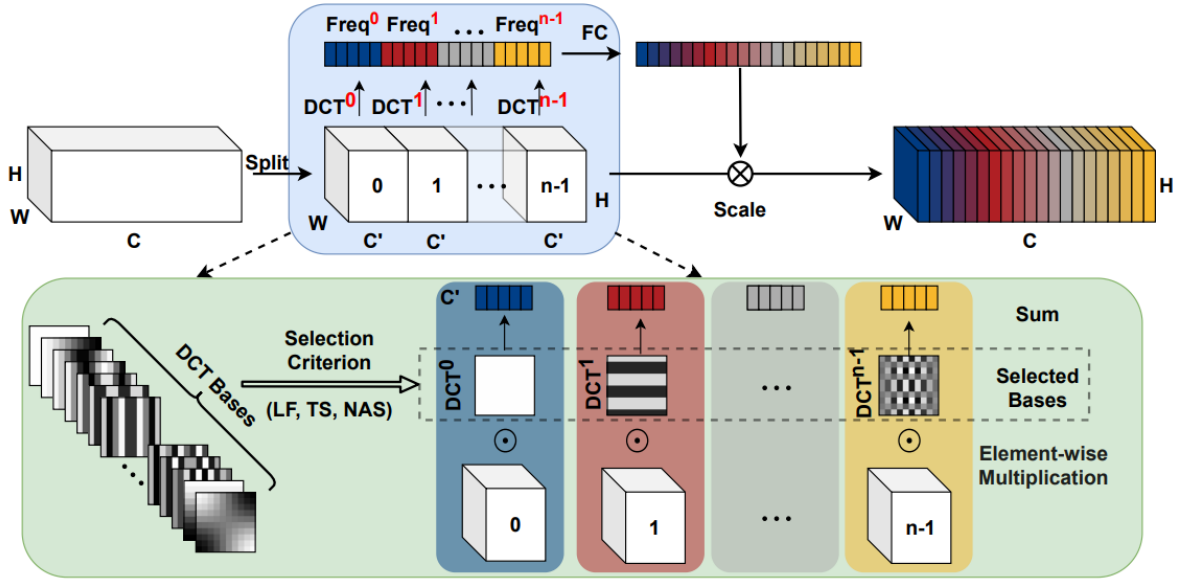

Fig. 25. The structure of Frequency Channel Attention (FcaNet) [89].

Given an input feature map $X \in \mathbb{R}^{C\times H\times W}$, the multi-spectral channel attention mechanism first divides X into multiple segments $x^i \in \mathbb{R}^{\acute{C}\times H\times W}$. A 2D Discrete Cosine Transform (DCT) is then applied to each segment x^i. Notably, pre-computed DCT results can be reused to reduce computational costs. After transforming each segment, the outputs are concatenated into a single vector. This vector is then passed through fully connected layers with ReLU activation, followed by a sigmoid function to generate the final attention vector, like the Squeeze-and-Excitation (SE) block. This process can be expressed as:

$$\begin{cases} s = F_{\text{eca}}(X, \theta) = \sigma\big(W_2 \delta(W_1 D)\big) \\ Y = sX \end{cases} \tag{44}$$

$$D = \text{DCT}\big(\text{Group}(X)\big) \tag{45}$$

Where Group(·) denotes the operation of splitting the input into multiple groups, and DCT(·) refers to the 2D Discrete Cosine Transform.

3.8. Spatial Attention

Spatial attention functions as a mechanism for adaptively selecting important spatial regions, essentially determining where the network should focus. In contrast to channel attention, which emphasizes generating attention weights along the channel dimension, spatial attention derives its attention scores from spatial patches within the feature maps, rather than across channels. Despite this distinction, the sequence of operations involved in computing attention remains largely similar [8]. This mechanism enables neural networks to identify and concentrate on spatial locations that are most relevant to the task at hand. The core idea behind spatial attention is to highlight critical regions within an image, which proves especially beneficial in tasks such as object detection, semantic segmentation, and person re-identification. Spatial attention specifically targets the most informative portions of the spatial feature map—defined by width and height dimensions. By leveraging this mechanism, spatial information from the input image is transformed into a new representation space where the most salient information is preserved [9].

Various strategies have been proposed for implementing spatial attention mechanisms, each employing a distinct approach. In the sections that follow, we take a closer look at several of the most commonly used models.

3.8.1. Spatial Attention in CBAM

As previously noted, the Convolutional Block Attention Module (CBAM) [79] introduces an attention-based feature refinement framework composed of two sequential sub-modules: channel attention and spatial attention. While the channel attention focuses on what features are important, the spatial attention targets where the most informative regions are located, offering complementary information. To compute the spatial attention map, CBAM leverages inter-spatial relationships by first applying average-pooling and max-pooling operations along the channel dimension. These pooled features are then concatenated to form a compact yet effective feature descriptor. Prior studies have demonstrated that performing pooling along the channel axis helps to highlight spatially informative areas [94]. Next, a convolutional layer is applied to the concatenated descriptor to produce a 2D spatial attention map. This map is designed to selectively emphasize or suppress specific regions of the input feature map. In essence, the pooled features capture essential channel information, which is then aggregated and transformed via convolution to guide the network's focus across spatial dimensions. The complete process is illustrated in Fig. 26 and is mathematically formulated as follows:

$$f_{sp} = \sigma\big(Conv_{7\times7}([MaxPool(f); AvgPool(f)])\big) \tag{46}$$

where σ stands for the sigmoid function and $Conv_{7\times7}$ indicates a convolution operation with the 7×7 kernel size.

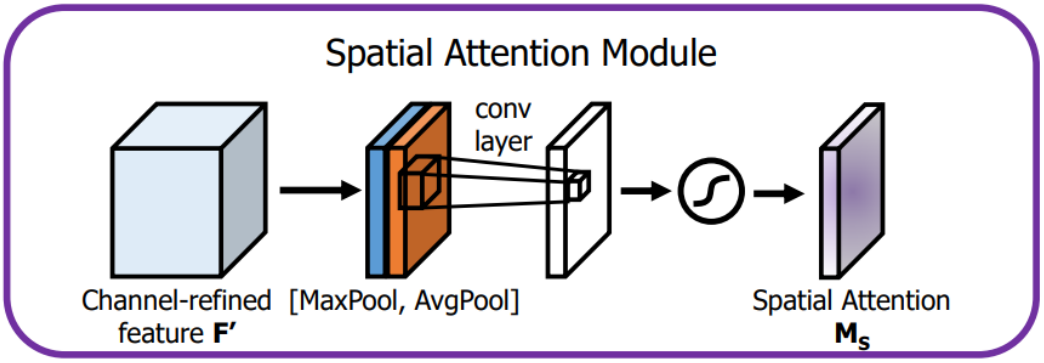

Fig. 26. Spatial attention module of CBAM model [79].

3.8.2. Criss-Cross Attention Network (CCNet)

The Criss-Cross attention module [95] captures contextual information in both horizontal and vertical directions to enhance pixel-wise representational capacity. It is designed to model full-image dependencies from local feature representations while keeping computational and memory costs lightweight.

The Criss-Cross attention module is illustrated on Fig. 27. Starting with a local feature map $H \in \mathbb{R}^{C\times W\times H}$, the module applies two convolutional layers with 1×1 filters to generate feature maps Q and K. Using these maps, the Affinity operation produces attention map A. For each position u in the spatial dimension of Q, a vector Q_u is obtained. The set Ω_u can be obtained by extracting feature vectors from K that lie in the same row or column as position u. $\Omega_{i,u}$ denotes the i-th element of Ω_u. The Affinity operation is then defined as follows:

$$d_{i,u} = Q_u \, {\Omega_{i,u}}^T \tag{47}$$

where $d_{i,u} \in$ D denotes the correlation degree between feature Q_u and $\Omega_{i,u}$. To compute the attention map A, a softmax layer is applied to D along the channel dimension. Another convolutional layer with 1×1 filters is then applied to H to generate V for feature adaptation. For each position u in the spatial dimension of V, a vector V_u and the set Φ_u are obtained, where Φ_u contains the feature vectors in V hat lie in the same row or column as u. The Aggregation operation then collects the contextual information:

$$H'_u = \sum_{i \in |\Phi_u|} A_{i,u}\Phi_{i,u} + H_u \tag{48}$$

where ${H'}_u$ u denotes a feature vector at position u in the output feature map H'. $A_{\mathrm{i,u}}$ represents a scalar value at channel i and position u in A. To enhance the local features and strengthen the pixel-wise representation, contextual information is incorporated into the local feature H. Consequently, the model attains a broad contextual view and selectively aggregates information guided by the spatial attention map. These enriched feature representations provide complementary benefits and exhibit greater robustness for semantic segmentation.

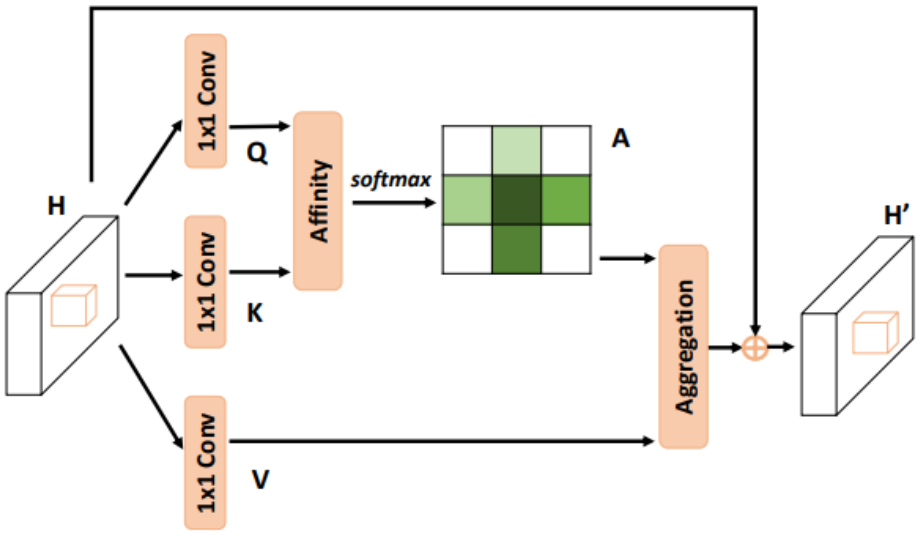

Fig. 27. Criss-Cross Attention module [95].

Building on the Criss-Cross Attention module, Huang et al. [95] introduced the Criss-Cross Network (CCNet) to more effectively and efficiently capture contextual information. An overview of CCNet for semantic segmentation is shown in Fig. 28.

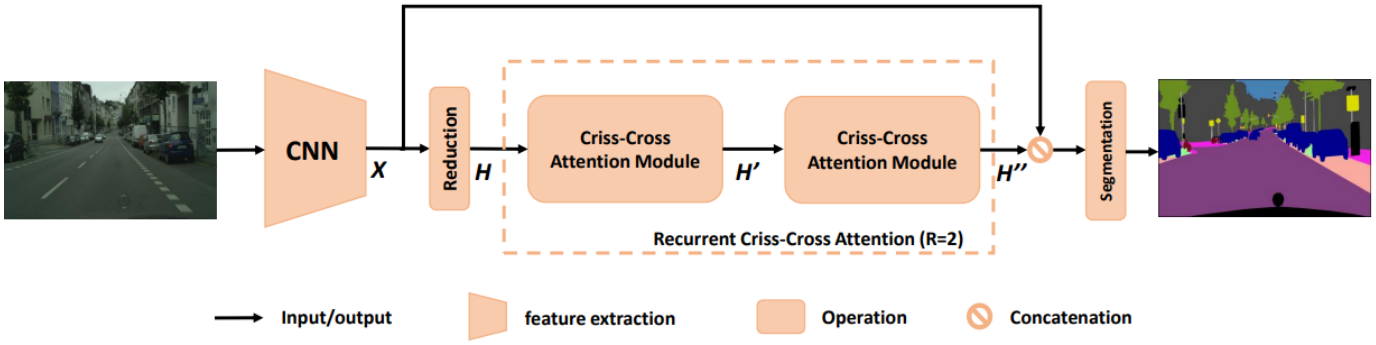

Fig. 28. The overview of the CCNet for semantic segmentation [95].

Although Criss-Cross attention can capture contextual information along both horizontal and vertical directions, the connections between a pixel and its surrounding pixels outside the Criss-Cross path remain absent. To overcome this limitation, CCNet introduces a Recurrent Criss-Cross Attention (RCCA) operation built upon the Criss-Cross attention mechanism. The RCCA module can be unfolded into R loops. In the first loop, Criss-Cross attention takes the CNN-extracted feature maps H as input and produces output feature maps H', with both maps sharing the same dimensions. In the second loop, the attention mechanism then takes H' as input and generates H''. As illustrated in Fig. 28, the RCCA module with two loops $(R = 2)$ is capable of harvesting full-image contextual information from all pixels, thereby generating new feature maps enriched with dense and comprehensive context.

3.8.3. Spatial Pyramid Attention Network (SPAN)

Hu et al. [96] introduced the Spatial Pyramid Attention Network (SPAN), a framework designed for the detection and localization of various forms of image manipulation. SPAN captures inter-patch relationships across multiple spatial scales by employing a pyramid structure composed of local self-attention blocks. This enables the model to identify and localize different manipulation types effectively, even in scenarios where fine-tuning is not applied. The architecture of SPAN consists of three primary components: a feature extractor, a spatial pyramid attention block, and a decision module. Fig. 29 provides an overview of the complete framework of the Spatial Pyramid Attention Network.

The feature extraction component in SPAN utilizes the Wider & Deeper VGG Network [97, 98] as its backbone, integrating SRMConv2D [99] and BayarConv2D [100] layers to capture detailed features from both visual artifacts and noise patterns. After obtaining the embeddings from the pre-trained backbone, an additional convolutional layer is applied to adapt these features for input into the spatial attention module. The pyramid spatial attention propagation module is specifically designed to model spatial dependencies across multiple scales of the pixel representations. This module employs five recursive layers of local self-attention blocks to capture contextual information from neighboring pixels or patches. Each self-attention block includes a residual connection that links its input to its output, ensuring that fine-grained multi-scale details are preserved throughout the attention pathway. To produce the final output mask, a series of 2D convolutional layers is applied, followed by a Sigmoid activation function that generates a soft tampering mask, highlighting the regions of potential image manipulation.

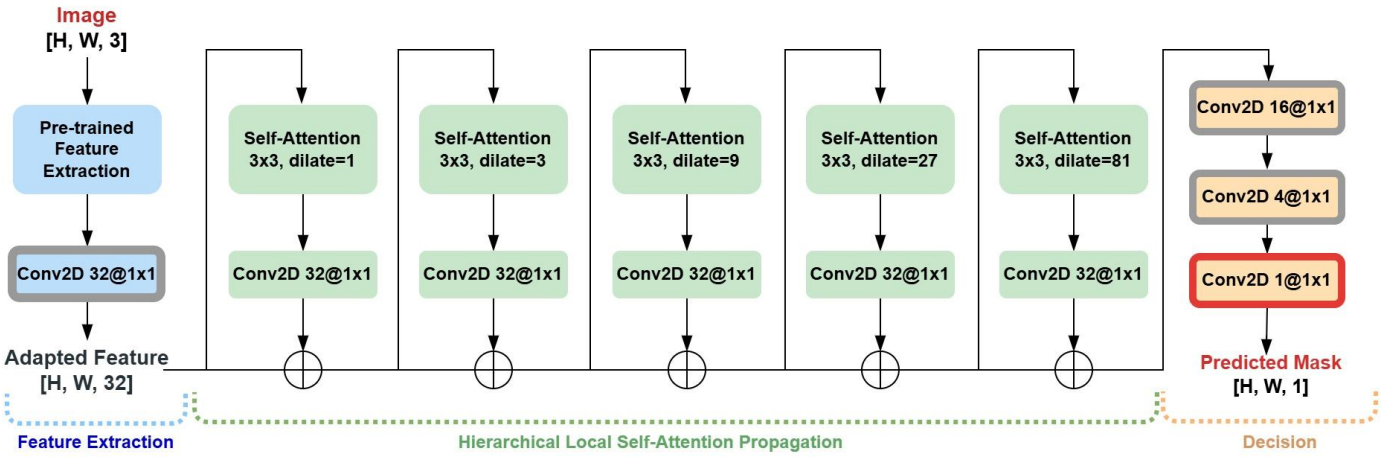

Fig. 29. Overview of the Spatial Pyramid Attention Network.

3.8.4. Spatial-Spectral Self-Attention (TSA)

Meng et al. [101] introduced a Spatial-Spectral Self-Attention (TSA) module designed to jointly capture spatial and spectral dependencies in a manner that is independent of input order. This module is integrated into an encoder-decoder architecture to facilitate high-quality image reconstruction. The detailed structure of the TSA module is depicted in Fig. 30.

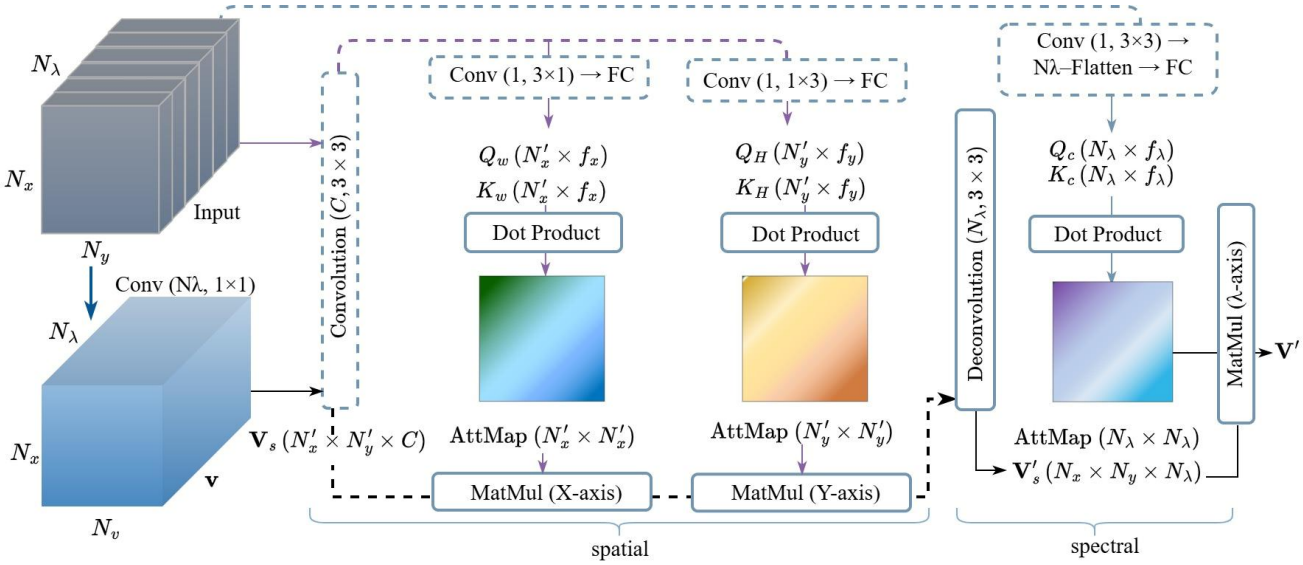

Fig. 30. Spatial-Spectral Self-Attention (TSA) module.

The modeling of spatial correlation in the TSA module is performed by processing the x-axis and y-axis separately, followed by an aggregation step that ensures order independence. Specifically, the input features are projected into query (Q) and key (K) representations for each spatial dimension independently. The kernel size and feature dimensions are defined individually for each axis, where the number of kernels corresponds to the number of attention heads, and the size determines the feature dimensionality. The Q and K representations specific to each dimension are then used to compute attention maps, which sequentially modulate their respective dimensions, x and y, ensuring that the overall process remains invariant to the order of operations. Once the attention modulation is complete, the features are passed through a deconvolution layer to complete the spatial correlation modeling.

In spectral correlation modeling, each spectral channel, represented as a two-dimensional spatial plane, is flattened into a feature vector to capture relationships between channels. The input features are projected into query (Q) and key (K) representations, which are then used to compute attention maps along the spectral axis. Due to consistent image patterns at corresponding spatial locations, adjacent spectral channels exhibit higher correlation. This phenomenon is captured through a spectral smoothness property embedded in the attention maps. To quantify similarity between spectral channels, a normalized cosine distance is employed, serving as the spectral embedding metric. These similarity scores are scaled and integrated with the coefficients in the attention maps, which are subsequently used to modulate the "Value" component within the self-attention mechanism. This modulation enforces a spectral smoothness constraint, enhancing the model's ability to preserve continuity and structure across the spectral dimension.

3.9. Channel & Spatial attention

Channel & spatial attention mechanisms integrate the strengths of both channel attention and spatial attention by adaptively focusing on the most relevant features and regions within an image [102]. Early advancements in this area were introduced by the Residual Attention Network [103] and the Convolutional Block Attention Module (CBAM) [79], both of which highlighted the significance of enhancing informative features across spatial and channel dimensions Subsequent research has expanded and refined these mechanisms for various applications. Notable examples include SCA-CNN [102], scSE [104], bottleneck attention module (BAM) [105], Channel and spatial attention network (CS-Net) [106], Dual attention network (DANet) [84], Channelized Axial Attention (CAA) [107], Relation-Aware Global Attention (RGA) [108], Coordinate Attention [109], and global-and-local attention (GALA) [110] each contributing different strategies to leverage joint spatial-channel attention in deep neural networks. In the following sections, we examine several of the more common models in greater detail.

3.9.1. Residual attention network

The Residual Attention Network [103] is a deep convolutional architecture that integrates attention mechanisms with residual connections, drawing inspiration from the ResNet framework [111]. This network is constructed by sequentially stacking multiple

attention modules, each designed to enhance feature representation. An individual attention module consists of two parallel branches: the trunk branch and the mask branch. The trunk branch is responsible for processing features and can adopt any advanced architectural component, such as a pre-activation residual unit or an inception block. In contrast, the mask branch employs a bottom–up top–down architecture to generate a soft attention mask, which shares the same spatial dimensions as the trunk output. This structure is intended to replicate the fast, feedforward-feedback attention mechanism observed in cognitive processes. The generated attention mask serves as a gating mechanism, modulating the activations of the trunk branch in a manner analogous to the gating functions in Highway Networks [112]. The overall output of an attention module H is computed by combining the modulated features from both branches .The output of Attention Module H is:

$$H_{i,c}(x) = M_{i,c}(x) * T_{i,c}(x) \tag{49}$$

where $c \in \{1, \dots, C\}$ is the channel index and i spans all spatial positions. End-to-end training is possible for the entire structure. Within each attention module, a bottom–up top–down feedforward architecture is employed to capture both spatial and cross-channel dependencies, contributing to consistent improvements in performance.

3.9.2. SCA-CNN

A convolutional neural network called SCA-CNN, which integrates spatial and channel-wise attentions into a CNN, is presented by Chen et al. [102] SCA-CNN dynamically adjusts the sentence production context in multi-layer feature maps during the image captioning task, encoding the visual attention's where (i.e., attentive spatial locations at various layers) and what (i.e., attentive channels). The overview of the CNN (SCA-CNN) model's spatial and channel-wise attentions is displayed on Fig. 31. The output of the $(l-1)$-th convolutional layer is the initial feature map V^l for the l-th layer. First, the channel-wise attention weights β^l are obtained using the channel-wise attention function Φ_c. These are then multiplied in the feature map's channel-wise. An attentive feature map X^l is then produced by multiplying the spatial attention weights α^l in each spatial region using the spatial attention function Φ_s.

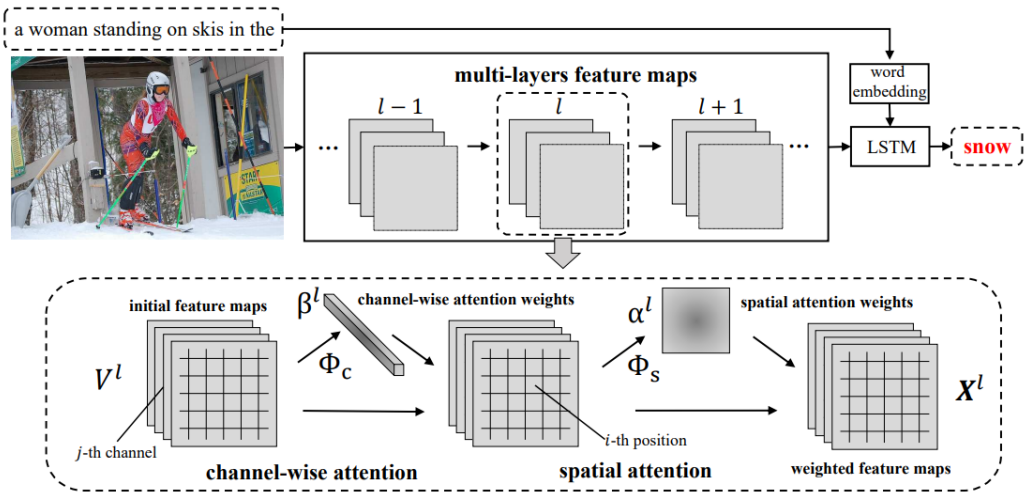

Fig. 31. The overview of SCA-CNN model [102].

3.9.3. Convolutional Block Attention Module (CBAM)

Woo et al. [79] introduced the Convolutional Block Attention Module (CBAM), which enhances feature representation by sequentially applying channel attention followed by spatial attention. This design allows the model to selectively emphasize the most informative feature channels as well as the most relevant spatial regions. An overview of the CBAM architecture is illustrated in Fig. 32. By combining channel attention (detailed in Section 3.5.3) and spatial attention (explained in Section 3.6) in a cascaded manner, CBAM effectively captures both cross-channel and spatial feature relationships. This enables the network to learn not only what to focus on but also where to focus, thereby improving task-specific performance.

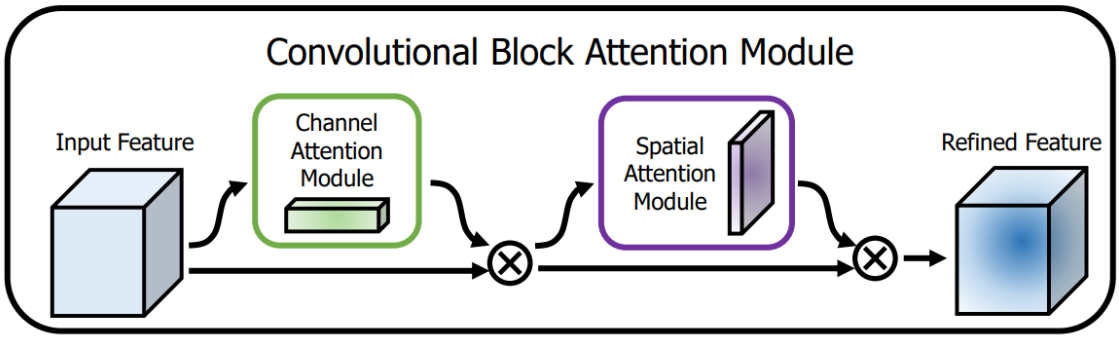

Fig. 32. The overview of CBAM [79].

3.9.4. Spatial and Channel SE blocks (scSE)

Roy et al. [104] introduce three distinct Squeeze-and-Excitation (SE) modules, namely cSE, sSE, and scSE, for the purpose of image segmentation. The cSE module (Channel Squeeze and Excitation) compresses spatial dimensions while enhancing channel-wise features. In contrast, the sSE module (Spatial Squeeze and Excitation) operates conversely by compressing channel information and emphasizing spatial features. The scSE module combines both strategies by independently recalibrating feature maps along the spatial and channel dimensions and then merging their outputs.

The scSE attention module builds upon the original SE mechanism and provides an enhanced approach tailored for image classification. It comprises two main components: the Spatial Squeeze and Channel Excitation (cSE) block and the Channel Squeeze and Spatial Excitation (sSE) block. The cSE block implements a channel-wise attention mechanism that aggregates feature information across channels, similar to conventional attention techniques such as SE. What distinguishes the cSE block is its use of a dimensionality reduction followed by an expansion on the attention weights, which facilitates efficient channel-wise integration while reducing computational complexity and improving processing speed. In parallel, the sSE block enhances spatial features by applying attention across spatial locations. Like the cSE block, it calculates attention weights and applies them to the input feature map; however, instead of utilizing Global Average Pooling (GAP) as in the cSE block, it unfolds the spatial dimensions. The sSE block employs a convolutional layer with a 1 × 1 kernel and a single output channel to integrate spatial information, thus avoiding reliance on GAP [113].

Fig. 33 illustrates the architectural structure of the scSE module. By combining the sSE and cSE blocks, the scSE attention mechanism simultaneously captures and integrates information across both spatial and channel dimensions. The outputs from each block are subsequently merged by summation along the channel axis.

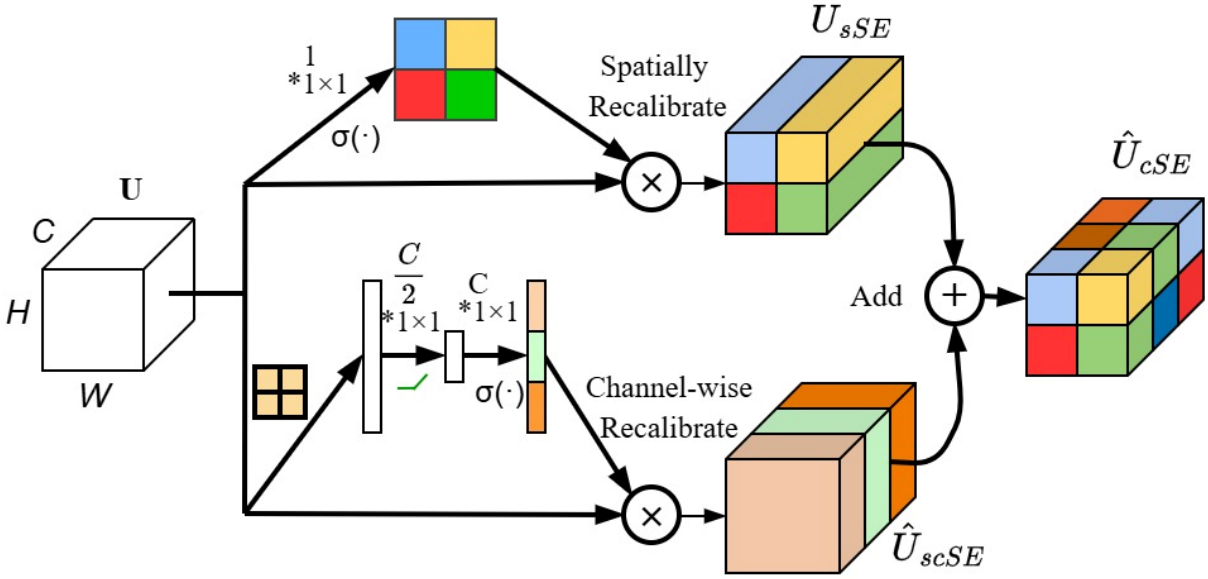

Fig. 33. The structure of scSE attention module.

3.9.5. Dual attention network (DANet)

In scene segmentation tasks, the presence of diverse objects and background elements such as variations in scale, lighting, and viewpoint poses significant challenges. Due to the inherently local receptive fields of convolutional operations, feature representations for pixels belonging to the same class may still exhibit noticeable variations. These discrepancies can lead to intra-class inconsistency, thereby reducing segmentation accuracy. To mitigate this issue, Fu et al. [84] proposed the Dual Attention Network (DANet), which leverages self-attention mechanisms to adaptively refine local semantic features. The approach incorporates two key components: a position attention module and a channel attention module, designed to capture global dependencies along spatial and channel dimensions, respectively. By modeling rich contextual relationships across local features, DANet significantly improves segmentation performance.

Fig. 34 presents an overview of the Dual Attention Network (DANet). This architecture incorporates two distinct attention modules that capture global contextual information from local features produced by a dilated residual network, thereby improving feature representations for pixel-wise prediction. Given an input feature map X, convolutional layers are first applied within the position attention module to produce intermediate feature maps. These are then processed through three steps to extract spatial long-range dependencies: (1) a spatial attention matrix is computed to model relationships between spatial positions; (2) this matrix is multiplied with the original features to aggregate contextual information; and (3) the result is combined with the initial features via an element-wise addition, yielding updated representations enriched with spatial context. In parallel, the channel attention module captures long-range dependencies across channels. Its procedure is similar to the position attention mechanism, except that the attention matrix is computed across channel dimensions instead of spatial ones. Finally, the outputs from both attention branches are fused, producing enhanced feature representations tailored for accurate pixel-level prediction.

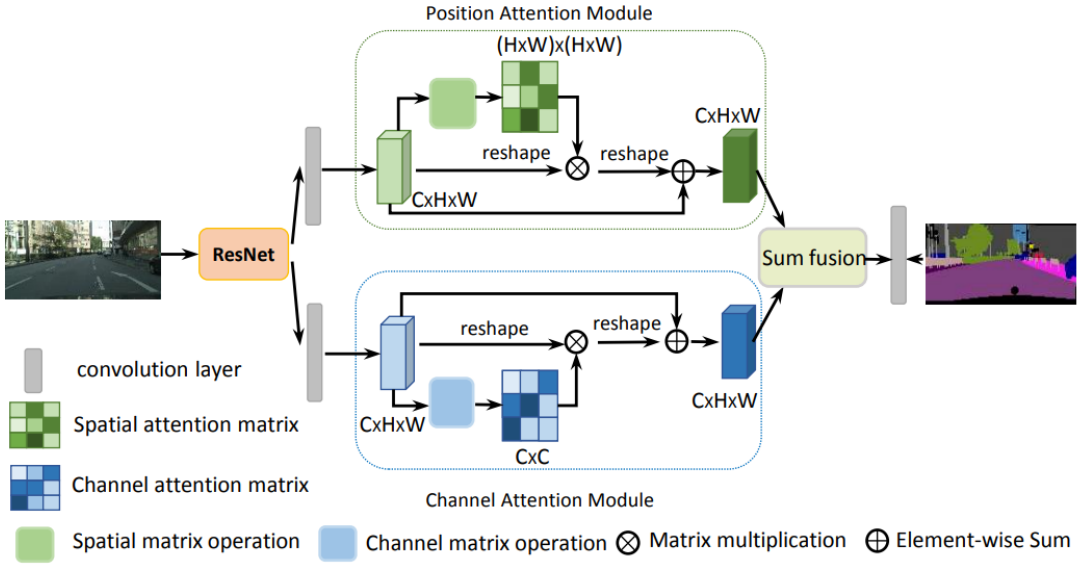

Fig. 34. An overview of the Dual Attention Network [84].

3.10. Temporal Attention

Temporal attention operates as a mechanism that dynamically selects important time steps, guiding the model toward the most relevant portions of the input sequence. It is widely applied in time-series analysis, where it enables the capture of both short- and long-term temporal dependencies. By focusing on informative time intervals, temporal attention helps models learn meaningful patterns across diverse time scales.

Li et al. [114] introduced the Global-Local Temporal Representation (GLTR) framework to capture multi-scale temporal cues in video sequences. The GLTR architecture comprises two primary components: the Dilated Temporal Pyramid (DTP) convolution module and the Temporal Self-Attention (TSA) mechanism. The DTP module applies multiple parallel dilated temporal convolutions [115] to extract short-term dependencies across neighboring frames. In contrast, the TSA module captures long-range temporal relationships, mitigating challenges such as occlusion and noise in video data. An overview of the GLTR architecture is shown on Fig. 35.

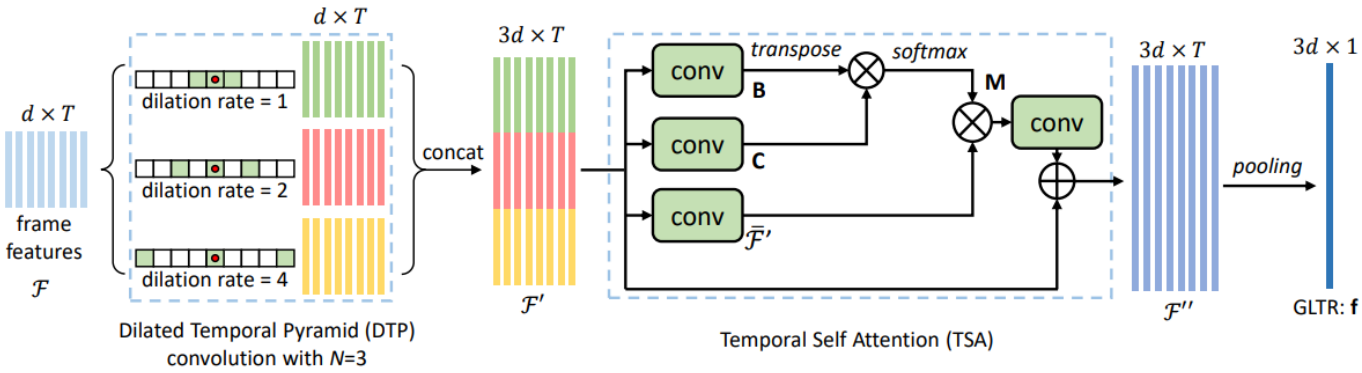

Fig. 35. An overview of the Global-Local Temporal Representation (GLTR) which consists DTP and TSA [114].

The core idea of the Temporal Self-Attention (TSA) module is to compute a $T \times T$ attention mask $\mathbf{M}$ that encodes the contextual relationships among all frame features. As shown in Fig. 33, given an input $F' \in R^{Nd \times T}$, TSA first applies two convolutional layers followed by Batch Normalization and ReLU to generate feature maps B and C of size $(Nd/\alpha) \times T$. A temporal attention mask $\boldsymbol{M}$ of size $T \times T$ is then obtained by multiplying $\boldsymbol{C}$ with the transpose of B. The attention mask M is applied to update the input feature F', embedding additional global temporal cues. First, F' is passed through a convolutional layer to produce an intermediate feature map $\bar{F}'$ of size $(Nd/\alpha) \times T$. This intermediate map is then multiplied by M and processed through another convolutional layer to restore its size to $Nd \times T$. Finally, the resulting feature map is combined with the original F' via a residual connection, yielding the refined temporal feature F''. The TSA computation can be formally expressed as

$$\begin{cases} F'' = W * (\bar{F}' \cdot M) + F' \\ F'' \in R^{Nd \times T} \end{cases} \tag{50}$$

Here, the final convolutional kernel is denoted by W. To simplify residual connection optimization, W is initialized to zero. In the TSA module, α regulates the parameter size. The final GLTR representation $f \in R^{Nd}$ is obtained by applying average pooling to the updated temporal feature F''.

In another study, Liu et al. [116] introduced a generic temporal module known as the Temporal Adaptive Module (TAM), designed to capture complex motion patterns in video sequences. TAM can be seamlessly integrated into existing 2D CNN architectures (e.g., ResNet) to construct a video network. As illustrated in Fig. 36, they further developed a powerful video architecture, TANet, based on this temporal module.

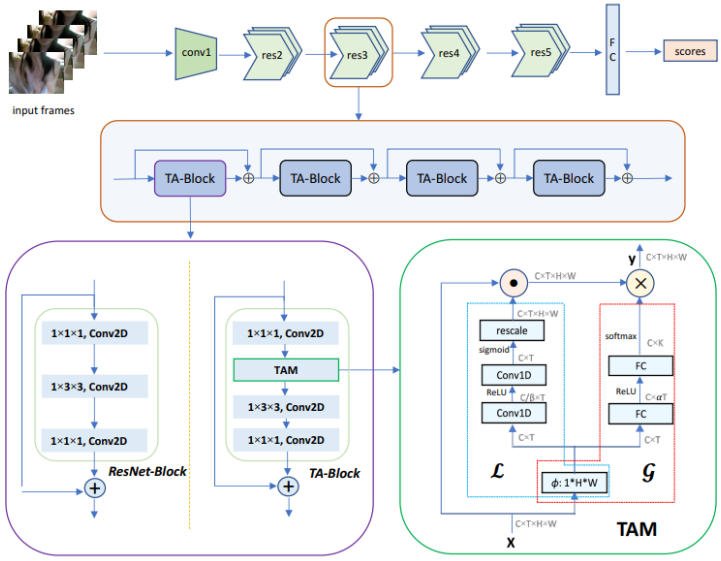

Figure. 36. The overall architecture of TANet [116].

The Temporal Adaptive Module (TAM) is designed with two guiding principles: high efficiency and strong flexibility. To minimize computational cost, the feature map is first reduced through global spatial pooling, after which TAM is applied in a channel-wise manner to preserve efficiency. It consists of two branches, namely a local branch (L) and a global branch (g). The local branch learns a location-sensitive importance map that enhances discriminative features, while the global branch produces location-invariant weights that adaptively integrate temporal information through convolution. The TAM is formally formulated as follows:

$$Y = \mathcal{G}(X) \otimes (\mathcal{L}(X) \odot X) \tag{51}$$

where $\otimes$ represents the convolution operation, and $\odot$ denotes element-wise multiplication.

The two branches target different aspects of temporal information: the local branch employs temporal convolution to capture short-term dependencies and emphasize important features, whereas the global branch leverages fully connected layers to model long-range temporal structure and guides adaptive temporal aggregation.

Cui et al. [117] proposed the ConvTransformer Attention Network (CTAN), which consists of two main components: the Temporal Attention Block (TAB) and the ConvTransformer Block (CTB), designed for temporal action detection. Inspired by attention mechanisms used in the image domain, the TAB dynamically emphasizes temporal positions that contain prominent action-related features. Meanwhile, the CTB models global transitions and captures fine-grained variations across adjacent video segments.

The Temporal Attention Block (TAB) assigns weights to each temporal position based on the pre-extracted feature $\{x_t\}_{t-1}^{T}$ When a temporal position contains action-related cues, such as motion information, TAB emphasizes that position to enhance its contribution. In contrast, if a temporal position primarily carries redundant details, such as background information, TAB suppresses it to reduce noise.

As illustrated in Fig. 37, the Temporal Attention Block (TAB) introduces a novel attention mechanism that exploits inter-temporal dynamics in video sequences. To achieve this, both average pooling and max pooling are applied across the channel axis to extract temporal information from the feature map. This operation compresses the channel dimension, producing two temporal representations: $F_{avg}(x) \in \mathbb{R}^{T\times 1}$ from average pooling, which aggregates channel-wise information and captures motion cues, and $F_{max}(x) \in \mathbb{R}^{T\times 1}$ for the max-pooled features. from max pooling, which preserves the most discriminative features. These descriptors are then concatenated to form a joint feature representation $\psi(x) \in \mathbb{R}^{T\times 2}$, enriching the overall feature representation. The combined feature descriptor is subsequently processed by an adaptive convolution block composed of a 1D channel convolution, Layer Normalization (LN), and a ReLU activation. This block analyzes the fused representation to determine whether max-pooled or average-pooled features are more informative for capturing action-related patterns, thereby producing an initial attention map $M(x) \in \mathbb{R}^{T\times 1}$. The attention map is then normalized using a sigmoid function, which is applied after the convolution to dynamically emphasize the most relevant features. This step forms the essence of the TAB attention mechanism and yields the final attention map, which can be expressed as:

$$\psi(x) = [F_{avg}(x) \cdot F_{max}(x)] \tag{52}$$

$$M(x) = \sigma\left(Conv(\psi(x))\right) \tag{53}$$

Here, σ, denotes the sigmoid function, $Conv$ represents the 1D convolution layer followed by Layer Normalization and ReLU activation, and $[\cdot]$ indicates the concatenate operation. Given the attention map, a non-linear transformation is applied to the pre-extracted features x_t using a multi-layer perceptron (MLP) with two fully connected (FC) layers. To minimize parameter overhead, the MLP employs a hidden activation size of $\mathbb{R}^{T\times D}$.

The feature representation derived from the TAB can be mathematically expressed as follows:

$$f_{TAB} = LN(MLP(x) \otimes M(x)) = LN\left(\left(w_1\left(ReLU(w_0(x))\right)\right) \otimes M(x)\right) \tag{54}$$

Here, $\otimes$ represents the direct product operation, which combines the outputs of the MLP with the matrix $M(x)$, thereby enhancing the interaction among features extracted and processed by the TAB. LN and $ReLU$ refer to the layer normalization and ReLU activation function, respectively. The weight matrices are denoted as $W_0 \in \mathbb{R}^{T\times D}$ and $W_0 \in \mathbb{R}^{T\times D_0}$. Furthermore, $D = D_0 / r$, where r is the reduction ratio, and the value of r is specified in the experiments section.

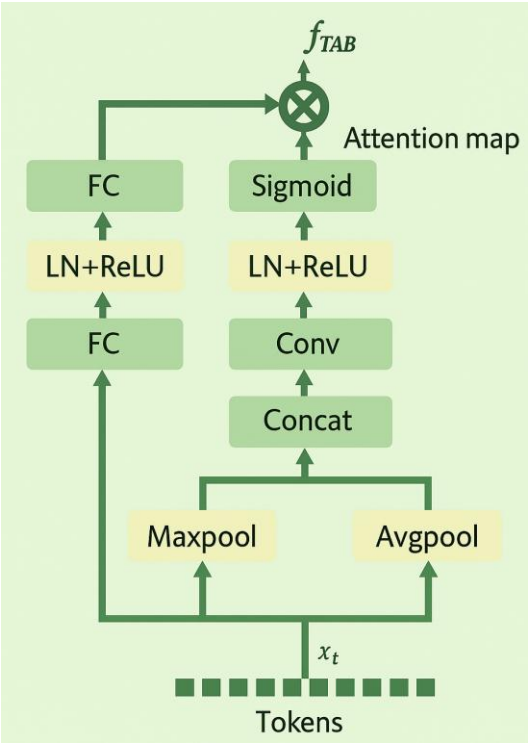

Fig. 37. Temporal Attention Block.

3.11. Spatial-Temporal Attention

Spatial and temporal attention jointly exploit the strengths of both mechanisms by adaptively selecting salient spatial regions and key frames. Numerous studies have applied spatiotemporal attention across diverse time-series tasks, with a particular focus on video analysis [118-125].

Song et al. [126] propose an end-to-end multi-layered LSTM network that integrates both spatial and temporal attention mechanisms for action recognition. The spatial attention module automatically highlights the most important joints within each frame, while the temporal attention module assigns varying levels of importance across different frames. Fig. 38 illustrates the overall architecture, which consists of a main LSTM network along with spatial and temporal attention subnetworks. Owing to the strong interdependence among these three components, training the network remains a considerable challenge.

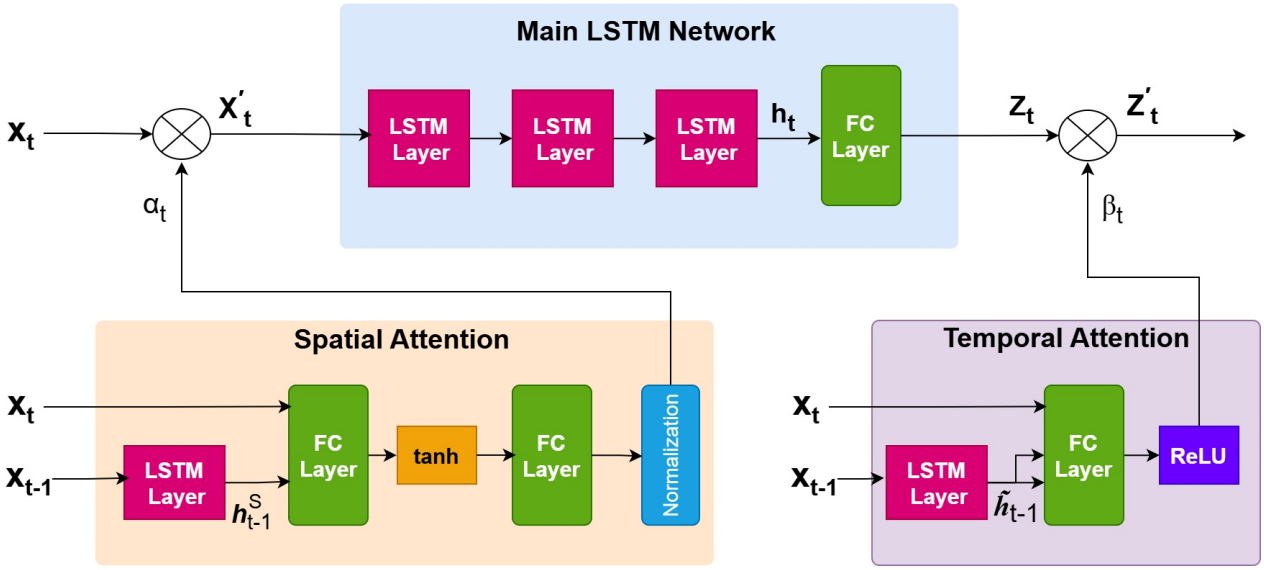

Fig. 38. Overall design of an LSTM-based spatial-temporal attention network.

Yan et al. [127] introduced a novel spatial-temporal attention mechanism (STAT) within an encoder-decoder neural network for video captioning. When describing a video, humans generally emphasize semantically important spatial-temporal segments rather than recounting every detail. Following this intuition, before generating each target word, the decoder first identifies semantically relevant objects in each frame through spatial attention, and then applies temporal attention to capturing their movements and interactions across consecutive frames. Conventional temporal-only approaches often fail to detect crucial regions within frames, leading to recognition errors and loss of important details. The proposed STAT overcomes these issues by jointly modeling spatial and temporal structures, enabling the decoder to select significant regions within the most relevant temporal segments for accurate word generation.

Liu et al. [128] introduced the Residual Spatial-Temporal Attention Network (R-STAN), a feed-forward convolutional network that integrates residual learning with spatial-temporal attention for video action recognition. This framework enhances the network's focus on both spatial and temporal discriminative features. R-STAN adopts a two-stream architecture, where each stream is composed of stacked Residual Spatial-Temporal Attention Blocks (R-STABs). The spatial-temporal modules embedded within these residual blocks generate attention-aware representations across spatial and temporal dimensions, effectively minimizing redundant information. By leveraging residual learning, the model is able to construct a highly deep network capable of capturing complex spatial-temporal dependencies from video data. As the depth increases, the attention-aware features produced by successive R-STABs adapt and evolve accordingly. The architecture of the Residual Spatial-Temporal Attention Block (R-STAB) is shown on Fig. 39.

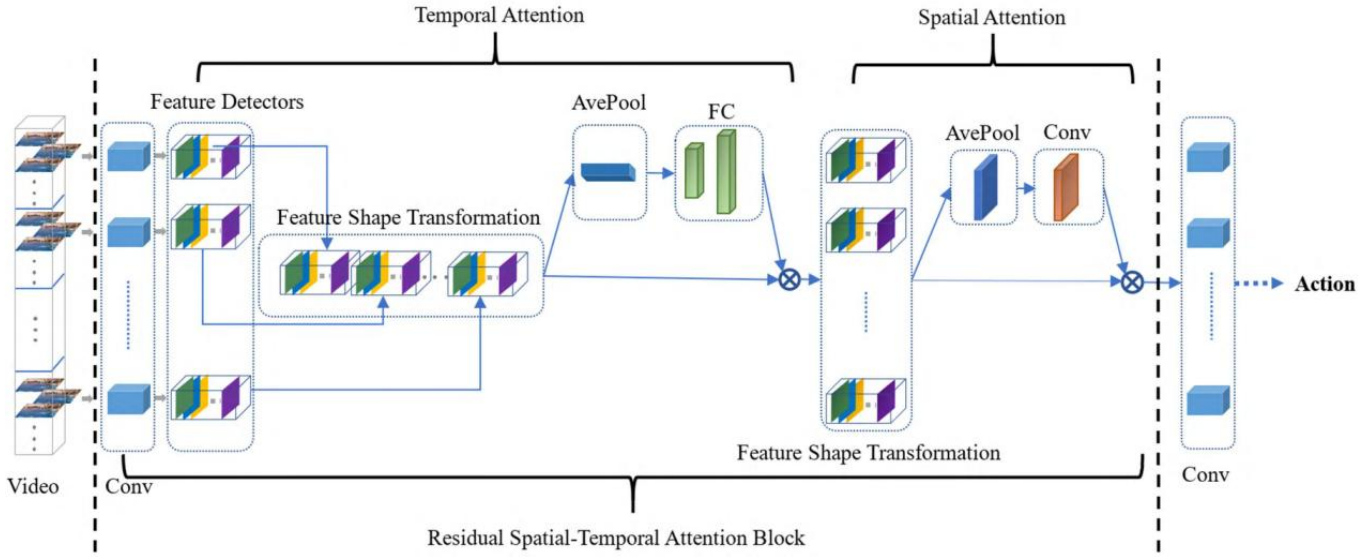

Fig. 39. The architecture of residual spatial-temporal attention blocks (R-STAB) [128].

Each block incorporates conventional convolutional layers together with temporal and spatial attention modules. To capture temporal weights across video segments, the feature detectors generated by the residual block are first transformed into feature detectors that represent a single segment. This transformation reframes the task of temporal weight learning as channel-wise weight learning. A spatial-wise average pooling operation is then applied to compress spatial information, followed by fully connected layers that extract temporal attention features and adjust the transformed feature detectors. The detectors are subsequently remapped to represent multiple video segments. Spatial attention is then derived using average pooling and a convolutional layer. Finally, the processed feature detectors are propagated to the next R-STAB in the network.

3.12. Cross Attention

Cross attention refers to the process where each element of a query sequence focuses on elements within a separate key/value sequence in order to extract the most relevant information.

Lee et al. [129] introduce Stacked Cross Attention, a framework designed to uncover latent alignments by leveraging both image regions and sentence words as contextual cues to infer image-text similarity. The method applies cross attention in two sequential stages: Image-to-Text and Text-to-Image. In the Image-to-Text stage, the model attends to sentence words relative to each image region, then evaluates the importance of each region by comparing it with the attended sentence information to determine whether the region is mentioned. Similarly, in the Text-to-Image stage, the model attends to image regions for each word and then adjusts attention weights across the words accordingly.

The image-text cross attention process involves three steps: (a) computing cosine similarity for all image-text pairs; (b) calculating a weighted sum of the attention scores, where image-based attention is obtained via softmax; (c) applying LogSumExp pooling [130] to derive the final similarity score. These steps are mirrored for the Text-to-Image direction, except that in second step the weights are computed with a text-based softmax.

Hou et al. [131] introduced the Cross Attention Network (CAN) to improve feature discriminability in few-shot classification. To address the unseen class challenge, they proposed the Cross Attention Module (CAM), which draws inspiration from human strategies in few-shot learning. When recognizing samples from unseen classes with only a few labeled examples, humans typically identify the most relevant regions between labeled and unlabeled samples. Similarly, CAM constructs a cross-attention map from both the class feature map and the query feature map, enabling the model to emphasize the target object. This process relies on correlation estimation and meta fusion to enhance discriminative capability by weighting features with cross attention maps. In addition, a transductive inference algorithm is employed to mitigate the data-scarcity problem by iteratively incorporating unlabeled queries into the support set, thereby making the class features more representative.

The Cross Attention Module (CAM) models the semantic relationship between class and query features, enabling the network to concentrate on target objects and thereby strengthen the subsequent matching process. Fig. 40 depicts the architecture of the CAM.

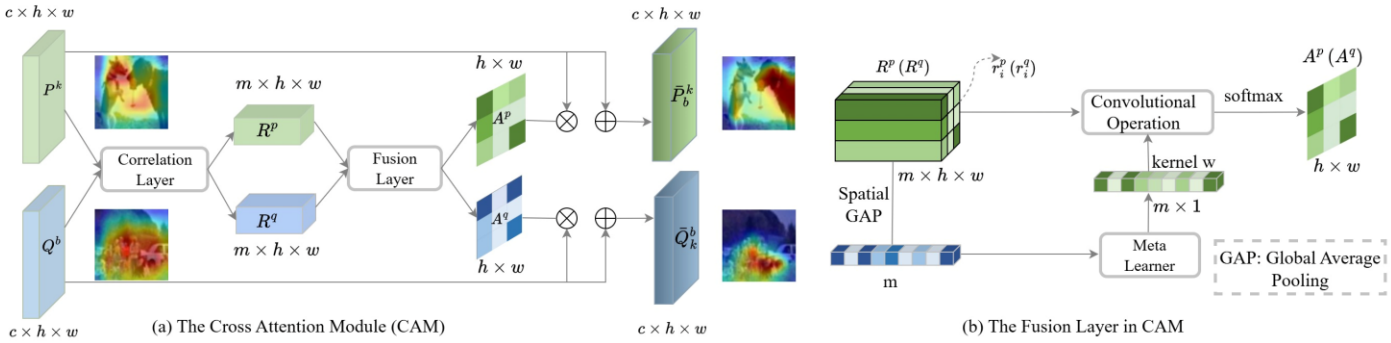

Fig. 40. (a) Cross Attention Module (CAM). (b) The Fusion Layer in CAM.

CAM employs two main components: a correlation layer and a fusion layer. The correlation layer computes a correlation map between input class feature map (P) and the query feature map (Q) using cosine distance, which guides the generation of cross-attention maps. The fusion layer then takes the class correlation map R^p as input and applies a convolutional operation with an $m \times 1$ kernel to fuse each local correlation vector into an attention scalar. A softmax function normalizes these scalars to obtain the class attention at the i^{th} position. Furthermore, a meta-learner adaptively generates the kernel based on the correlation between the class and query features. To achieve this, a global average pooling (GAP) operation is applied to extract an averaged query correlation vector, which is subsequently fed into the meta-learner.

3.13. Axial Attention

A straightforward extension of self-attention, axial attention [132] achieves an $O(N^{(d-1)/d})$ savings in computation and memory for a d-dimensional input tensor with N elements. It scales more efficiently with the dimension of input data. On modern accelerators, axial attention is simple to implement and does not require custom kernels to function.

Axial attention maintains the original dimensions of the input data tensor and applies either masked or unmasked attention along one axis at a time. It focuses on a chosen axis k of the tensor x, mixing information along that axis while treating the other axes as independent. This method is straightforward to implement. A common approach involves transposing the tensor so that axis k aligns with the batch axis, applying standard attention mechanisms as a subroutine, and then reversing the transpose. Alternatively, this can be implemented using the Einsum operation, which is available in most deep learning libraries.

When the data is an image, column attention is applied to allow information to be mixed within columns while preserving the independence of different columns. For analogous reasons, row attention is also required. On a square image of size $N = S \times S$, axial attention operates on S sequences of length S, resulting in a total computation of $O(S.S^2 = O(N\sqrt{N})$. This corresponds to an $O(\sqrt{N})$ reduction in computation compared to standard self-attention. More generally, for a d-dimensional tensor with $N = S^d$, axial attention achieves $O(N^{(d-1)/d})$ computation relative to standard attention. Since a single axial attention layer along one axis does not cover the entire receptive field, stacking two axial attention layers allows the model to capture a global receptive field

Axial attention can be directly integrated into conventional Transformer layers to form Axial Transformer layers. These layers

retain the fundamental components of the original Transformer architecture, relying on the same core building blocks.

3.14. Flash Attention

FlashAttention is an optimized algorithm designed to accelerate the attention mechanism while reducing memory consumption. Transformers are often computationally expensive and memory-intensive on long sequences, as the time and memory complexity of self-attention scale quadratically with sequence length. Although various approximate attention methods have been proposed to lower computational complexity at the expense of model quality, they generally fail to provide significant improvements in actual wall-clock runtime.

Dao et al. [133] argue that a crucial yet often overlooked aspect in attention algorithms is IO-awareness. This involves accounting for memory read and write operations across different levels of GPU memory, including the faster on-chip SRAM and the slower high-bandwidth memory (HBM) [134].

To address this, they introduced FlashAttention, an attention algorithm that computes exact attention while significantly reducing memory access. Its primary goal is to avoid repeated reading and writing of the attention matrix to and from HBM. This presents two main challenges: (i) performing the softmax reduction without access to the complete input, and (ii) avoiding storage of the large intermediate attention matrix required for the backward pass. To overcome these challenges, two established strategies are applied. First, the attention computation is reorganized by dividing the input into blocks and processing them in multiple passes, which enables the softmax reduction to be performed incrementally. Second, the softmax normalization factors from the forward pass are stored so that attention during the backward pass can be recomputed directly on-chip, which is faster than retrieving the intermediate attention matrix from HBM.

FlashAttention is implemented in CUDA to enable precise control over memory access and to merge all attention operations into a single GPU kernel. Despite the increased number of floating point operations (FLOPs) [135] caused by recomputation, the algorithm executes faster and consumes less memory than standard attention, primarily due to the substantial reduction in HBM access.

They further demonstrated that FlashAttention can serve as a foundational component for realizing the potential of approximate attention algorithms by mitigating their memory access overhead. As a proof of concept, they introduced block-sparse FlashAttention, a sparse attention variant that achieves a 2-4X speedup over FlashAttention and scales to sequence lengths of up to 64k. This approach demonstrates that block-sparse FlashAttention attains superior IO complexity compared to FlashAttention, with improvements proportional to the degree of sparsity.

FlashAttention leverages the asymmetric GPU memory hierarchy to deliver significant memory savings and runtime acceleration. Nevertheless, its performance remains well below that of optimized general matrix-multiply (GEMM) operations [136] achieving only 25–40% of the theoretical peak FLOPs/s. This inefficiency arises from suboptimal work partitioning across thread blocks and warps on the GPU, resulting in low occupancy or redundant shared memory operations. To address these limitations, FlashAttention-2 [137] was introduced with enhanced work partitioning. Specifically: (1) the algorithm is refined to reduce the number of floating point operations that are not matrix multiplications, (2) attention computation is parallelized, even for a single head, across multiple thread blocks to increase occupancy, and (3) tasks within each thread block are more evenly distributed among warps to reduce communication through shared memory. Collectively, these optimizations yield nearly a twofold speedup over the original FlashAttention.

FlashAttention-3 [138] was also introduced, incorporating three key techniques to further improve performance on newer GPU architectures. These are: (1) overlapping computation and data movement through warp specialization, (2) interleaving block level matrix multiplication with softmax operations, (3) applying block quantization and incoherent processing that takes advantage of hardware support for FP8 (8-bit Floating Point) [139] low-precision arithmetic.

4. Transformer

As previously noted, the Transformer is one of the most prominent deep learning architectures that effectively employs the attention mechanism [20]. It operates using a self-attention-based encoder-decoder framework. The encoder is composed of a stack of identical layers, each containing two primary sublayers: the first is a multi-head self-attention mechanism, and the second is a position-wise fully connected feed-forward network. Additionally, residual connections [140] and layer normalization [141] are applied around each sublayer to facilitate stable and efficient training. The decoder generates the output sequence based on the intermediate representation produced by the encoder. Like the encoder, it also consists of a stack of identical layers, but with an added third sublayer. This additional sublayer applies multi-head attention over the encoder's output, enabling the decoder to attend to relevant parts of the encoded input. Like the encoder, each sublayer in the decoder is wrapped with residual connections and normalization [142]. The complete architecture of the Transformer, illustrating both the encoder (left) and decoder (right), is shown in Fig. 41.

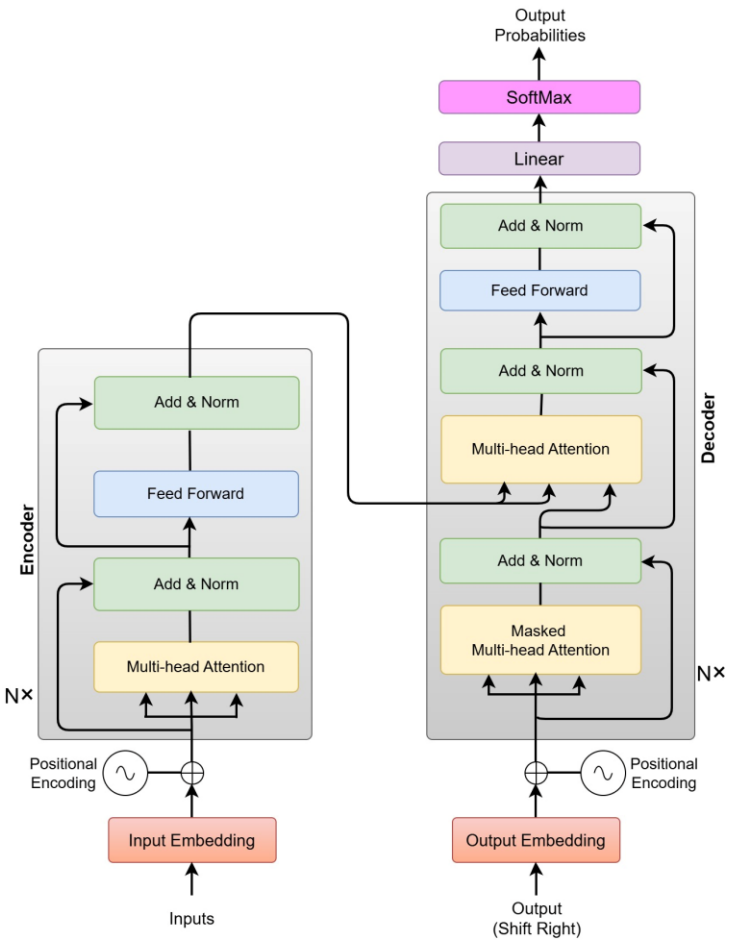

Fig. 41. The architecture of the Transformer model.

Multi-Head Attention: The Transformer architecture utilizes a multi-head self-attention mechanism to improve its capacity for capturing relationships and dependencies among elements within a sequence (as discussed in Section 3.3).

Position-wise Feed-Forward Networks: In addition to the attention sublayers, each layer of both the encoder and decoder includes a position-wise feed-forward network. This component is independently applied to each position in the sequence using the same transformation across all positions. The feed-forward network consists of two linear layers separated by a ReLU activation function, enabling non-linearity and richer representations.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2 \tag{55}$$

Positional Encoding: Because the Transformer architecture does not rely on recurrence or convolution, it lacks an inherent mechanism to capture the sequential order of tokens. To compensate for this, positional encoding is introduced at the input stage of both the encoder and decoder stacks. These encodings are added to the input embeddings, leveraging the fact that both share the same dimensionality, d_{model}. This addition enables the model to incorporate information about the relative and absolute positions of tokens, thereby enhancing its understanding of sequence structure.

In the original Transformer design, positional information is encoded using sine and cosine functions at varying frequencies:

$$\begin{cases} PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \end{cases} \tag{56}$$

where pos denotes the position and i represents the dimension. Each dimension of the positional encoding follows a sinusoidal pattern, with wavelengths ranging geometrically from 2π to 10000· This design was chosen to enable the model to easily capture relative positional relationships. Notably, for any fixed offset k, the encoding PE_{pos+k} be represented as a linear function of PE_{pos}.

Traditional sequence-to-sequence (Seq2Seq) models based on recurrent neural networks (RNNs) can be effectively replaced by attention-based architectures. In self-attention layers, the query, key, and value vectors are all derived from the same input sequence through different projection matrices. Unlike RNN, which require sequential, time-consuming training due to their iterative nature, Transformer models allow for fully parallelized training, enabling simultaneous learning of all input features. This significantly enhances computational efficiency and greatly reduces the training time required [143, 144].

Transformer architecture has demonstrated remarkable adaptability, leading to the development of numerous variants tailored to address domain-specific challenges. While the fundamental principles of the Transformer remain consistent, ongoing research has introduced various enhancements to improve its performance and adaptability for specialized applications.

Transformer models are numerous and have been developed across a wide range of architectures, each tailored to specific tasks, data types, or performance requirements. They can be grouped in various ways based on their internal structure, intended application, training methodology, and other distinguishing characteristics. In this article, we aim to investigate the most common and practically useful among them. To bring clarity to this diverse landscape, we propose a principled taxonomy for the Transformer models covered in this survey, grounded in the application domains and architectural innovations introduced in their original papers. Our classification not only helps to highlight the evolution of the Transformer family but also serves as a practical guide for researchers and practitioners seeking to select or design models for specific problems. Fig. 42 presents a visual summary of the Transformer models discussed in this article, arranged according to the proposed taxonomy.

We first examine models developed for Natural Language Processing (NLP) tasks, which can be divided into four subgroups. The first subgroup consists of encoder-only models, typically used for Natural Language Understanding (NLU) tasks such as text classification, named entity recognition (NER), and question answering (QA). The second subgroup comprises decoder-only models, which are designed for Natural Language Generation (NLG) tasks, including text generation, code synthesis, and creative writing. The third subgroup covers encoder-decoder models, commonly applied to sequence-to-sequence (seq2seq) tasks that require transforming an input sequence into an output sequence, examples include summarization, machine translation, and paraphrase generation. The last subgroup includes models specifically developed for processing long contexts, such as handling entire books, and long documents, where standard Transformers would struggle due to quadratic memory constraints.

Next, we analyze a number of the most common Transformer models developed for computer vision tasks. We first examine patch-based models, which are widely used in image classification by dividing an image into fixed-size patches and treating them similarly to word tokens in NLP. The second group comprises hierarchical models, which progressively reduce spatial resolution while increasing feature depth, enabling efficient handling of large-scale images and dense prediction tasks. Additionally, we analyze specialized models such as DETR (Detection Transformer), designed for object detection, and ViViT (Video Vision Transformer), developed for video recognition.

Lastly, we analyze Transformer models developed to achieve substantial gains in computational and memory efficiency. These models are particularly well-suited for deployment in resource-constrained environments, such as edge devices, mobile phones, or embedded systems, as well as for real-time tasks like online translation, speech recognition, and interactive AI assistants. By reducing the quadratic complexity of standard self-attention, these architectures enable practical deployment of Transformer-based solutions in settings where computational budgets are limited, without sacrificing task performance.

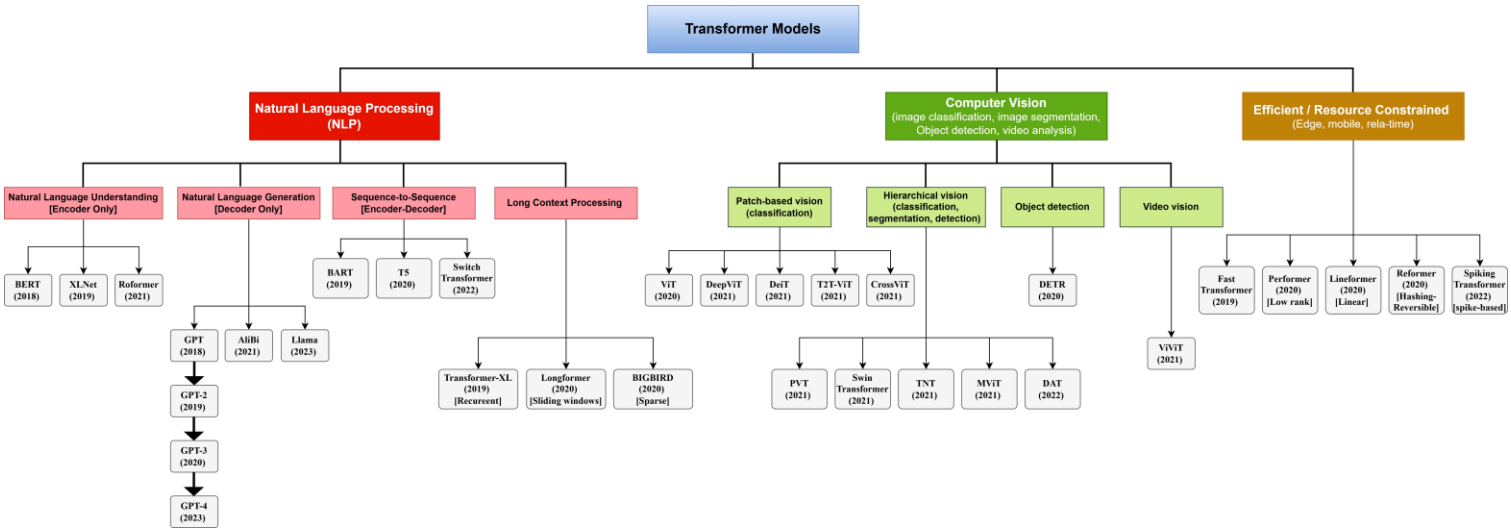

Fig. 42. Visual taxonomy of Transformer models.

4.1. BERT

Bidirectional Encoder Representations from Transformers (BERT) [145] is a multi-layer, bidirectional Transformer encoder specifically designed for unsupervised pre-training in natural language understanding (NLU) tasks. BERT is structured to learn deep bidirectional representations by conditioning on both the left and right context across all layers, allowing it to capture richer language features compared to unidirectional models. The BERT framework consists of two main phases: pre-training and fine-tuning. During the pre-training stage, the model is trained on large-scale unlabeled text using various pre-training objectives. In the fine-tuning phase, the pre-trained BERT model is adapted to specific downstream tasks by initializing it with the learned parameters and updating all parameters using labeled task-specific data. Although all downstream tasks start with the same pre-trained model, each task is associated with its own fine-tuned version. Fig. 43 illustrates the BERT architecture applied to a question-answering task.

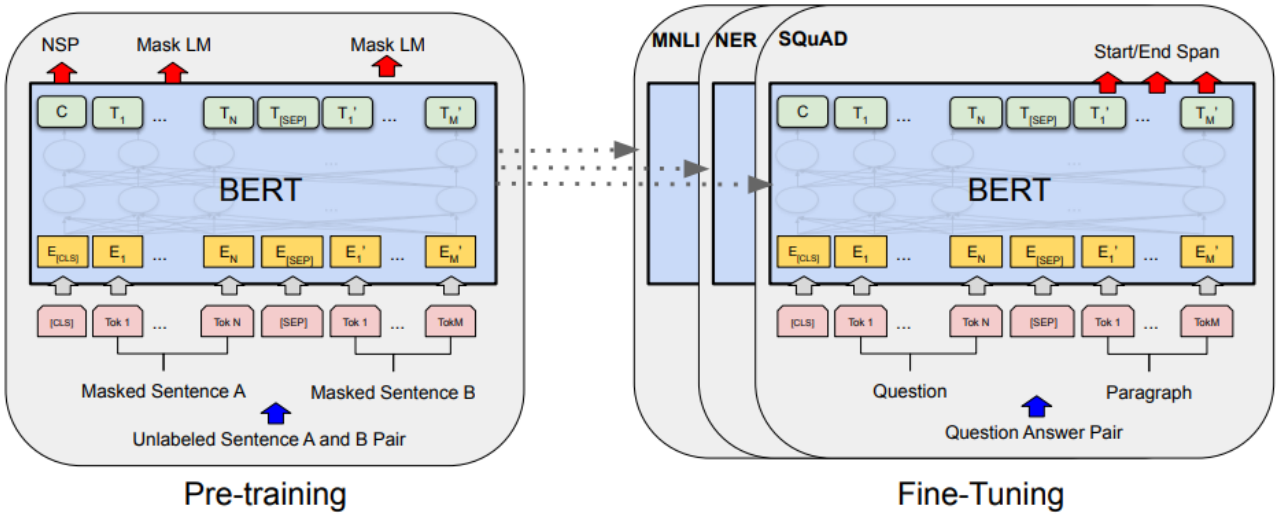

Fig. 43. Overall pre-training and fine-tuning procedures for BERT [145].

To enable BERT to effectively handle a wide range of downstream tasks, its input representation is designed to clearly encode both single sentences and sentence pairs (e.g., Question, Answer) within a single token sequence. BERT utilizes WordPiece embeddings [146] with a vocabulary size of 30,000 tokens. Each input sequence begins with a special classification token ([CLS]), whose final hidden state is used as a holistic representation of the sequence for classification-related tasks. When processing sentence pairs, both sentences are combined into a single token sequence. To distinguish between them, BERT uses two mechanisms: (1) a special separator token ([SEP]) is inserted between the sentences, and (2) a segment embedding is added to each token to indicate whether it belongs to sentence A or sentence B. This structured input format allows BERT to model relationships between sentences effectively.

Fig. 44 shows that the input embedding is denoted by $\boldsymbol{E}$, the final hidden representation of the special [CLS] token is represented as $C \in \mathbb{R}^H$, and the final hidden vector corresponding to the i^{th} input token is denoted as $T_i \in \mathbb{R}^H$. Each input embedding is computed as the sum of the token embeddings, segmentation embeddings, and positional embeddings.

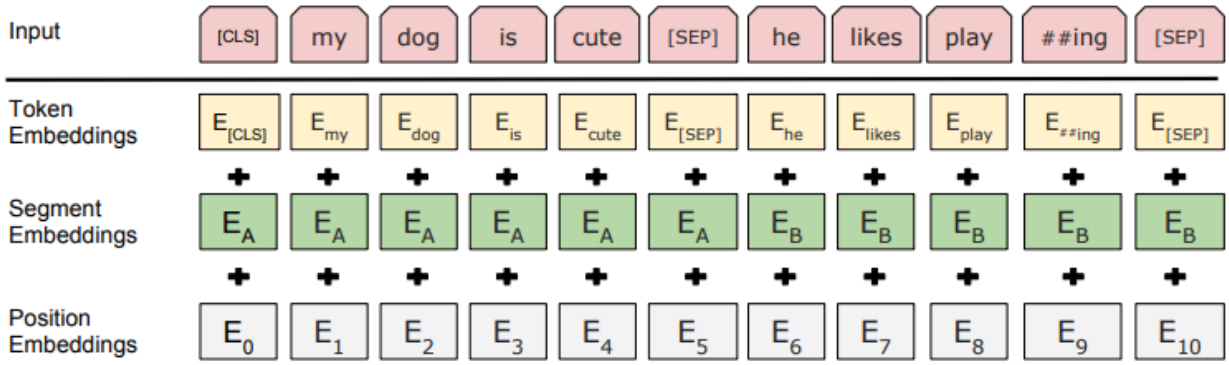

Fig. 44. BERT input representation [145].

Several studies have modified the BERT model to achieve better optimization. ALBERT [147] is designed to enhance BERT's training speed and reduce memory usage by introducing two parameter reduction techniques. The first technique is factorized embedding parameterization, where the large vocabulary embedding matrix is decomposed into two smaller matrices, effectively decoupling the size of the hidden layers from the vocabulary embedding size. This allows for increasing the hidden dimensions without significantly enlarging the vocabulary parameters. The second technique involves cross-layer parameter sharing, which restricts parameter growth as the network depth increases. In a similar direction, TinyBERT [148] has been proposed to speed up inference and shrink model size while preserving accuracy. Additionally, PoWER-BERT [149] has been developed to enhance BERT's inference efficiency.

RoBERTa [150] demonstrates that the original BERT model was significantly undertrained. Its performance can be notably enhanced by extending training time, using larger batch sizes with more data, eliminating the next sentence prediction (NSP) objective, training on longer sequences, and applying dynamic masking strategies during training. In particular, RoBERTa adopts dynamic masking, removes the NSP loss by training on full sentences, utilizes large mini-batches, and incorporates a larger byte-level BPE (Byte-Pair Encoding) vocabulary [151]. Similarly, SpanBERT [152] is introduced as a pre-training approach to improve representation learning and better predict text spans. SpanBERT builds upon BERT by (1) masking continuous random spans instead of individual tokens, and (2) training the span boundary representations to reconstruct the full content of the masked span, without depending on token-level predictions within the span. On the other hand, DistilBERT [153] shows that comparable performance to BERT can be achieved with significantly smaller models by employing knowledge distillation techniques [154].

DeBERTa [155] enhances RoBERTa by introducing two innovative techniques. The first is the disentangled attention mechanism, in which each word is represented by two separate vectors: one for content and one for position. The attention scores between words are computed through disentangled matrices that independently account for content and relative positional information. The second technique involves an improved mask decoder that incorporates absolute positional information within the decoding layer to better predict masked tokens during pre-training.

In addition, a range of models has been developed based on or inspired by BERT architecture to serve domain-specific applications. These include FineBERT [156], MentalBERT [157], DocBERT [158] BERTimbau [159], ClimateBERT [160], RareBERT [161], BioBERT [162], and EHR-BERT [163].

4.2. GPT

The Generative Pre-Trained Transformer (GPT), developed by OpenAI, is a class of Transformer-based models that has demonstrated exceptional performance in natural language processing (NLP) tasks through a two-phase training approach: unsupervised pre-training followed by supervised fine-tuning.

In 2018, Radford et al. [164] introduced the original GPT framework, which enabled strong natural language understanding using a task-agnostic model. The approach involves pre-training the model on a large and diverse corpus of unlabeled text, enabling it to acquire substantial world knowledge and effectively capture long-range dependencies. This foundational knowledge is then leveraged during a fine-tuning phase on various discriminative tasks such as question answering, semantic similarity, entailment recognition, and text classification. Their results showed that generative pre-training followed by discriminative fine-tuning could significantly improve performance across these tasks. A notable feature of their method is the use of task-aware input formatting during fine-tuning, which facilitates effective knowledge transfer with minimal changes to the model architecture. The fine-tuning process introduces only a small number of task-specific parameters, and downstream tasks are addressed by simply fine-tuning all pre-trained weights.

Subsequent iterations of the model expanded its capabilities: GPT-2 [165] was released in 2019, followed by GPT-3 [166] in 2020, and GPT-4 [167] in 2023, each bringing improvements in scale, performance, and generalization.

4.3. Transformer-XL

Transformer-XL [168], short for Transformer-Extra Long, was introduced to improve language modeling by enabling the capture of dependencies that extend beyond fixed-length contexts, all while maintaining temporal consistency. The model incorporates two key innovations: a segment-level recurrence mechanism, and a relative positional encoding scheme. Together, these components allow Transformer-XL to effectively model long-range dependencies and address the problem of context fragmentation, which often limits standard Transformer architectures.

Segment-level Recurrence Mechanism: To overcome the constraints imposed by a fixed-length context window, Transformer-XL incorporates a segment-level recurrence mechanism into its architecture. Specifically, the hidden state representations computed while processing each segment are cached and subsequently provided as supplementary context when the model processes the following segment. Although gradients do not propagate across segment boundaries, this design enables the network to draw upon information from earlier segments, thereby extending its effective context range, enhancing its capacity to model long-range dependencies, and preventing context fragmentation. Figure 44 illustrates the Transformer-XL model with a segment length of 4.

To formalize this mechanism, consider two consecutive segments of length L, defined as $S_{\mathcal{T}} = \left[S_{\mathcal{T},1}, \ldots\ldots, S_{\mathcal{T},L}\right]$ and $S_{\mathcal{T}+1} =$

$[S_{\mathcal{T}+1,1}, \ldots\ldots, S_{\mathcal{T}+1,L}]$. Let the n-th layer hidden state sequence produced for the $\mathcal{T}$-th segment $S_{\mathcal{T}}$ be denoted by $h_{\mathcal{T}}^{n} \in \mathbb{R}^{L\times d}$, where d is the hidden dimension. Accordingly, the n-th layer hidden state sequence for segment $S_{\mathcal{T}+1}$ is computed schematically as follows,

$$\begin{cases} \widetilde{\boldsymbol{h}}_{\mathcal{T}+1}^{n-1} = [\mathrm{SG}(\boldsymbol{h}_{\mathcal{T}}^{n-1}) \circ h_{\mathcal{T}+1}^{n-1}], \\ \boldsymbol{q}_{\mathcal{T}+1}^{n}, \boldsymbol{k}_{\mathcal{T}+1}^{n}, \boldsymbol{v}_{\mathcal{T}+1}^{n} = h_{\mathcal{T}+1}^{n-1}\mathbf{W}_{\mathrm{q}}^{\top}, \tilde{h}_{\mathcal{T}+1}^{n-1}\mathbf{W}_{\mathrm{k}}^{\top}, \tilde{h}_{\mathcal{T}+1}^{n-1}\mathbf{W}_{\mathrm{v}}^{\top}, \\ \boldsymbol{h}_{\mathcal{T}+1}^{n} = \mathrm{Transformer-Layer}(\boldsymbol{q}_{\mathcal{T}+1}^{n}, \boldsymbol{k}_{\mathcal{T}+1}^{n}, \boldsymbol{v}_{\mathcal{T}+1}^{n}). \end{cases} \tag{57}$$

Here, $SG(\cdot)$ denotes the stop-gradient operation, the notation $[h_u \circ h_v]$ represents the concatenation of two hidden state sequences along the length dimension, and W refers to the learnable model parameters.

In contrast to the standard Transformer, the fundamental distinction is that the keys $k_{\mathcal{T}-1}^{n}$ and values $v_{\mathcal{T}-1}^{n}$ are conditioned on the extended context $\tilde{h}_{\mathcal{T}-1}^{n-1}$ and $h_{\mathcal{T}}^{n-1}$, which are cached from the preceding segment. This design choice is highlighted by the green paths in Fig. 45a. When applied across every pair of consecutive segments in a corpus, this mechanism effectively establishes a segment-level recurrence in the hidden states, allowing the effective context to extend well beyond the boundary of two segments alone.

It is worth noting, however, that the recurrent dependency between $h_{\mathcal{T}+1}^{n}$ and $h_{\mathcal{T}}^{n-1}$ shifts one layer downward per segment, distinguishing it from the same-layer recurrence found in conventional RNN-based language models. As a result, the maximum possible dependency length scales linearly with both the number of layers and the segment length, i.e., $O(N \times L)$, as illustrated by the shaded region in Fig. 45b.

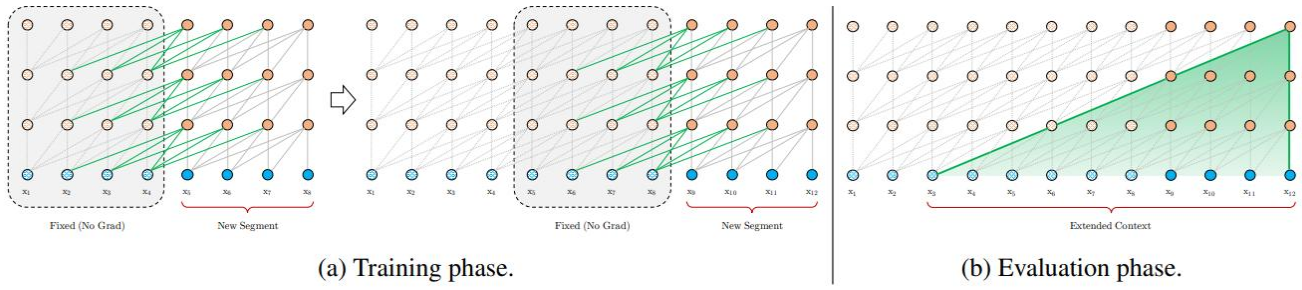

Fig. 45. Illustration of the Transformer-XL model with a segment length 4 [168].

Relative Positional Encodings: In the standard Transformer architecture, every position is assigned a unique absolute positional encoding (e.g., positions 1, 2, 3, …). However, when hidden states are reused from a prior segment, applying the same encoding scheme directly would result in both the cached and current tokens receiving identical absolute position indices, introducing a critical positional ambiguity. To circumvent this failure mode, the model instead encodes only relative positional information within the hidden states. Crucially, a query token only needs to know its relative distance to the keys it attends to, rather than their absolute positions within the sequence.

4.4. XLNet

XLNet [169] is a generalized autoregressive (AR) pretraining method that unifies the strengths of both autoencoding (AE) and autoregressive (AR) approaches through a novel permutation-based language modeling objective. Its architecture extends Transformer-XL by incorporating a two-stream attention mechanism, allowing it to effectively model bidirectional contexts while remaining compatible with the autoregressive training framework.

XLNet was developed to integrate the complementary strengths of two distinct modeling paradigms: the bidirectional contextual understanding of BERT (an autoencoding model) and the strong generative capabilities of Transformer-XL (an autoregressive model), while overcoming the core limitations inherent to each. Since both autoregressive (AR) language modeling and BERT offer distinct yet complementary advantages, a fundamental question arises as to whether a single pretraining objective can unify the benefits of both approaches without inheriting their respective weaknesses. XLNet addresses this through the permutation language modeling objective, which preserves the strengths of AR modeling while simultaneously enabling the model to capture bidirectional contextual dependencies. Crucially, XLNet does not modify the natural left-to-right ordering of the input sequence. Rather, it alters the factorization order of the probability computation during the training phase. Specifically, for any given input sequence, XLNet stochastically samples a permutation of the token order at each training step and computes the conditional probability under that permuted factorization.

In addition, XLNet introduces two separate sets of hidden representations in place of a single unified representation. The Content Stream operates as the standard self-attention mechanism, encoding both the token itself and its surrounding context; it is employed for all tokens except the one currently being predicted. The Query Stream, by contrast, is a newly introduced stream that has access only to the positional embedding of the current token and the contextual information derived from other tokens via the content stream. Importantly, it has no access to the content of the token it is predicting.

Additionally, two key techniques originally introduced in Transformer-XL, namely, the relative positional encoding scheme and the segment recurrence mechanism are incorporated into XLNet. Fig. 46 presents an overview of permutation language modeling in conjunction with the two-stream attention mechanism.

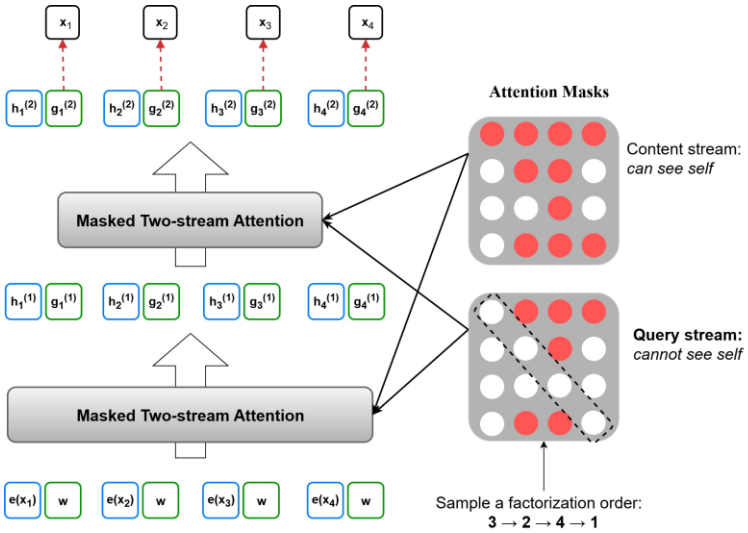

Fig. 46. An overview of the permutation language modeling objective during training, incorporating the two-stream attention mechanism

4.5. BART

Bidirectional and Auto-Regressive Transformers (BART) [170] is a denoising autoencoder designed for pretraining sequence-to-sequence models and can be applied to a wide variety of downstream tasks. It integrates two transformer models: BERT [145] as the bidirectional encoder and GPT [164], as the autoregressive decoder. The training process for BART involves (1) introducing noise into the text with an arbitrary corruption function and (2) training the model to reconstruct the original input. Its architecture follows the standard sequence-to-sequence Transformer design, employing a bidirectional encoder over the corrupted input and a left-to-right autoregressive decoder. A major strength of this approach is the flexibility of the noise function, allowing for diverse transformations of the original text, including modifications to its length.

Following GPT, In BART, ReLU activation functions are replaced to GeLUs [171] and parameters are initialized from N (0, 0.02). The base model consists of 6 encoder and 6 decoder layers, while the large model uses 12 layers in each. The architecture is closely aligned with BERT but with two key differences: (1) each decoder layer incorporates cross-attention over the encoder's final hidden layer, as in standard sequence-to-sequence transformers; and (2) BERT introduces an additional feed-forward network before word prediction, which BART omits. Overall, BART has roughly 10% more parameters compared to BERT model of equivalent size.

BART demonstrates particularly strong performance when fine-tuned for text generation but also performs competitively on comprehension-based tasks. On a variety of text production tasks, BART delivers new state-of-the-art outcomes while performing comparably to RoBERTa [150] on discriminative tasks.

4.6. Fast Transformer

The Fast Transformer [57] introduces multi-query attention (as discussed in Section 3.3.1) as an alternative to traditional multi-head attention, significantly reducing memory bandwidth usage and thereby improving processing efficiency.

4.7. T5

Text-to-Text Transfer Transformer (T5) [172] is a pre-trained language model built on the transformer architecture. It introduces a unified text-to-text framework, enabling the handling of any natural language processing (NLP) task by representing both inputs

and outputs as natural language sequences. The main concept behind T5 is to frame every text processing task as a "text-to-text" problem, accepting text as input and generating new text as output. This formulation makes it possible to apply the same model, training objective, optimization procedure, and decoding strategy uniformly across all tasks. T5 can be scaled flexibly by adjusting its parameter size, ranging from 60M up to 11B, which allows it to achieve strong performance across a wide set of NLP benchmarks. Additionally, it leverages a full-attention mechanism, which helps capture long-range dependencies and complex semantic relationships in text. T5 has been effectively utilized in multiple NLP applications, including machine translation, summarization, question answering, and sentiment analysis. The T5 model employs a standard encoder–decoder structure similar to the original Transformer, as shown in Fig. 47.

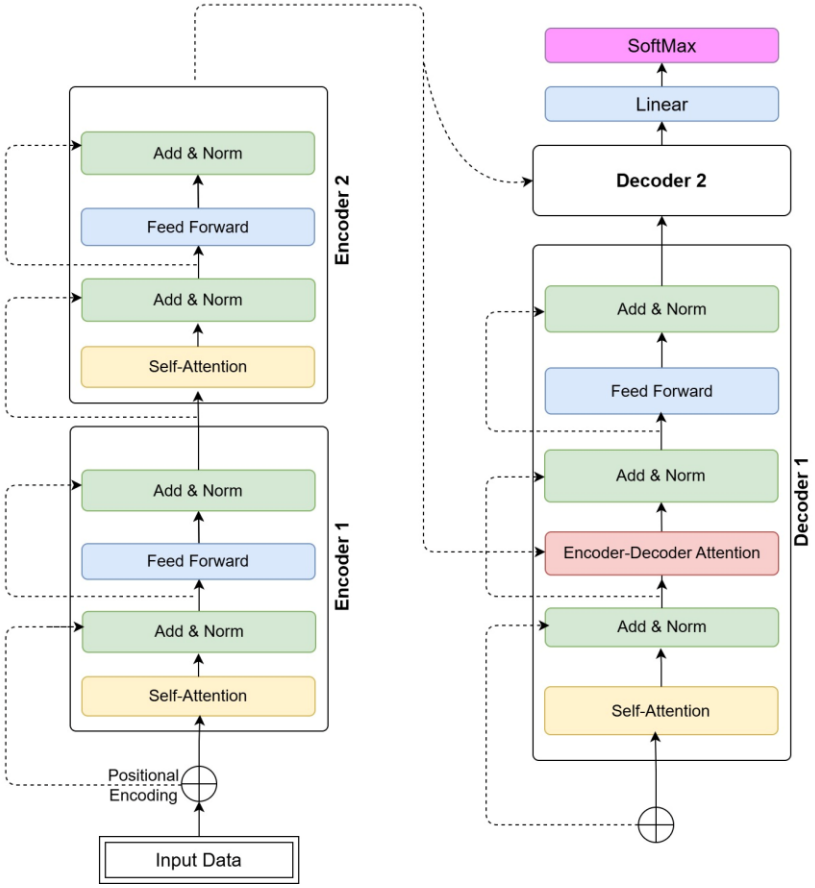

Fig. 47. Architecture of the T5 model.

Input tokens are first converted into embeddings and then processed by the encoder, which is organized into repeated blocks. Each block contains two core layers: a self-attention module followed by a small feed-forward network. Before each subcomponent, layer normalization [141] is applied, using a variant that rescales activations without introducing bias terms. After normalization, residual skip connections [140] combine each subcomponent's input with its output. Dropout [173] is also applied across feed-forward layers, skip connections, and attention modules at both the input and output of the encoder stack.

The decoder mirrors this design but introduces an additional attention mechanism after each self-attention layer to incorporate information from the encoder. Additionally, the decoder's self-attention mechanism employs a type of autoregressive or causal self-attention, which only permits the model to focus on previous outputs. The decoder's final block projects into a dense layer with a softmax activation, whose weights are tied to the input embedding matrix. All attention mechanisms are divided into multiple heads, with their results concatenated before further processing. T5 uses a simplified positional encoding, where each embedding corresponds to a scalar value added directly to the logits used in attention computation. For efficiency, positional embeddings are shared across all layers, though within each layer, individual heads maintain distinct embeddings.

A key aspect of T5's text-to-text framework is the use of task-specific prefixes, which convert every NLP task into a text generation problem. For instance, in sentiment analysis, the model prepends the prefix "sentiment:" to the input sentence and produces "positive" or "negative" as the output. This approach enables a single model to handle diverse tasks without requiring modifications to its architecture or training objective [174].

4.8. Longformer

Longformer [175] is a Transformer-based architecture designed to efficiently handle long documents, enabling a wide variety of document-level Natural Language Processing (NLP) tasks without the need for chunking inputs, shortening sequences, or relying on complex architectures to merge information across chunks. Its attention mechanism combines two components: a windowed local-context self-attention and a task-driven global attention that incorporates inductive bias from the specific task. The local attention is primarily responsible for capturing contextual representations, whereas the global attention enables Longformer to generate full-sequence representations required for prediction. In this model, the full self-attention matrix is sparsified through an "attention pattern" that specifies which input locations attend to each other. Unlike standard full self-attention, this attention pattern scales linearly with sequence length, making Longformer efficient for processing extended inputs.

Fig. 48 illustrates the comparison between the full self-attention matrix and the attention configurations used in Longformer. In the sliding-window approach, each token attends to a fixed number of neighboring tokens, emphasizing local context. When multiple layers of this windowed attention are stacked, the receptive field expands, allowing higher layers to incorporate information from the entire sequence. With a window size w, each token attends to $\frac{1}{2}w$ tokens on each side (Fig.48b). The computation complexity of this mechanism is $O(n \times w)$, which grows linearly with sequence length n. For a Transformer with L layers, the receptive field at the top layer reaches $L \times w$.

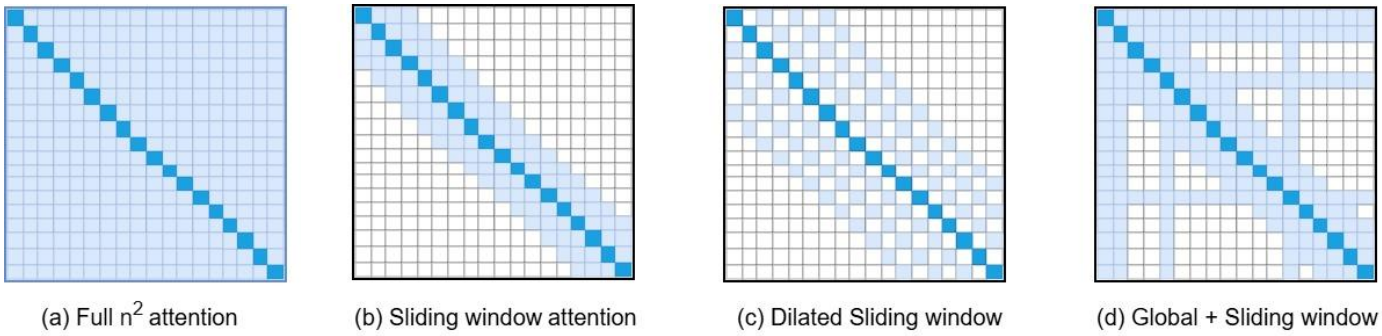

Fig. 48. Comparing the full self-attention pattern and the configuration of attention patterns in Longformer.

To expand the receptive field without adding computational cost, the sliding window can be "dilated," similar to the approach used in dilated CNNs [176], where the window includes gaps of size dilation d (Fig. 48c). In multi-head attention, each head calculates its own attention score, and using varied dilation settings across heads enhances performance. This setup allows some heads without dilation to capture local context, while others with dilation attend to longer-range context.

Windowed and dilated attention alone lack the flexibility to capture task-specific representations. To address this, "global attention" is introduced at a few pre-selected input positions. This operation is defined symmetrically: tokens with global attention attend to all tokens in the sequence, and all tokens attend back to them. Fig. 48d illustrates a sliding window attention combined with global attention applied at selected custom positions. Because the number of such tokens is small compared to n and does not scale with sequence length, the overall complexity of combining local and global attention remains $O(n)$. Although the choice of global attention locations is task dependent, it provides a straightforward way to incorporate inductive bias into the attention mechanism and is significantly simpler than prior task-specific methods that rely on complex architectures to merge information from smaller chunks of input.

4.9. BIGBIRD

BIGBIRD [177] employs a sparse attention mechanism that reduces the quadratic complexity of Transformers to linear, enabling improved performance on tasks requiring long contextual information. It serves as a universal approximator of sequence functions and is Turing complete, thereby retaining the expressive properties of the standard quadratic full-attention model. From a theoretical perspective, BIGBIRD leverages additional global tokens to maintain the model's expressive capacity. The approach is inspired by graph sparsification techniques and carefully examines where the proof of Transformer expressiveness fails when full attention is replaced with the proposed sparse attention pattern.

The BIGBIRD model employs a generalized attention mechanism, applied in each Transformer layer to process input sequences. This mechanism is described by a directed graph D where the vertex set is $[n] = \{1, \ldots, n\}$. The arcs (directed edges) specify the inner products that the attention mechanism will compute. For a node i in D, let $N(i)$ denote its out-neighbors; then the i-th output of the generalized attention mechanism is defined as

$$ATTN_D(X)_i = x_i + \sum_{h=1}^{H} \sigma\left(Q_h(x_i)K_h\left(X_{N(i)}\right)^T\right) \cdot V_h\left(X_{N(i)}\right) \quad (58)$$

Here, $Q_h, K_h : \mathbb{R}^d \to \mathbb{R}^m$ are the query and key functions, while $V_h : \mathbb{R}^d \to \mathbb{R}^d$ represents the value function. The symbol σ denotes a scoring function such as softmax or hardmax, and H indicates the number of attention heads. Additionally, $X_N(i)$ corresponds to the matrix created by stacking the vectors $\{x_j : j \in N(i)\}$, rather than all input tokens.

When D represents the complete digraph, the standard Transformer's full quadratic attention mechanism [20] is recovered. The challenge of reducing self-attention's quadratic complexity can therefore be formulated as a graph sparsification problem. It is well established that random graphs act as expanders and can approximate complete graphs in various ways, including through their spectral properties [178, 179]. For the sparse random graph used in the attention mechanism, two key requirements arise: a short average path length between nodes and a sense of locality. Based on this, sparse attention is introduced where each query attends to r randomly chosen keys (Fig. 49a).

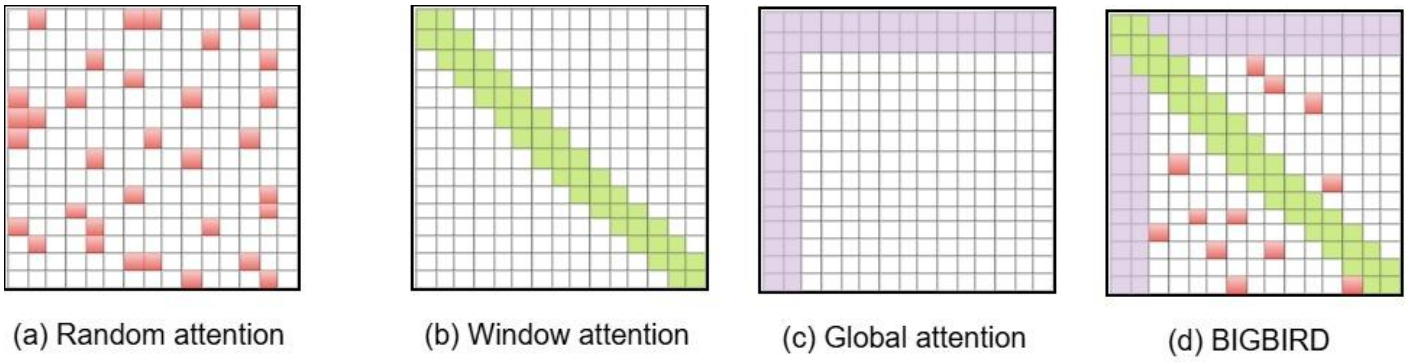

Fig. 49. Building blocks of the attention mechanism used in BIGBIRD.

The second motivation for BIGBIRD is that many NLP and computational-biology datasets exhibit strong locality of reference. In this phenomenon, much of a token's information can be inferred from its neighbors. To encode this locality, BIGBIRD uses a sliding-window attention pattern, for window width w, the query at position i attends only to keys in the range $i - w/2$ to $i + w/2$ (Fig. 49b). The design is complemented by a small set of global tokens that connect to every position to capture long-range dependencies (Fig. 49c).

In particular, BIGBIRD is composed of three essential components: (1) every token attends to a group of r randomly chosen tokens, (2) each token also attends to w local neighboring tokens, and (3) a set of g global tokens that interact with the entire sequence. The resulting attention mechanism for BIGBIRD (Fig. 49d) incorporates all three: each query attends to r random keys, $w/2$ neighboring tokens on either side of its position, and additionally includes g global tokens.

4.10. Performer

Performer [180] is a Transformer variant designed to approximate full-rank softmax attention with theoretical accuracy guarantees while reducing computational requirements from quadratic to linear in both time and memory. Unlike methods that depend on structural assumptions such as sparsity or low-rank constraints, this approach achieves efficiency without such restrictions.

To approximate softmax attention kernels, Performer introduces a novel method called Fast Attention Via Positive Orthogonal Random features (FAVOR+), which is also of independent interest for scalable kernel-based approaches. FAVOR+ can be applied to efficiently represent any kernelizable attention mechanism beyond softmax. This expressive capability is essential for enabling large-scale comparisons between softmax and alternative kernels, which standard Transformers cannot achieve, and for exploring optimal attention kernels. Performer is a linear model that remains fully compatible with standard Transformers while offering strong theoretical guarantees, including unbiased (or nearly unbiased) estimation of the attention matrix, uniform convergence, and low estimation variance.

4.11. Linformer

Large Transformer models have achieved remarkable success in delivering state-of-the-art performance across a wide range of natural language processing tasks. Nevertheless, their training and deployment become extremely expensive for long sequences,

since the Transformer's standard self-attention mechanism requires $O(n^2)$ time and memory relative to the sequence length.

Wang et al. [181] demonstrated that self-attention can be efficiently approximated through a low-rank matrix, enabling a reformulation of the mechanism that scales linearly rather than quadratically in both time and memory. Their approach partitions the standard dot-product attention into multiple smaller components via linear projections, which together yield a low-rank representation of the original attention process. Building on this idea, they introduced Linformer, a model that delivers accuracy comparable to conventional Transformers while achieving substantial gains in computational and memory efficiency.

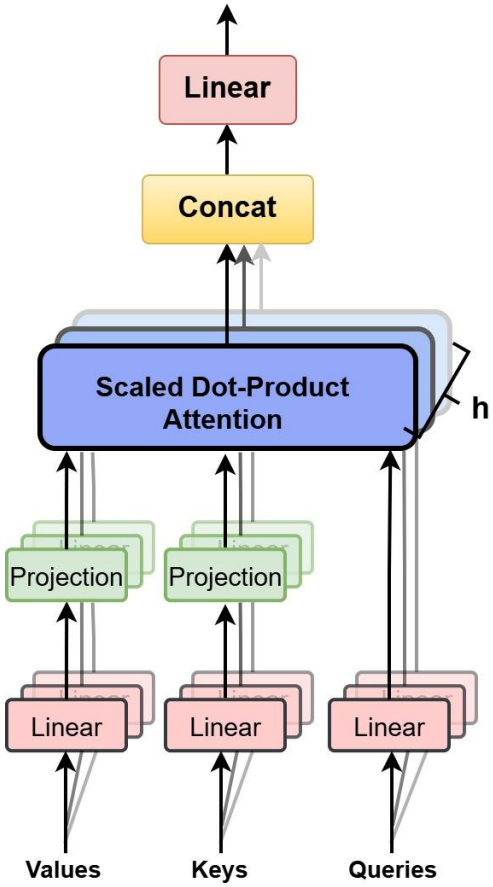

Fig. 50. Multi-Head Linear self-attention (Linformer).

In Linformer, a new self-attention mechanism is introduced that enables the contextual mapping $P \cdot VW_i^V$ to be computed with linear time and memory complexity relative to sequence length. The proposed linear self-attention, illustrated in Fig. 50, is based on the idea of incorporating two projection matrices $E_i, F_i \in \mathbb{R}^{n \times k}$ during the computation of keys and values. Specifically, the original key and value layers, KW_i^K and VW_i^V, which are of dimension $(n \times d)$ are projected into $(k \times d)$-dimensional representations using these matrices. Finally, the context embeddings for each $head_i$ are computed as $\bar{P} \cdot (F_i VW_i^V)$.

$$\overline{head_i} = Attention\left(QW_i^Q, E_i KW_i^K, F_i VW_i^V\right) = \underbrace{softmax\left(\frac{\left(QW_i^Q (E_i KW_i^K)^T\right)}{\sqrt{d_k}}\right)}_{\bar{P}:\, n \times k} \cdot \underbrace{(F_i VW_i^V)}_{k \times d} \tag{59}$$

These operations require only $O(nk)$ time and space complexity. Consequently, by selecting a relatively small projected dimension k, such that $k \ll n$, it becomes possible to greatly reduce both memory usage and computational cost.

4.12. Reformer

Although large Transformer models consistently achieve state-of-the-art performance across many tasks, training them is

computationally costly, particularly on long sequences. Reformer [182] introduces two techniques to improve the efficiency of Transformers. First, dot-product attention is replaced with an attention mechanism based on locality-sensitive hashing (LSH), which reduces the complexity from $O(L^2)$ to $O(L\ log\ L)$, where L is the sequence length. Second, reversible residual layers are used instead of standard residuals, allowing activations to be stored only once during training rather than N times, where N is the number of layers.

Reformer combines the modeling capacity of Transformer models with an architecture optimized for efficient execution on long sequences while keeping memory usage low, even in models with many layers. This makes large, high-capacity Transformer models more practical and accessible. Moreover, the ability to process long sequences allows Reformer to be applied to a variety of generative tasks. In addition to producing extended coherent text, it broadens the applicability of Transformer models to domains such as time-series forecasting, music composition, and image and video generation.

4.13. RoFormer

RoFormer [183] is an improved Transformer that introduces Rotary Position Embedding (RoPE) to more effectively capture positional information. Positional encoding in general has proven useful within Transformer architectures, as it provides essential supervision for modeling dependencies between elements at different sequence positions. RoPE is a novel technique that encodes absolute positions through a rotation matrix while simultaneously incorporating explicit relative positional dependencies into the self-attention mechanism. Theoretical analysis shows that relative positions can be naturally expressed via vector operations in self-attention, while absolute positional information is embedded through rotation. Importantly, RoPE offers several advantages, such as flexibility in sequence length, a natural decay of inter-token dependencies as relative distance increases, and the ability to integrate relative positional encoding into linear self-attention.

RoPE incorporates Euler's formula multiplication ($e^{i\theta}$) into query and key vectors as a relative positional embedding [184]. Specifically, for the $n-th$ query and $m-th$ query, where $\boldsymbol{q}_n, \boldsymbol{k}_m \in \mathbb{R}^{1\times d_{head}}$, RoPE is applied as

$$\begin{cases} q'_n = q_n e^{i\,n\,\theta} \\ k'_m = k_m e^{i\,m\,\theta} \end{cases} \tag{60}$$

The (n,m) −th entry of the attention matrix is then computed as

$$A'_{(n,m)} = Re[q'_n k'^{*}_m] = Re\left[q_n k^{*}_m e^{i\,(n-m)\theta}\right] \tag{61}$$

where Re $[\cdot]$ denotes the real part of a complex number and $*$ indicates the complex conjugate. By applying these rotations $e^{i\theta n}$ and $e^{i\theta m}$, RoPE incorporates relative positions $(n-m)$, which depend directly on the token pair (n,m), into the attention matrix in a rotational form. In practice, RoPE transforms $\boldsymbol{q}_n, \boldsymbol{k}_m \in \mathbb{R}^{1\times d_{head}}$ into complex vectors $\bar{\boldsymbol{q}}_n, \bar{\boldsymbol{k}}_m \in \mathbb{C}^{1\times (d_{head}/2)}$. This is done by treating the $(2t)$-th dimension as the real component and the $(2t+1)$-th dimension as the imaginary component. With this formulation, RoPE yields the same attention value as $\boldsymbol{q}_n \boldsymbol{k}_m^T = Re\left[\bar{\boldsymbol{q}}_n \bar{\boldsymbol{k}}_m^{*}\right]$ but with reduced computational overhead. Furthermore, RoPE employs multiple frequencies θ_t distributed across the channel dimensions of both query and key vectors.

$$\theta_t = 10000^{-t/(d_{head}/2)}, where\ t \in \{0, 1, \dots, d_{head}/2\} \tag{62}$$

In conclusion, the rotation matrix $R \in \mathbb{C}^{N\times (d_{head}/2)}$ is expressed as

$$R(n,t) = e^{i\,\theta_t n} \tag{63}$$

and applied to the query and key using the Hadamard product $\circ$ as

$$\begin{cases} \bar{q}' = \bar{q} \circ R \\ \bar{K}' = \bar{K} \circ R \\ A' = Re\left[\bar{q}'\bar{k}'\,*\right] \end{cases} \tag{64}$$

It should be noted that with RoPE, attention matrix A′ represents relative positions in the rotational form $e^{i(n-m)\theta_t}$ across $d_{head}/2$ frequencies, this provides significant performance benefits to the Transformer, particularly for extrapolation during inference when dealing with periodic functions.

4.14. ALiBi

A key question is how models can extrapolate at inference time to handle sequences longer than those observed during training. [185] showed that the sinusoidal (absolute) position embedding approach prevents Transformers from extrapolating to inputs beyond the training length. Their study revealed that extrapolation can be enabled by modifying the positional representation method, though most existing approaches remain inefficient for this purpose. Rotary Position Embedding (RoPE) [183], and Relative Position embedding [186] are two substitutes for the original sinusoidal position embedding approach that offer better extrapolation. The superior of these, the T5 bias, requires more memory and parameters and is significantly slower than the sinusoidal method.

To address these limitations, they proposed a simpler and more efficient method called Attention with Linear Biases (ALiBi). ALiBi serves as a straightforward replacement for prior positional methods and enables extrapolation. Instead of adding positional embeddings to word embeddings, ALiBi introduces a bias to the query-key attention scores, applying a penalty proportional to their distance. Beyond extrapolation, ALiBi achieves better perplexity than sinusoidal embeddings in smaller models (<1B parameters, trained on limited data), and comparable perplexity in large-scale billion-parameter models trained on extensive data. The method is easy to implement, does not add runtime overhead or extra parameters, and only requires a negligible increase in memory.

In the standard Transformer architecture, positional embeddings are combined with word embeddings at the input layer. For a subsequence of length L, the attention sublayer calculates the attention score for the ith query $\boldsymbol{q}_i \in \mathbb{R}^{1\times d}, (1 \le i \le L)$ in each head, using the first i keys $\boldsymbol{K} \in \mathbb{R}^{i\times d}$, with $\boldsymbol{d}$ denoting the head dimension:

$$Softmax\ (\boldsymbol{q}_i \boldsymbol{K}^{\boldsymbol{T}}) \tag{65}$$

The attention scores are then multiplied with the values to produce the output of the attention sublayer. In ALiBi, positional embeddings are never introduced at any stage of the network. The sole modification occurs after the query-key dot product, where a fixed, non-trainable bias is added:

$$Softmax\ (\boldsymbol{q}_i \boldsymbol{K}^{\boldsymbol{T}} + m \cdot [-(i-1), \dots, -2, -1, 0]) \tag{66}$$

Here, the scalar m represents a head-specific slope that is predetermined before training begins.

Fig. 51 illustrates the linearly biased attention mechanism. For each head, ALiBi introduces a constant bias (shown on the right of figure) to every attention score $\boldsymbol{q}_i \cdot \boldsymbol{K}_j$ (shown on the left of figure). As in the standard attention sublayer, the softmax function is applied to these scores, while the remaining computation remains unchanged. The scalar m is a head-specific constant that is fixed and not updated during training.

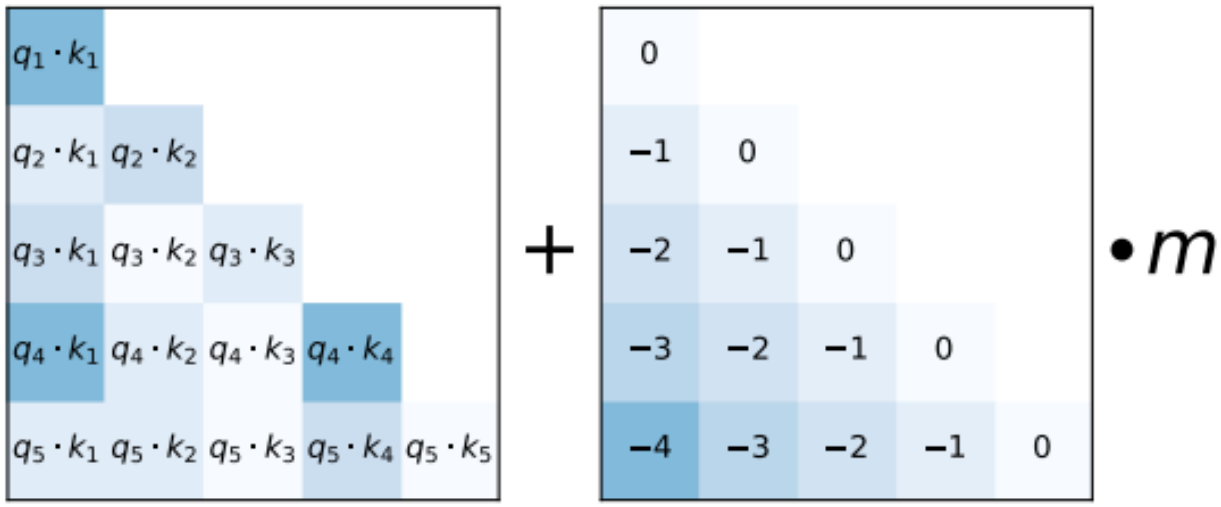

Fig. 51. The linearly biased attention method [185].

4.15. Switch Transformers

Switch Transformer [187] is proposed as a simple yet computationally efficient approach to scaling up the number of parameters in Transformer models. Switch Transformers represent a scalable and efficient approach for natural language learning. In this architecture, the conventional dense feed-forward network (FFN) layer of the Transformer is replaced by a sparse Switch FFN layer. In standard deep learning models, the same set of parameters is typically applied across all inputs. In contrast, Mixture of Experts (MoE) models [188] challenge this paradigm by dynamically selecting different parameter subsets for each individual input. This leads to a sparsely activated model architecture with an extremely large number of parameters, yet with a fixed computational cost per input. Despite the success of MoE models in several applications, their broader adoption has been limited due to challenges such as architectural complexity, high communication overhead, and instability during training. To overcome these issues, the Switch Transformer was introduced as a simplified version of the Sparsely-Gated Mixture of Experts (MoE) [189] It offers a design that is more interpretable, easier to train, and significantly more sample-efficient compared to dense models of equivalent size.

4.16. LLaMA

LLaMA [190] is a transformer-based language model that integrates a series of enhancements later proposed and adopted in various other architectures.
Inspired by GPT3 [166], LLaMA improves training stability by normalizing the input of each transformer sub-layer rather than its output, using the RMSNorm [191] normalization function. Following PaLM [192], it replaces the ReLU activation with the SwiGLU [193] activation function to enhance performance. Furthermore, drawing from GPTNeo [194] LLaMA eliminates absolute positional embeddings and instead applies rotary positional embeddings (RoPE) [183] at every layer of the network.

Several additional optimizations are introduced to accelerate training. First, an efficient implementation of the causal multi-head attention operator, inspired by [133, 195] is used to reduce memory usage and computation. Second, the amount of activation recomputation during the backward pass with checkpointing is minimized to further improve training efficiency. Finally, overlapping activation computation with inter-GPU communication over the network helps maximize throughput.

4.17. Vision Transformer (ViT)

The Vision Transformer (ViT) [22] is a novel architecture that adapts the Transformer framework for computer vision tasks such as image classification. Fig. 52 illustrates an overview of ViT. In this model, an input image is divided into fixed-size patches. While standard Transformers operate on 1D sequences of token embeddings, ViT processes 2D images by reshaping the input image $X \in \mathbb{R}^{H\times W\times C}$ into a sequence of flattened 2D patches $X_p \in \mathbb{R}^{N\times(P^2\cdot C)}$, Here, (H,W) denotes the original image resolution, C is the number of channels, (P,P) is the spatial resolution of each patch, and $N = HW/P^2$ is the total number of patches, which also defines the input sequence length for the Transformer. Each patch is flattened and projected into a fixed-dimensional latent space of size D using a trainable linear projection. The same latent dimension D is maintained throughout all Transformer layers.

$$\begin{cases} z^0 = \left[x_{class}; x_p^1 E; x_p^2 E; \cdots; x_p^N E\right] + E_{pos} \\ E \in \mathbb{R}^{(P^2\cdot C)\times D} \\ E_{pos} \in \mathbb{R}^{(N+1)\times D} \end{cases} \tag{67}$$

The patch embeddings are the result of this projection. Like BERT's [class] token, the series of embedded patches $(z_o^o = X_{class})$, is prepended with a learnable embedding, whose state at the Transformer encoder's output z_L^o represents the picture representation y (Eq. 22).

$$y = LN(z_L^0) \tag{68}$$

During pre-training and fine-tuning, z_L^o. is connected to a classification head during pre-training and fine-tuning. At pre-training and fine-tuning times, an MLP with a single hidden layer and a single linear layer, respectively, implement the classification head. The patch embeddings are supplemented with position embeddings in order to preserve positional information. They also make use of the common learnable 1D position embeddings. The encoder receives the sequence of embedding vectors that are produced as input. MLP blocks and multi-headed self-attention are arranged in alternating layers to construct the Transformer encoder [20]. Before each block, the layer norm (LN) is applied, and after each block, residual connections are applied [196].

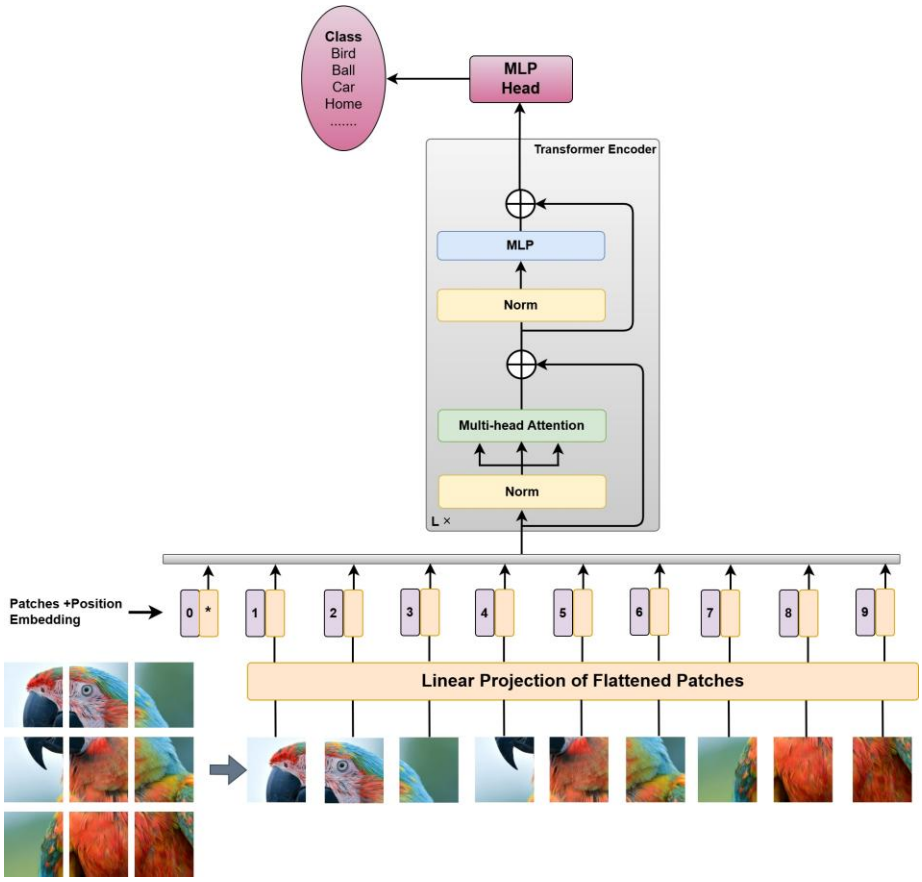

Fig. 52. Vision Transformer (ViT) overview.

4.18. Detection Transformer (DETR)

Detection Transformer (DETR) [197] introduces a novel approach to object detection by leveraging transformers along with a bipartite matching loss to enable direct set prediction. The architecture of DETR is composed of three main components: a backbone, a transformer module, and prediction heads. The overall structure of DETR is notably straightforward and is illustrated in Fig. 53.

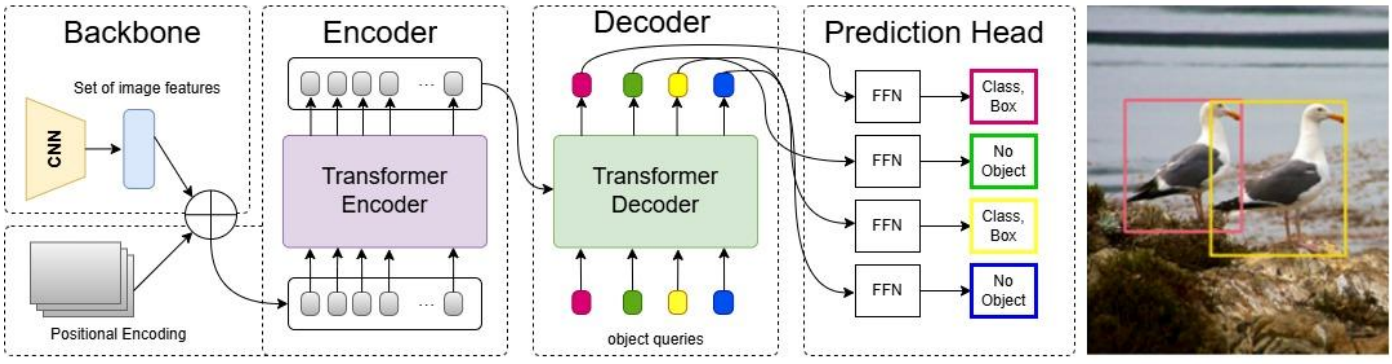

Fig. 53. Architecture of the Detection Transformer (DETR) model.

DETR employs a standard CNN backbone to extract a 2D feature representation from the input image. This feature map is then flattened and combined with positional encodings before being fed into the transformer encoder. The transformer decoder receives a fixed, small set of learned positional embeddings, referred to as object queries, and attends to the encoded features. Each output embedded produced by the decoder is subsequently passed through a shared feed-forward network (FFN), which is responsible for classifying the objects and predicting the corresponding bounding boxes.

4.19. DeepViT

Zhou et al. [198] showed that, unlike convolutional neural networks (CNNs), which often improve in performance when made deeper, Vision Transformers (ViTs) quickly reach a saturation point as depth increases. As the transformer becomes deeper, the attention maps gradually converge and eventually become almost identical across layers. In other words, the feature maps in higher layers of deep ViT models tend to be nearly the same. This phenomenon, termed attention collapse, demonstrates that in deeper layers the self-attention mechanism fails to capture effective representations, thereby limiting the model's potential performance gains.

To address the issue of attention collapse and enable deeper scaling of Vision Transformers, the authors propose a simple, yet effective self-attention mechanism called Re-attention. This mechanism leverages the multi-head self-attention (MHSA) framework and regenerates attention maps through a learnable exchange of information across different attention heads.

Specifically, they generate a new set of attention maps by dynamically aggregating the attention maps from different heads, using them as the basis. To achieve this, a learnable transformation matrix $\Theta \in \mathbb{R}^{H\times H}$ is defined and used to mix the multi-head attention maps into regenerated ones, which are then multiplied with V. The Re-attention mechanism is formally expressed as:

$$Re - Attention(Q,K,V) = Norm\left(\Theta^T\left(Softmax\left(\frac{QK^T}{\sqrt{d}}\right)\right)\right)V \tag{69}$$

where the transformation matrix Θ is multiplied with self-attention map A along the head dimension. Norm is a normalization function which is applied to reduce layer-wise variance. Θ is end-to-end learnable.

4.20. Data-Efficient Image Transformers (DeiT)

One of the main limitations of the Vision Transformer (ViT) is its high computational cost during training to achieve strong generalization performance. To address this issue, DeiT [23] was proposed as a more efficient alternative that does not require large-scale datasets or extensive computational resources to attain competitive classification results. DeiT introduces a novel distillation strategy based on a distillation token, which functions similarly to the class token but is designed to mimic the label predicted by a teacher model. Both the class and distillation tokens participate in the attention mechanism within the transformer, allowing effective information transfer. This transformer-specific distillation approach significantly outperforms traditional distillation methods.

4.21. T2T-ViT

The Tokens-to-Token Vision Transformer (T2T-ViT) [199] is a vision Transformer architecture designed to overcome the limitations of the original Vision Transformer (ViT) by more effectively modeling image structural information and enhancing feature representation.

While ViT tends to underperform compared to convolutional neural networks (CNNs) when trained from scratch on mid-sized datasets such as ImageNet [200], this performance gap is attributed to two main issues: 1) Its simplistic tokenization approach fails to capture important local structures such as edges and textures formed by adjacent pixels, resulting in low training sample efficiency. 2) Its attention backbone contains redundancy, limiting feature expressiveness under constrained computational budgets and limited data.

To address these challenges, T2T-ViT introduces two core innovations: 1) A layer-wise Tokens-to-Token (T2T) transformation, which progressively restructures images into tokens by recursively aggregating neighboring tokens. This enhances the model's ability to capture local patterns while reducing token sequence length. 2) An efficient backbone architecture with a deep and narrow design, inspired by empirical insights from CNNs, which improves model capacity while maintaining computational efficiency.

Notably, T2T-ViT reduces both the parameter count and multiply-accumulate operations (MACs) of standard ViT by 50%, while achieving over 3.0% higher accuracy when trained from scratch on ImageNet. It also outperforms ResNet models [111] and achieves performance comparable to MobileNet architectures [201] under the same training settings.

4.22. CrossVit

The Cross-Attention Multi-Scale Vision Transformer (CrossViT) [202] is a dual-branch vision transformer designed to learn multi-scale features, proposed to improve image classification accuracy. CrossViT employs a simple, yet effective token fusion scheme based on cross-attention, which integrates features across different scales with linear complexity in both computation and memory.

Fig. 54 illustrates the architecture of the CrossViT model, which consists primarily of K multi-scale transformer encoders. Each encoder has two branches: (1) the **L-Branch**, a large primary branch that processes coarse-grained patches of size Pl, with more

transformer encoder layers and larger embedding dimensions; and (2) the **S-Branch**, a smaller complementary branch that handles fine-grained patches of size Ps, with fewer encoders and smaller embedding dimensions. The outputs of these two branches are fused L times throughout the network, and the final CLS tokens from both branches are used for prediction. as in ViT [22] , a learnable positional embedding is added to each token in both branches before entering the multi-scale transformer encoder to encode positional information.

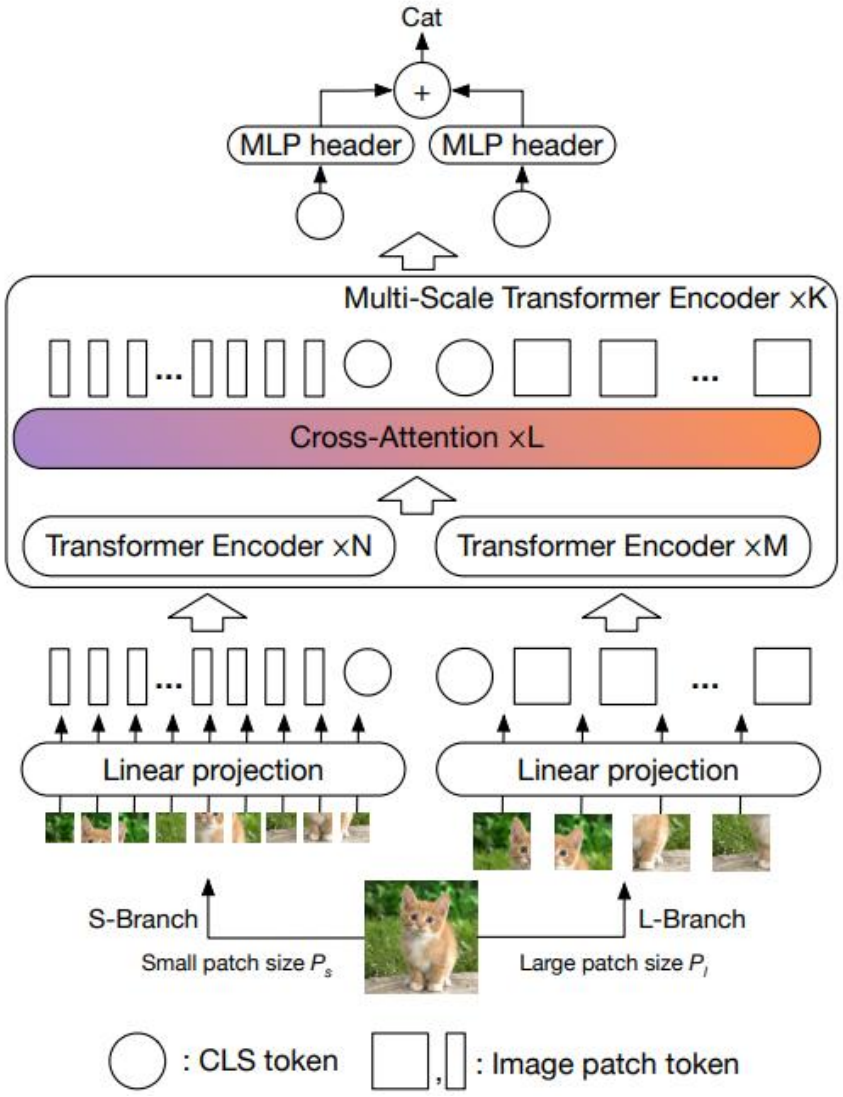

Fig. 54. The illustration of Cross-Attention Multi-Scale Vision Transformer (CrossViT) [202].

To efficiently integrate image patch tokens of different scales, a fusion mechanism based on cross-attention was developed to enable information exchange between the two branches in linear time. In this approach, each branch uses its CLS token as an intermediary to interact with the patch tokens of the other branch and then projects the exchanged information back to its own branch. Because the CLS token already summarizes high-level information from all patch tokens within its branch, interacting with tokens from the other branch introduces complementary scale information. After this fusion, the CLS token again interacts with its original patch tokens in the next transformer encoder layer, propagating the cross-branch information and enhancing the representation of each patch token.

Fig. 55 illustrates the cross-attention module for the large branch; it collects patch tokens from the S-branch and concatenates its CLS token to them. The same procedure is applied to the S-branch by swapping the indices l and s.

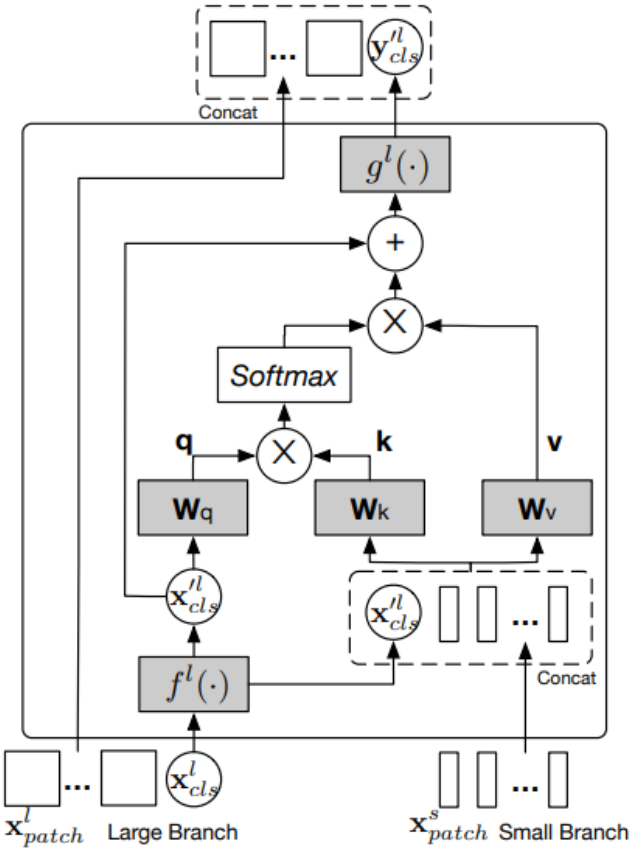

Fig. 55. Cross-attention module of CrossViT [202].

4.23. Pyramid Vision Transformer (PVT)

Pyramid Vision Transformer (PVT) [203] is a Transformer-based architecture tailored for advanced visual prediction tasks such as semantic segmentation and object recognition. A high-level illustration of the PVT framework is presented in Fig. 56.

Four steps in the process produce feature maps with varying scales. The architecture of each stage is the same and includes L_i Transformer encoder layers and a patch embedding layer. The initial step is splitting an input image of size $H \times W \times 3$, into $\frac{HW}{4^2}$ patches, each of which is $4 \times 4 \times 3$. After that, a linear projection is applied to the flattened patches, resulting in embedded patches with a size of $\frac{HW}{4^2} \times C_1$. A Transformer encoder with L_i is then used to process the embedded patches and a position embedding. The result is then molded into a feature map F_1 with a size of $\frac{H}{4} \times \frac{W}{4} \times C_1$Similarly, the following feature maps are produced utilizing the input feature map from the previous stage: F_2, F_3, and F_4, whose distances from the input image are 8, 16, and 32 pixels, respectively. The feature pyramid $\{F_1, F_2, F_3, F_4\}$, makes it simple to use PVT for the majority of downstream tasks, such as semantic segmentation, object detection, and picture classification.

The Pyramid Vision Transformer (PVT) offers several notable advantages for computer vision applications: (1) Unlike Vision Transformers (ViT), which often produce low-resolution outputs and demand high computational and memory resources, PVT supports training on densely partitioned images, enabling high-resolution outputs essential for dense prediction tasks. Additionally, it employs a progressive shrinking pyramid structure to reduce the computational burden associated with large feature maps. (2) PVT combines the strengths of both Convolutional Neural Networks (CNNs) and Transformers, serving as a unified, convolution-free backbone that can seamlessly replace traditional CNN architecture across a variety of vision tasks. (3) PVT significantly enhances the performance of multiple downstream tasks, including object detection, instance segmentation, and semantic segmentation.

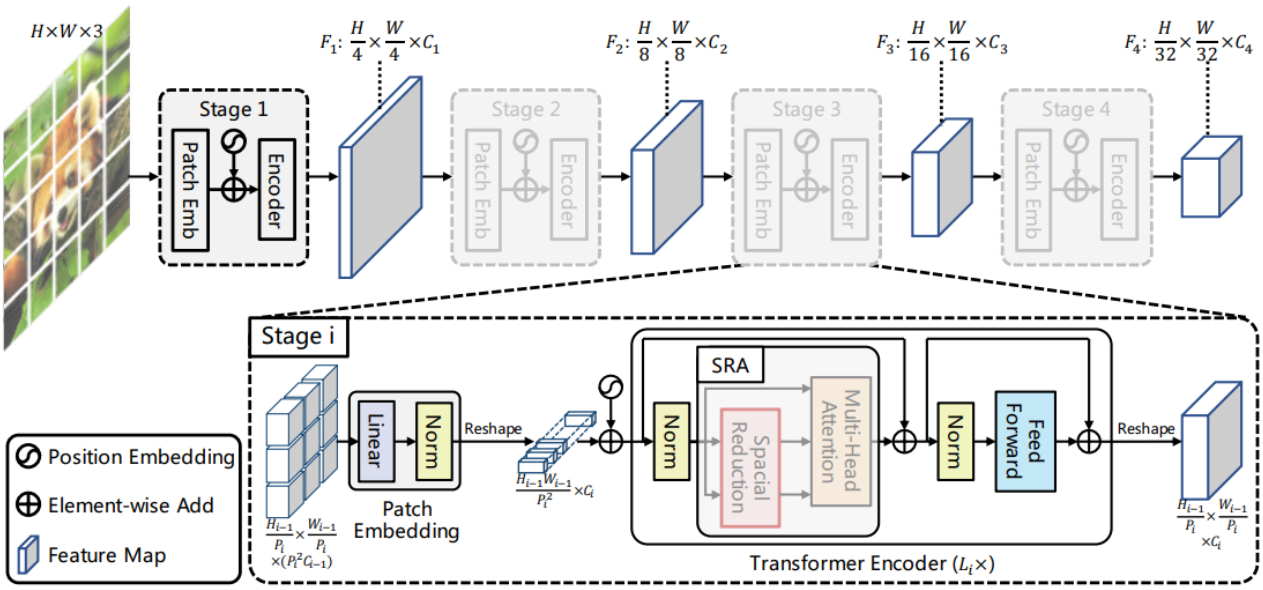

Fig. 56. Overall architecture of Pyramid Vision Transformer (PVT) [203].

4.24. Swin Transformer

The Swin Transformer [204] is a hierarchical Transformer model that constructs visual representations using a shifted window mechanism. This approach improves computational efficiency by restricting self-attention operations to non-overlapping local windows while still enabling cross-window connections to capture broader context. Thanks to its hierarchical structure, the Swin Transformer is capable of modeling visual features at multiple scales and maintains linear computational complexity with respect to image size. This makes it suitable for a wide range of computer vision tasks, including semantic segmentation, object detection, and image classification.

An overview of the Swin Transformer architecture is shown in Fig. 57. The model first divides an input RGB image into non-overlapping patches using a patch splitting module, similar to the strategy used in ViT. Each patch is interpreted as a "token," and its feature representation is initialized as a concatenation of the raw RGB pixel values. In the implementation described in [204], a patch size of 4 × 4 is used, resulting in a feature dimension of 4 × 4 × 3 = 48. This raw feature vector is then passed through a linear embedding layer, which projects it into a target embedding space of arbitrary dimensionality, denoted as C.

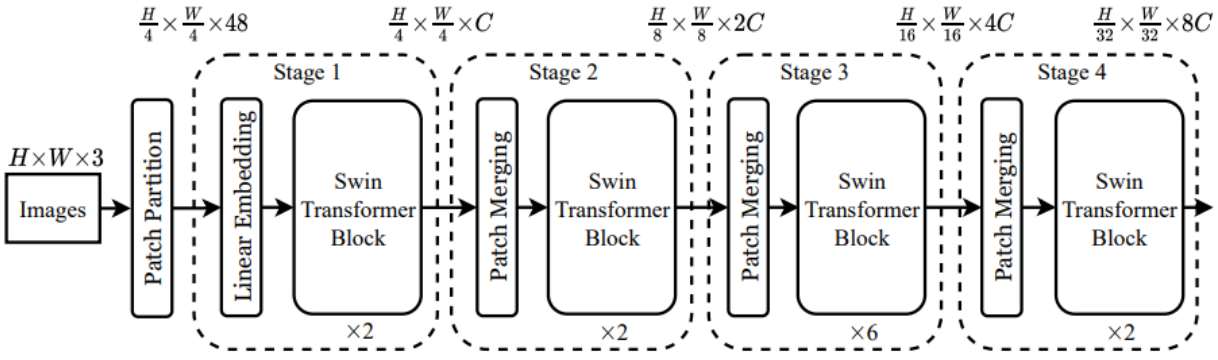

Fig. 57. The architecture of a Swin Transformer [204].

On these patch tokens, a number of Transformer blocks with modified self-attention computation (Swin Transformer blocks) are used. "Stage 1" refers to the Transformer blocks and the linear embedding, which preserve the amount of tokens ($\frac{H}{4} \times \frac{W}{4}$ As the network becomes deeper, patch merging layers minimize the number of tokens to create a hierarchical representation. The first patch merging layer applies a linear layer to the 4C-dimensional concatenated features after concatenating the features of each group of two-by-two neighboring patches. The output dimension is set to 2C, and the number of tokens is decreased by a multiple of 2×2 = 4

(2× downsampling of resolution). After feature transformation, Swin Transformer blocks are applied, maintaining the resolution at $\frac{H}{8} \times \frac{W}{8}$. "Stage 2" refers to this initial patch merging and feature transformation block. With output resolutions of $\frac{H}{16} \times \frac{W}{16}$ and $\frac{H}{32} \times \frac{W}{32}$, respectively, the process is performed twice as "Stage 3" and "Stage 4." Together, these steps yield a hierarchical representation with feature map resolutions comparable to those of common convolutional networks, such as VGG [205] and ResNet [111].

Swin Transformer Block: This block is constructed by replacing the standard multi-head self-attention (MSA) module within a typical Transformer block with a shifted window-based attention module, while retaining the other components unchanged. As depicted in Fig. 58, a Swin Transformer block comprises a shifted window-based MSA (Multi-head Self Attention) module, followed by a two-layer MLP with a GELU activation function applied between the layers. Additionally, Layer Normalization (LN) is performed before both the MSA and MLP modules, and residual connections are applied after each of these modules. Thanks to the shifted window partitioning strategy, consecutive Swin Transformer blocks are computed as follows:

$$\hat{z}^{l} = W - MSA\left(LN(z^{l-1})\right) + z^{l-1} \tag{70}$$

$$z^{l} = MLP\left(LN(\hat{z}^{l})\right) + \hat{z}^{l} \tag{71}$$

$$\hat{z}^{l+1} = SW - MSA\left(LN(z^{l})\right) + z^{l} \tag{72}$$

$$z^{l+1} = MLP\left(LN(\hat{z}^{l+1})\right) + \hat{z}^{l+1} \tag{73}$$

W-MSA and SW-MSA indicate window-based multi-head self-attention utilizing normal and shifted window partitioning configurations, respectively, whereas $\hat{z}^{l}$ and z^{l} l indicate the output features of the (S)W-MSA module and the MLP module for block l, respectively.

The shifted window partitioning strategy establishes connections between adjacent non-overlapping windows from the preceding layer and has proven to be effective across various tasks such as image classification, object detection, and semantic segmentation.

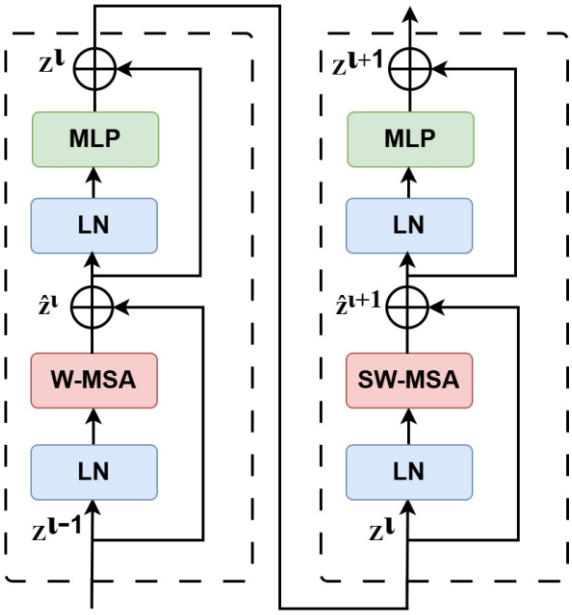

Fig. 58. Two successive Swin Transformer Blocks.

4.25. TNT

The Transformer in Transformer (TNT) model [206] is a vision Transformer designed for visual recognition tasks. It captures both local and global representations by employing a dual-transformer structure, consisting of an inner transformer for modeling fine-grained local features and an outer transformer for capturing broader contextual information.

The Transformer-in-Transformer (TNT) architecture for visual recognition, as illustrated in Fig. 59, enhances the representational capacity of visual transformers by introducing a hierarchical structure.

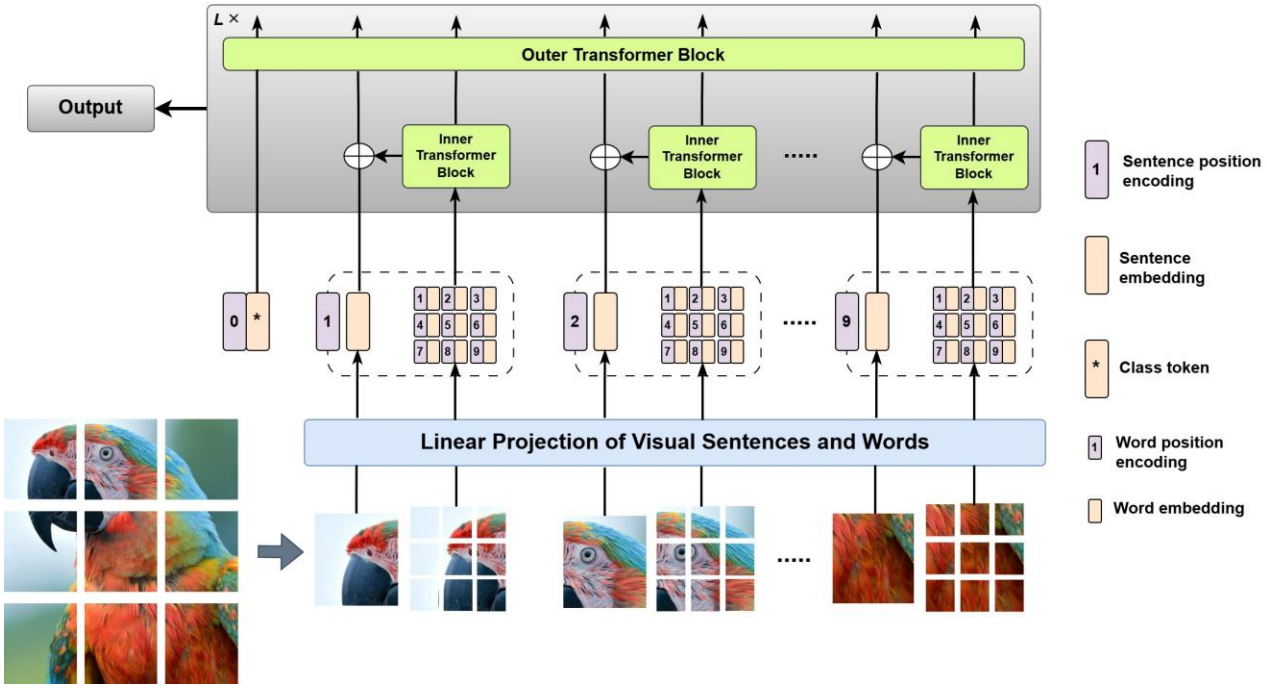

Fig. 59. Transformer-in-Transformer (TNT) framework.

Initially, input images are divided into patches referred to as "visual sentences," which are further partitioned into smaller sub-patches termed "visual words." This design allows the model to capture more detailed visual features. In addition to the standard transformer blocks that operate on visual sentences to extract high-level features and attention maps, the TNT model embeds an additional sub-transformer to focus specifically on the features and interactions among visual words within each sentence. These word-level attentions and features are computed independently using a shared sub-network, ensuring that the increase in parameters and floating-point operations (FLOPs) remains minimal. The resulting word-level features are then aggregated into their corresponding visual sentence representation. A class token is included in the architecture and passed through a fully connected head for the final visual recognition task. The TNT framework enables the extraction of visual information at a finer granularity and provides features with more details.

PyramidTNT [207] is an enhanced version of the original TNT model, incorporating a pyramidal architecture along with a convolutional stem to significantly improve its performance and representational capacity.

4.26. Multiscale Vision Transformers (MViT)

Multiscale Vision Transformers (MViT) [208] is a transformer-based architecture developed for visual data such as images and videos. Its main goal is to integrate the concept of multiscale feature hierarchies into the transformer framework. The model progressively increases feature complexity while reducing visual resolution in a hierarchical fashion.

In contrast to conventional transformers, which preserve a fixed channel capacity and spatial resolution across all layers, Multiscale Transformers employ multiple channel-resolution "scale" stages. Starting from the original image resolution with a relatively small channel dimension, these stages progressively increase channel capacity while reducing spatial resolution. This process creates a multiscale pyramid of feature activations within the transformer, effectively bridging transformers with the concept of multiscale feature hierarchies.

Fig. 60 illustrates the architecture of Multiscale Vision Transformers (MViT). In the initial layers, the model operates at high spatial resolution to capture simple low level visual details, thanks to its lightweight channel capacity. In contrast, the deeper layers focus on spatially coarser yet more complex features to capture high level visual semantics. The primary advantage of the multiscale transformer lies in its ability to leverage the dense nature of visual signals, a property that is even more pronounced in space time visual data such as videos.

A noteworthy benefit of MViT design is the presence of strong implicit temporal bias. Vision transformer models trained on natural videos suffer no performance decay when tested on videos with shuffled frames. This indicates that these models are not effectively using the temporal information and instead rely heavily on appearance. In contrast, when testing the MViT model on shuffled frames, significant accuracy decay is observed, suggesting reliance on temporal information.

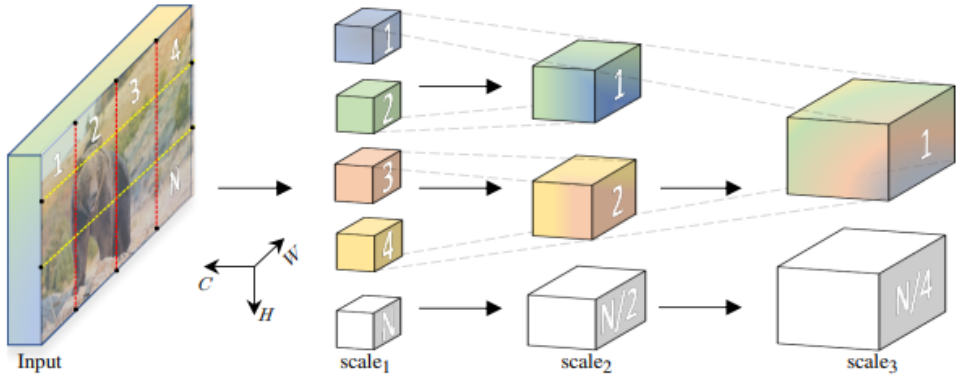

Fig. 60. The architecture of Multiscale Vision Transformers (MViT) [208].

4.27. ViViT

The Video Vision Transformer (ViViT) [209] is a transformer-based architecture for video classification, inspired by the success of transformers in image classification.

The central operation of this model is self-attention, applied to a sequence of spatio-temporal tokens extracted from the input video. To handle the large number of tokens in video data, the model employs factorization strategies along the spatial and temporal dimensions, improving both efficiency and scalability. To further support training on smaller datasets, the model incorporates regularization during training and leverages pretrained image models. Fig. 61 illustrates the ViViT architecture for video classification, while the right side of the figure depicts different factorization strategies corresponding to spatial and temporal attention patterns.

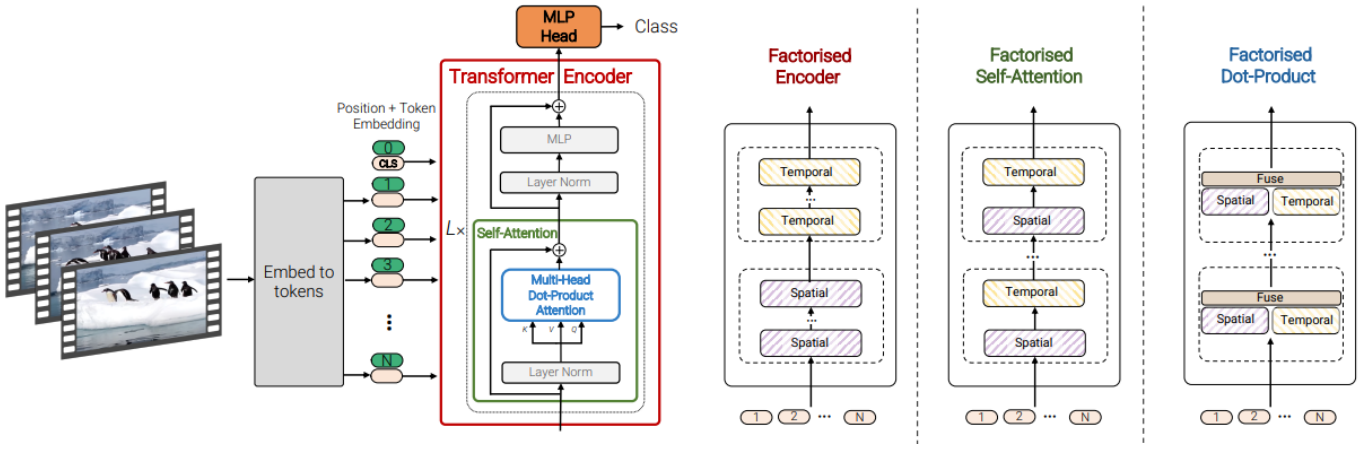

Fig. 61. The architecture of ViViT model [209].

As shown in fig. 53 To effectively process a large number of spatio-temporal tokens, the four model variants are developed which factorise different components of the transformer encoder over the spatial- and temporal-dimensions. These factorisations correspond to different attention patterns over space and time.

Model 1 (Spatio-temporal attention): In this model, all spatio-temporal tokens extracted from the video are passed directly through the transformer encoder. Tokens from all spatial and temporal positions are concatenated into an extended sequence and processed by a standard Transformer. Each transformer layer models every pairwise interaction among all spatio-temporal tokens and consequently captures long-range interactions across the video beginning with the first layer. However, due to this exhaustive pairwise modeling, Multi-Headed Self Attention incurs quadratic computational complexity in relation to the number of tokens. This complexity is especially relevant for video data, where the token count increases linearly with the number of input frames.

Model 2 (Factorised encoder): This architecture comprises two distinct transformer encoders, as depicted in Fig. 62. The first, a spatial encoder, restricts its interactions to tokens sharing the same temporal index. Following L_s layers, a representation $h_i \in \mathbb{R}^d$ is produced for each temporal index. This representation is derived either from an encoded classification token (if prepended to

the input) or from global average pooling applied to the spatial encoder's output tokens. The resulting frame-level representations h_i are concatenated into a matrix $H \in \mathbb{R}^{n_t \times d}$, which is subsequently processed by a temporal encoder comprising L_t transformer layers. This temporal encoder models interactions between tokens originating from different temporal indices, and its output is then classified. Although this model incorporates more transformer layers (and thus more parameters) than Model 1, it demands fewer floating point operations (FLOPs). This reduction in complexity arises because the two separate transformer blocks yield a complexity of $O((n_h \cdot n_w)^2 + n_t^2)$, in contrast to Model 1's $O((n_t \cdot n_h \cdot n_w)^2)$.

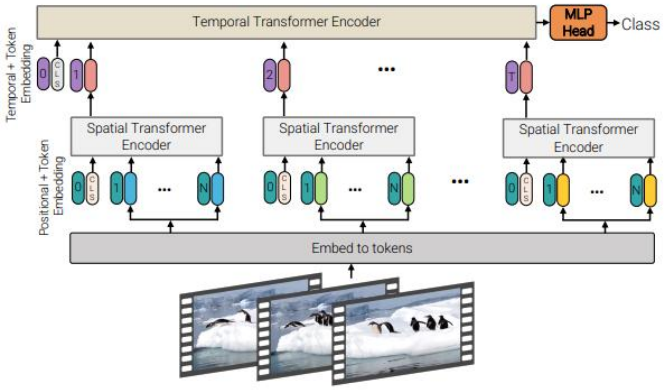

Fig. 62. An overview of the factorised encoder [209].

Model 3 (Factorised self-attention): This model has the same number of transformer layers as Model 1, but factorises the multi-headed self-attention operation rather than computing it across all token pairs z^l at layer l. As shown in Fig. 63, it first applies self-attention spatially (across tokens sharing the same temporal index), then temporally (across tokens sharing the same spatial index). Each self-attention block thus models spatio-temporal interactions more efficiently than Model 1 by factorising the operation over two smaller sets of elements. As a result, this model achieves the same computational complexity as Model 2.

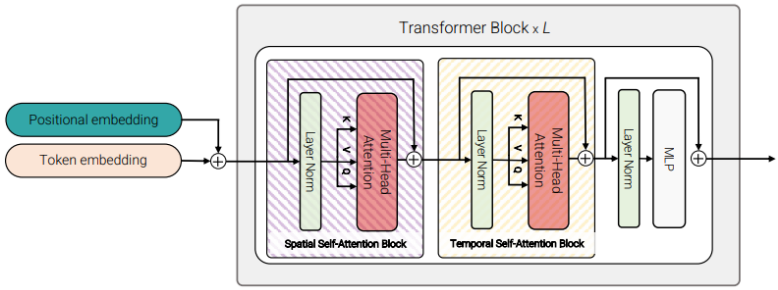

Fig. 63. An overview of the factorised self-attention [209].

Model 4 (Factorised dot-product attention): This model achieves the same computational complexity as Models 2 and 3, while preserving the parameter count of the unfactorised Model 1. Within a given layer, spatial and temporal information are processed in parallel via a split of the attention heads: certain heads attend to spatial relationships, whereas others attend to temporal relationships. Although the factorisation of the spatial and temporal dimensions is analogous in spirit to Model 3, the distinction lies in the factorisation of the multi-head dot-product attention operation itself (Fig. 1). More concretely, attention weights for each token are computed independently over the spatial and temporal dimensions, employing different heads for each dimension.

4.28. Deformable Attention Transformer (DAT)

Employing dense attention mechanisms, such as those in ViT, results in high memory usage and computational overhead, and can cause feature representations to be influenced by irrelevant regions outside the area of interest. Conversely, sparse attention mechanisms used in models like PVT or Swin Transformer are data-agnostic and may restrict the network's capacity to capture long-range dependencies. To address these limitations, the Deformable Attention Transformer (DAT) [210] was proposed a hierarchical

vision Transformer adaptable for both image classification and dense prediction tasks. DAT introduces a novel deformable self-attention module, where the key and value positions in self-attention are selected in a data-dependent manner. In this architecture, the standard multi-head self-attention (MHSA) layers of the Transformer are replaced by deformable attention modules and combined with MLPs to form deformable vision transformer blocks. This design allows the DAT model to learn sparse attention patterns dynamically based on input data and to effectively model geometric transformations in visual information.

DAT adopts a pyramid structure that is broadly suited to a wide range of vision tasks requiring multiscale feature maps. As shown in Fig. 64, an input image of size $H \times W \times 3$ is initially embedded through a 4×4 non-overlapping convolution with a stride of 4, followed by a normalization layer, yielding patch embeddings of shape $\frac{H}{4} \times \frac{W}{4} \times C$. A hierarchical feature pyramid is then constructed through a four-stage backbone with progressively increasing stride.

Within the third and fourth stages of DAT, alternating local attention and deformable attention blocks are employed. Specifically, the feature maps are first processed by a window-based local attention module to consolidate local information, and subsequently forwarded through a deformable attention block to capture global dependencies among the locally enriched tokens. This interleaved design, combining attention blocks with both local and global receptive fields, enables the model to learn powerful and expressive representations.

Deformable attention is not applied in the first two stages, for two primary reasons. First, these early stages are dedicated primarily to learning local features, a context in which deformable attention provides limited benefit. Second, the keys and values at these early stages span a large spatial extent, which substantially raises the computational overhead associated with dot product computations and bilinear interpolation operations within deformable attention. Consequently, deformable attention is reserved exclusively for the final two stages, while the earlier stages employ the shifted-window attention mechanism from Swin Transformer, ensuring more computationally efficient and effective local feature extraction.

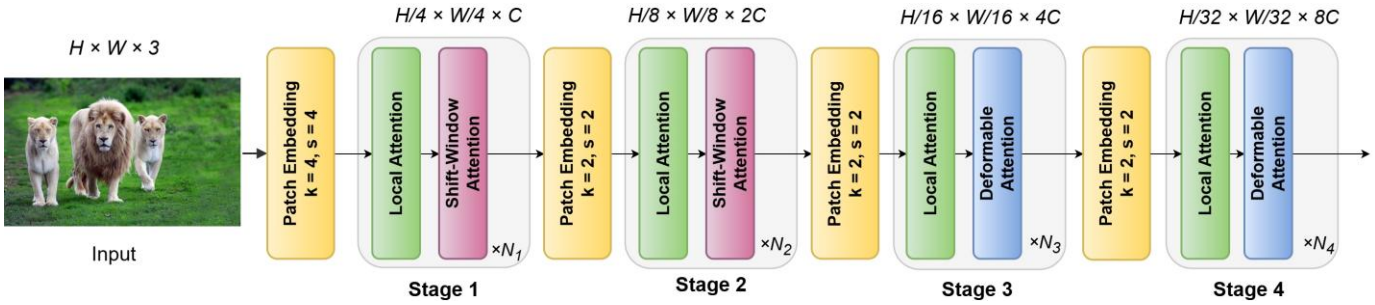

Fig. 64. An overview of the Deformable Attention Transformer (DAT) architecture.

Figure 65 depicts the Deformable Attention module. As seen on the left, a set of reference points is uniformly distributed across the feature map. The offsets of these points are learned from the query features through an offset network. Using these deformed sampling locations, the corresponding keys and values are projected from the sampled feature representations, as illustrated on the right. Furthermore, a relative position bias is derived from the deformed points and integrated into the multi-head attention computation, which subsequently produces the transformed output features.

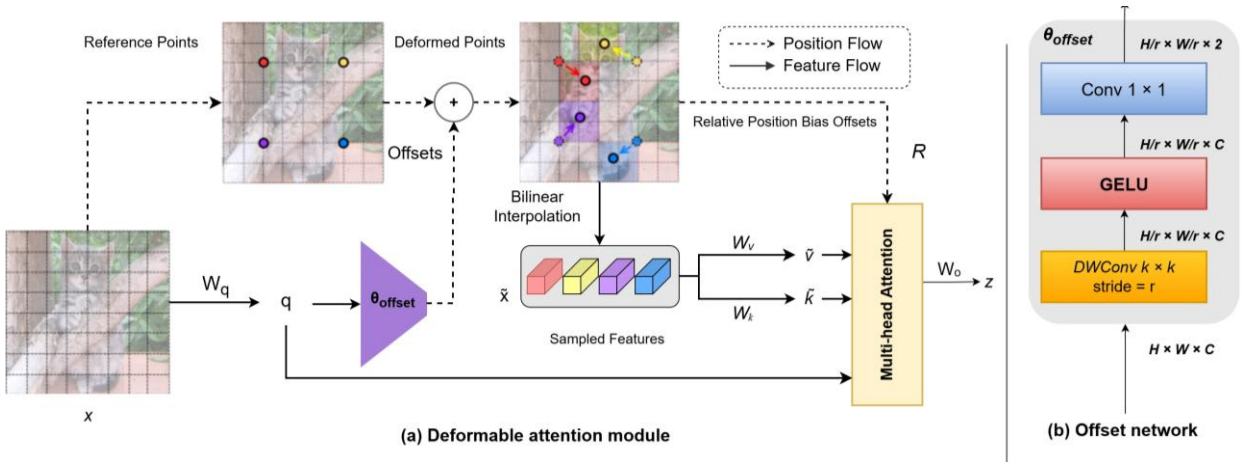

Fig. 65. The Deformable Attention module. (a) Information flows within the deformable attention mechanism. (b) Detailed structure of the offset generation network, with the input and output feature map dimensions annotated for each layer.

4.29. Spiking Transformer

Spiking Neural Networks (SNNs) have emerged as a promising alternative to traditional Artificial Neural Networks (ANNs), providing a biologically inspired and energy-efficient computational framework [211]. Unlike conventional deep learning models that process information through continuous-valued signals, SNNs rely on discrete spike sequences for both computation and communication. Spiking neurons transform continuous input values into spike sequences, with common examples including the Leaky Integrate-and-Fire (LIF) neuron [212] and PLIF [213], among others.

Self-attention, the core component of the Transformer architecture, not only enables selective focus on relevant information but also mirrors important characteristics of human biological systems [214, 215]. Given these biological parallels, incorporating self-attention into Spiking Neural Networks (SNNs) offers a promising direction for advancing deep learning. However, building Transformer-like models on SNNs requires a self-attention variant that is both effective and computationally efficient, particularly one that avoids multiplication operations.

To address this, Zhou et al. [216] propose a novel Spiking Self-Attention (SSA) mechanism together with a powerful framework called Spiking Transformer (Spikformer), which leverages both the representational strength of self-attention and the biological efficiency of SNNs. Fig. 66 provides an overview of Spikformer. Given a 2D image sequence $I \in \mathbb{R}^{T\times C\times H\times W1}$, the Spiking Patch Splitting (SPS) module linearly projects it into a D dimensional spike-form feature vector and then divides it into a sequence of N flattened spike-form patches$x$. Since floating-point position embeddings cannot be used in SNNs, a conditional position embedding generator [217] is employed to produce spike-form relative position embeddings (RPE), which are added to the patch sequence$x$ to obtain X_0. This generator consists of a 2D convolution layer (Conv2d) with kernel size 3, batch normalization (BN), and a spiking neuron ($\mathcal{SN}$) layer. The encoded sequence X_0 is then fed into the L-block Spikformer encoder. Similar to the standard ViT encoder, each Spikformer encoder block comprises a Spiking Self-Attention (SSA) module and an MLP block, with residual connections applied to both. SSA, the central component of the encoder, efficiently models local and global image information using spike-form Query (Q), Key (K), and Value (V) representations, without relying on softmax. Finally, a global average pooling (GAP) layer is applied to the processed feature, producing a D-dimensional representation that is passed to the fully connected classification head (CH) to generate the prediction Y.

The Spiking Self-Attention (SSA) module first generates the query (Q), key (K), and value (V) using learnable matrices. These are then transformed into spiking sequences through distinct spike neuron layers:

$$\begin{cases} Q = \mathcal{SN}_Q\left(BN(XW_Q)\right) \\ K = \mathcal{SN}_k\left(BN(XW_K)\right) \\ V = \mathcal{SN}_v\left(BN(XW_V)\right) \end{cases} \tag{74}$$

where $Q,K,V \in \mathbb{R}^{T\times N\times D}$. The attention matrix is computed using pure spike-form queries and keys, which contain only binary values (0 and 1). To control the potentially large results of matrix multiplication, a scaling factor s is introduced, following the design of vanilla self-attention [20]. Importantly, s does not alter the fundamental properties of SSA. As illustrated in Fig. 59, the spike-friendly SSA is defined as:

$$SSA'(Q,K,V) = \mathcal{SN}(QK^TV * s) \tag{75}$$

$$SSA(Q,K,V) = \mathcal{SN}\left(BN\left(Linear\left(SSA'(Q,K,V)\right)\right)\right) \tag{76}$$

Building on this foundation, Spike-driven Transformer [218] refines the architecture by introducing a novel self-attention block with masking and sparse addition to improve power efficiency. Inspired by MS-ResNet [219], it also redefines residual connections to propagate membrane potential rather than spikes.

Subsequent studies extended these ideas, giving rise to models such as Spikingformer [220], Qkformer [221], Spike-driven transformer v2 [222], Spikresformer [223], SWformer [224], Spike2former [225], and the Spiking Transformer with Spatial-Temporal Attention [226], each offering distinct optimizations for spike-based computation.

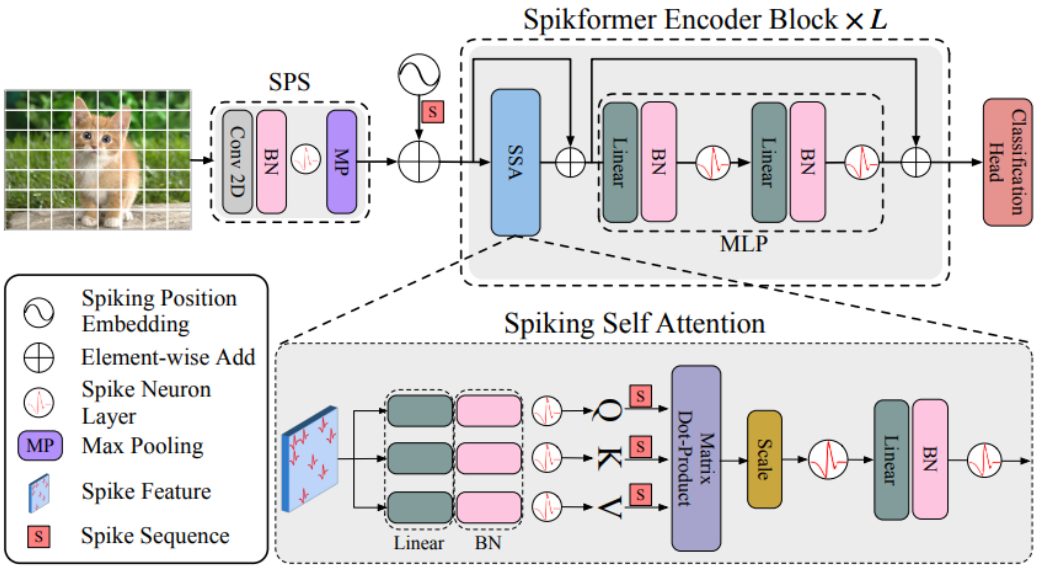

Fig. 66. The overview of Spiking Transformer (Spikformer) [216].

5. Application

Attention models have emerged as a prominent research focus due to their intuitive nature, versatility, and interpretability. They have been applied across numerous domains, yielding significant performance gains and enabling improved representation learning for entities such as documents, images, and graphs. In certain applications, such as question answering, machine translation, document representations/embeddings, and graph embeddings, these models have fundamentally transformed the field, largely because of their substantial impact on task performance [38, 39].

Given the wide scope of applications, this work concentrates on the use of attention modeling within the domains of Natural Language Processing, Computer Vision, Recommender Systems, and Sensor Data Analysis.

5.1. Natural Language Processing (NLP)

In the Natural Language Processing (NLP) domain, attention mechanisms enable models to focus on the most relevant parts of the input sequence, align input and output sequences, and capture long-range dependencies. Natural Language Processing encompasses several core subdomains, including language modeling, machine translation, question answering, speech recognition, text summarization, text classification and representation, sentiment analysis, and the development of pre-trained language models.

Language modeling involves predicting the next word, character, or token in a text sequence based on the preceding elements. It is a core task in natural language processing (NLP) and underpins many language-related applications. Language models typically operate in two main stages: pretraining and transfer learning. During pretraining, an objective function is used to learn the network's initial parameters, capturing a general language representation. This is followed by the transfer learning phase, in which the pretrained parameters are adapted or fine-tuned for a specific downstream task. To correctly align input and output words, the attention layer guides the decoder in identifying which inputs are most important, enabling it to focus on the relevant context when generating each output token. Initially, both inputs and targets are converted into embeddings, which serve as initial word representations. The attention mechanism then leverages these encoded representations [227].

In machine translation (MT), question answering (QA), and automatic speech recognition (ASR), attention primarily enables the alignment of input and output sequences while capturing long-range dependencies. Neural **Machine Translation** (NMT) employs neural networks to translate text between languages, where sentence alignment, particularly in longer sentences poses a significant challenge. Attention mechanisms are a central factor behind the success of sequence-to-sequence (seq2seq) models in tasks such as NMT and ASR. In seq2seq models, attention operates over the encoded sequence to guide the decoder in determining which parts of the sequence should be translated or recognized [228].

In automatic **speech recognition** (ASR) tasks, attention aligns acoustic frames by extracting information from anchor words to identify the primary speaker while suppressing background noise and competing speech. This ensures that only relevant speech information is passed to the decoder, offering a direct way to align each output symbol with specific input frames through selective noise decoding.

In **question answering** (QA), attention enhances question understanding by focusing on the most pertinent parts of the question and leveraging memory networks to store large amounts of information, thereby supporting more accurate answer retrieval [179].

Text classification involves assigning labels to text and has wide-ranging applications, such as topic labeling and spam detection. In these tasks, self-attention is primarily employed to construct more effective document representations [229].

Attention mechanisms also play an important role in **text/document summarization**. In summarization tasks, attention addresses several critical challenges: (1) modeling important keywords, (2) generating summaries of abstract sentences, (3) capturing hierarchical sentence structure, (4) minimizing repetitions and inconsistencies, and (5) producing concise sentences while preserving semantic meaning [39].

In **sentiment analysis**, self-attention enables models to focus on the words most relevant for determining the sentiment of the input [230].

Table 1 highlights several notable studies that demonstrate the application of attention mechanisms across various areas of natural language processing (NLP).

Table1. Attention methods in several natural language processing (NLP) sub-areas

Tasks	Methods	References
Language modeling	Transformer	Glam [231], Gopher [232], Chinchilla [233], LaMDA [234], Palm [192], LLaMA [190], LLaMA 2 [235]
	Transformer-XL	[168], Xlnet [169], Ernie 3.0 [236]
	GPT	GPT-Neo [194], [237]
Machine Translation	Transformer	[20], [238],[239]
	Cross-Attention	[240]
Question Answering	Transformer	[241], [242]
	Hierarchical Attention	[243], [244]
Automatic Speech Recognition (ASR)	Transformer	[245], [245], [246]
	BERT	[247]
Automatic Text Summarization	Transformer	[248], [249]
	BERT	[250], [251]
Text Classification and Representation	Graph Attention Networks (GAN)	[245], [252]
	Multi-head attention	[253], [254]
Sentiment Analysis	Graph Attention Networks	[74], [255], [256]
	BERT	[257], [258], [259], [260]

5.2. Computer vision

Visual attention has become widely adopted in core computer vision (CV) tasks, allowing models to concentrate on the most relevant image regions while capturing structural long-range dependencies among different components. Key areas of application include image classification, segmentation, object detection, image and video captioning, image generation, object tracking, and action recognition.

In **image classification,** attention mechanisms are used not only to identify and emphasize the most relevant regions within an input image but also to reduce the computational burden of CNNs by processing these regions at high resolution. Some approaches employ attention to automatically learn where to focus, highlighting the most informative areas, while others refine discriminative regions through feature recalibration or enhance ensemble predictors via attention.

Unlike traditional multi-scale feature fusion methods that compress an entire image into a static representation, attention enables the network to selectively emphasize key features without requiring additional supervision. This avoids redundancy from generating multiple similar feature maps and instead highlights salient features most useful for the task. Attention modules have also strengthened semantic **segmentation** networks, yielding more effective models for pixel-wise recognition [261].

Object detection seeks to predict a set of bounding boxes together with category labels for each object of interest. Visual attention significantly enhances this task by aiding in both object localization and recognition.

Beyond detection, visual attention has demonstrated strong effectiveness in structured prediction tasks such as **image/video captioning**. This success is rooted in the principle that human vision does not process an entire scene simultaneously; rather, it selectively attends to specific regions of the visual space as needed. Instead of encoding an image into a fixed static vector, attention enables image features to adapt dynamically to the sentence context, yielding richer and more descriptive captions—particularly for

cluttered scenes [102].

In **image generation**, attention mechanisms selectively focus on regions of the input image while iteratively reconstructing specific scenes. They were first introduced into generative adversarial networks (GANs) [262] to address the challenges of modeling images with structural constraints. Self-Attention Generative Adversarial Networks (SAGANs) [263] extend convolutional GANs by incorporating self-attention, where the response at each position is computed as a weighted sum of features from all positions. This design allows for efficient modeling of long-range dependencies, surpassing convolution operations that are limited to local receptive fields.

Object tracking in videos is a central problem in computer vision that has attracted significant research interest. Visual attention mechanisms address this challenge by directing the model to focus on the most relevant regions of the input, thereby enabling the extraction of more discriminative features.

Likewise, attention provides substantial benefits for **action recognition** in videos by capturing spatial and temporal dependencies. Classical approaches often face difficulties in isolating informative motion patterns from sequences of frames. Attention mechanisms overcome this limitation by concentrating the network's processing on the most relevant joints or motion features.

Table 2 highlights several representative studies showcasing the application of attention mechanisms across diverse domains of computer vision.

Table2. Attention methods in several computer vision (CV) sub-areas

Tasks	Methods	References
Image Classification	Channel Attention	Fcanet [89], [264], [265]
	Channel & Spatial Attention	[266], [267], [268]
	Vision Transformer (ViT)	Crossvit [202], MViT [208], [269], MedViT [270], [271]
Segmentation	Cross attention	[272], [273], [274]
	Criss-Cross Attention	Ccnet [95], [275]
	Channel & Spatial Attention	DANet [84], CS-Net [106], [276]
	Transformer	Spike2former [225], Segmenter [277], Transunet [278], Oneformer [279]
Object Detection	Transformer	DETR [197], ViT-YOLO [280], MViTv2 [281], SpikingViT [282]
	Channel & Spatial Attention	SCANet [283], [284]
	Cross Attention	[280], [285]
image/video Captioning	Channel & Spatial Attention	Sca-cnn [102], [286], [287]
	Spatial -Temporal Attention	STAT [127], [288]
	Transformer	[289], [290], [291]
Image Generation	Transformer	Cogview [292], Styleswin [293], Maskgit [294], Cogview2 [295], Muse [296]
Object Tracking	Spatial-Temporal Attention	[297], [298], [299]
	Transformer	Transtrack [300], Motr [301], Trackformer [302], [303]
Video Action Recognition	Spatial-Temporal Attention	[304], [305]
	Transformer	[306], Mm-vit [307],Svformer [308], [309]

5.3. Recommender Systems

Attention has been extensively applied in recommender systems, with key applications in session-based recommendation, sequential recommendation, and user and item representation learning.

In **session-based recommendation**, attention is used to capture users' short-term preferences by focusing on their most recent actions within a session. In **sequential recommendation**, it identifies the relative importance of past user interactions when predicting the next item. Furthermore, attention mechanisms aggregate features from user and item attributes to enhance representation learning.

Table 3 summarizes several influential studies that demonstrate the use of attention mechanisms across different domains of

recommender systems.

Table 3. Attention methods in a number of subfields of recommendation systems

Tasks	Methods	References
Session-Based Recommendation	Graph Attention	[310], [311]
Sequential Recommendation	Transformer	BERT4Rec [28], SSE-PT [312], [313]
User and Item Representation	Graph Attention	KGAT [314], MGAT [315], [316], [317]

5.4. Sensor Data Analysis

The attention mechanism has demonstrated strong effectiveness in the sensor data domain, with key applications in time-series analysis, human activity recognition, predictive maintenance, robotics, environmental monitoring, and related areas.

Sensor-based **Human Activity Recognition** (HAR) systems employ wearable or ambient sensors such as accelerometers, gyroscopes, and RFID devices to capture and classify human activities. Attention enhances these systems by focusing on the most informative time steps and identifying critical sensors or body parts, thereby improving recognition performance.

Predictive maintenance applies data analysis and machine learning to forecast equipment or machinery failures, enabling timely maintenance before breakdowns occur. Sensor data such as vibration, temperature, pressure, sound, and current are commonly used. Attention mechanisms highlight critical sensor readings that indicate degradation or impending failure, while also detecting anomalies or rare events across long sequences.

In **robotics**, sensors are essential for enabling robots to perceive and interact with their environment. They collect information about the surroundings, the robot's internal state, and ongoing tasks, supporting functions such as control, navigation, object detection, and manipulation. Attention mechanisms play a crucial role by enhancing perception, decision-making, and control, allowing robots to focus on the most relevant aspects of their sensor data or environment.

Environmental monitoring involves collecting, analyzing, and interpreting data on environmental conditions through diverse sensors and technologies. It is critical for detecting environmental changes, assessing pollution levels, and guiding policies related to sustainability and public health. Major domains include air quality monitoring, water quality monitoring, soil monitoring, and weather and climate monitoring. In this setting, attention mechanisms strengthen the analysis and interpretation of multi-sensor data by enabling models to prioritize the most relevant features, locations, and time steps, an especially important capability when handling large, noisy, and complex datasets.

Table 4 highlights several significant studies that demonstrate the application of attention mechanisms across sensor-driven domains.

Table 4. Attention mechanisms in sensor-based tasks

Tasks	Methods	References
Sensor-based HAR	Transformer	[318] [319], [320]
Predictive Maintenance	self-attention	[321], [322]
Robotics	Graph Attention	[323], [324], [325]
	Transformer	[326],[327], [328]
Environmental Monitoring	Transformer	[329], [330], [331], [331]
	Spatial-Temporal Attention	[332], [333]

6. Comparison of Models

In the previous section, we reviewed a range of attention mechanisms and Transformer-based models. These models are designed to handle different tasks and pursue varied goals and have been evaluated on diverse datasets using a range of performance metrics tailored to their applications. As such, creating a unified and fully objective comparison is inherently difficult. Nevertheless, to improve clarity and support reader understanding, we present a comparative overview of these models. Table 5 presents a structured cross-model comparison, outlining the key attention mechanisms, results, core ideas, advantages, limitations, and applications of each model. The term "key attention mechanism" refers to the fundamental attention method employed by a given model, helping readers understand how attention mechanisms relate to effective, task-performing models. This table serves as a concise reference, highlighting both similarities and differences among the models while acknowledging the inherent limitations of cross-task comparisons.

Among NLP-focused architectures, a clear evolutionary trajectory is visible. BERT's bidirectional encoder provides rich contextual representations well-suited for understanding tasks such as sentiment analysis and question answering, while GPT's causal decoder is naturally designed for autoregressive generation, excelling in language modeling and open-ended text synthesis. T5 unifies both paradigms within a text-to-text encoder-decoder framework, treating every NLP problem as text transformation and achieving remarkable flexibility across diverse tasks without sacrificing architectural simplicity. XLNet addresses BERT's masked-token independence assumption which ignores dependencies among masked positions, through permutation-based language modeling, enabling bidirectional context within an autoregressive setting. BART introduces controlled denoising as a pretraining objective, corrupting input text with arbitrary noise and learning to reconstruct the original, making it particularly effective for sequence-to-sequence generation and abstractive summarization. Switch Transformer demonstrates that sparse mixture-of-experts (MoE) layers can scale model capacity to trillions of parameters without proportional increases in computation by activating only a subset of experts per token. Meanwhile, LLaMA integrates Rotary Position Embedding (RoPE), SwiGLU activations, and RMSNorm into an efficient foundation widely adopted for instruction-following.

A central tension across all surveyed models is the trade-off between computational efficiency and representational power. Standard multi-head self-attention scales quadratically with sequence length ($O(n^2)$), imposing significant memory and time costs for long documents or high-resolution images. This bottleneck has motivated a wave of efficient attention variants. Linformer approximates self-attention through low-rank decomposition, achieving linear complexity while preserving key structural properties. Performer employs random feature approximations (FAVOR+) with theoretical accuracy guarantees, enabling unbiased or nearly unbiased estimation of the attention matrix. Reformer replaces dot-product attention with locality-sensitive hashing (LSH) to group similar keys into buckets, reducing complexity to $O(n\log n)$. BigBird and Longformer adopt hybrid local-global attention patterns, combining sliding windows with a small number of global tokens to preserve long-range dependencies at linear cost. Critically, FlashAttention takes a fundamentally different approach: rather than approximating attention, it achieves exact attention with dramatically reduced memory bandwidth through IO-aware tiled computation and has become the de facto standard for large-scale training. This highlights that hardware-aware exact computation can outperform approximate methods in real wall-clock time.

Another critical theme is the evolution of positional encoding strategies. The original Transformer relies on fixed sinusoidal embeddings that generalize poorly to sequences longer than those seen during training, a limitation known as limited length extrapolation. Transformer-XL addresses this through segment-level recurrence combined with relative position encodings, allowing context to persist across document boundaries and enabling memorization of long-range dependencies without recomputing hidden states from scratch. RoFormer introduces Rotary Position Embedding (RoPE), which encodes positions as rotation matrices applied directly to query and key vectors, naturally incorporating relative dependencies through a geometric formulation. RoPE now underpins modern large language models (LLMs) including LLaMA, and PaLM. ALiBi takes a yet more radical approach, replacing positional embeddings entirely with a fixed linear bias proportional to query-key distance, which penalizes attention scores for distant token pairs. This design enables strong extrapolation to sequences far longer than those seen during training with no additional parameters or compute overhead. Together, these schemes trace a trajectory from fixed sinusoidal embeddings toward lightweight, flexible, and inductive-bias-driven positional strategies that substantially improve length generalization.

In computer vision, models architectures have evolved from CNN-based attention mechanisms to patch-based vision Transformers, and further to hierarchical or multi-scale designs, object detection frameworks, and video models. CNN-based attention models such as SENet, CBAM, R-STAN, and ECA enhance standard convolutional neural networks by injecting lightweight attention modules. These modules adaptively recalibrate feature responses, allowing CNNs to focus on informative channels or spatial regions with minimal computational overhead.

The Vision Transformer (ViT) pioneered the application of a pure Transformer to image classification. It divides an image into fixed-size patches and processes them analogously to words in a sentence. However, ViT requires large-scale pre-training to compensate for its lack of inherent locality bias. DeiT mitigates this through knowledge distillation from a CNN, while T2T-ViT addresses it via progressive token aggregation that captures local structure within patches. CrossViT demonstrates that dual-branch cross-attention fusion outperforms single-branch models at comparable computational cost.

Despite its success in classification, the original ViT employs a single-scale, uniform patch size throughout the network, which proves inefficient for dense prediction tasks such as segmentation or object detection. This limitation spurred the development of hierarchical Vision Transformers. PVT employs Spatial Reduction Attention (SRA) to reduce the size of keys and values before applying attention. Swin Transformer achieves linear complexity via a shifted-window mechanism and a hierarchical design, making it the dominant backbone for dense prediction. TNT (Transformer-in-Transformer) captures fine-grained details using two transformers: an inner Transformer processes sub-patches within a patch, while an outer Transformer processes the patches themselves. MViT leverages multi-scale pooling attention for effective image and video understanding. DAT introduces a deformable attention module that learns to focus on the most relevant parts of an image on the fly.

For efficient object detection, the hybrid transformer architecture DETR (DEtection TRansformer) combines a conventional CNN backbone, a Transformer encoder–decoder, and learnable object queries to cast detection as a set prediction problem, eliminating many hand-designed components like anchors and non-maximum suppression. Finally, ViViT extends the Transformer architecture from images to videos. Its goal is to efficiently model complex spatiotemporal relationships across frames by factorising spatial and temporal attention, thereby reducing computational cost while capturing long-range dependencies.

Taken together, no single architecture dominates across all settings. Instead, model selection should be guided by the task type, dataset scale, domain, and compute budget.

It is important to note that the results presented in Table 5 are taken directly from the original papers of the respective models, ensuring consistency with originally reported performance figures. While these models are typically available in multiple sizes, ranging from Tiny, Small, and Base to Large and X-Large, each offering different trade-offs between accuracy and computational cost, the results shown in Table 5 reflect only the performance of the Base variant. Larger variants such as Large or X-Large would likely yield higher accuracy, whereas smaller versions like Tiny or Small would offer greater efficiency at the cost of reduced performance. Readers should take this limitation into account when comparing models or selecting a variant for their specific application.

Table 5. Comparative analysis of different models.

Model	Reference	Category	Key Attention Mechanism	Dataset	Metric and results	Key Idea / Innovation	Advantages	Limitations	Application
Vanilia Transformer	[20]	NLP	Multi-Head Attention (MHA)	WMT 2014 English-to-German	BLEU: 27.3	Original Encoder-Decoder Transformer,	Parallelizable, strong seq2seq performance	Quadratic attention cost $O(n^2)$, long-context issues	Machine Translation, Seq2seq Tasks
				WMT 2014 English-to-France	BLEU: 38.1				
BERT	[145]	NLP	Bidirectional Multi-Head Attention (MHA)	GLUE [334]	GLUE: 80.5	Bidirectional Encoder, Masked Language Modeling	Deep contextual embeddings, Strong fine-tuning	Pre-training expensive, no generative ability	Natural Language Understanding (NLU) tasks, (Question Answering (QA), Text Classification)
				SQuAD v1.1 [335] (question answering, QA)	F1: 93.2				
				SQuAD v2.0	F1: 83.1				
RoBERTa	[150]	NLP	Bidirectional MHA	GLUE	GLUE: 88.5	Optimized BERT (longer training, more data, dynamic masking)	Stronger performance than BERT	Higher computational cost	Natural Language Understanding (NLU) tasks
ALBERT	[147]	NLP	Bidirectional MHA	GLUE	GLUE:89.4	Optimized BERT (Parameter sharing + factorized embeddings)	Lightweight, Efficient	Relies on large-scale pretraining for good performance	Natural Language Understanding (NLU) tasks
				SQuAD v1.1	F1: 94.1				
				SQuAD v2.0	F1: 88.1				
GPT	[164]	NLP	Multi-Head Attention (MHA)	Tasks and datasets are various	-	Decoder-only, Autoregressive	Powerful text generation	Requires massive data/compute	LLMs, Text/Code Generation

Model	Reference	Category	Key Attention Mechanism	Dataset	Metric and results	Key Idea / Innovation	Advantages	Limitations	Application
Transformer-XL	[336]	NLP	Relative MHA	WikiText-103 [337]	PPL: 18.3	Segment recurrence + Relative positional encoding	Handles long context	Still quadratic attention cost	Language Modeling, Long Sequences
				One Billion Word [338]	PPL:21.8				
				Penn Treebank [339]	PPL: 54.52				
XLNet	[169]	NLP	Permutation-based MHA	GLUE	GLUE: 87.4	Permutation-based training (bidirectional & autoregressive)	Outperforms BERT on many benchmarks	Complex training	Natural Language Understanding (NLU) tasks
				SQuAD v1.1	F1: 94.0				
				SQuAD v2.0	F1: 87.8				
BART	[170]	NLP	BERT encoder + GPT decoder (MHA)	SQuAD v1.1	F1: 94.6	Encoder-decoder, denoising autoencoder	Good seq2seq generation	Pre-training cost	Seq2seq tasks Summarization, Translation
				SQuAD v2.0	F1: 89.2				
Fast Transformer	[57]	Efficient NLP	Multi-Query Attention (MQA)	WMT 2014 English-to-German	BLEU: 26.5	Single key-value per head, multiple queries	Memory-efficient	Reduced Expressiveness	Efficient Seq2seq tasks
T5	[172]	NLP	Multi-Head Attention (MHA)	Tasks and datasets are various	--	Text-to-text unified framework	General-purpose multitask	Huge pre-training cost	Seq2seq tasks Translation, summarization
Longformer	[175]	NLP	Local + Global Sparse Attention	Tasks and datasets are various	--	Sliding window + local/global attention	Handles long documents	May miss some global context	Document Classification, QA
BigBird	[177]	NLP	Random+ Local + Global Sparse Attention	Tasks and datasets are various	--	Random + window + global attention	Linear complexity, universal Approximator	More complex attention	QA, text Classification, text summarization
Performer	[180]	Efficient NLP	Kernel-based linear attention	--	--	Fast Attention Via Positive Orthogonal Random features (FAVOR+)	Linear scaling, efficient	Kernel choice affects accuracy	Efficient NLP, Long-sequence tasks

Model	Reference	Category	Key Attention Mechanism	Dataset	Metric and results	Key Idea / Innovation	Advantages	Limitations	Application
Linformer	[181]	Efficient NLP	Multi-Head Linear Self-Attention	SST-2 [340] (sentiment classification)	Acc: 93.2	Low-rank projection of keys/values	Linear complexity $O(n)$	Approximation may lose info	Long sequence tasks
				IMDB [341] (sentiment classification)	Acc: 94.2				
				QNLI [335](natural language inference)	Acc: 90.8				
				QQP [342] (textual similarity)	Acc: 90.5				
Reformer	[182]	Efficient NLP	Locality-Sensitive Hashing (LSH) Attention	WMT 2014 English-to-German	BLEU: 29.1	LSH attention + reversible residual layers	Sub-linear attention cost $O(Llog\ L)$	Approximation errors	Machine translation, Efficient NLP
RoFormer	[183]	NLP	Multi-Head Attention (MHA)	WMT 2014 English-to-German	BLEU: 27.5	Rotary Position Embedding (RoPE)	Efficient relative position modeling	Restrict expressiveness because of complex rotations in query-key	Natural Language Understanding (NLU) tasks
ALiBi	[185]	NLP	Linearly Biased Attention	Tasks and datasets are various	-	Linear bias for attention	extrapolate to longer sequences than seen during training	Limited Expressiveness	Natural language generation tasks, LLMs
Switch Transformer	[187]	NLP	Multi-Head Attention (MHA)	GLUE	GLUE: 85.2	Use of Mixture of Experts (MoE) in Transformer	Scales to trillion params efficiently	Routing complexity	Seq2seq tasks Large-scale LLMs
				Super GLUE [343]	GLUE: 73.0				
LLaMA	[190]	NLP	Multi-Head Attention (MHA)	Tasks and datasets are various	-	Use RoPE+ RMSNorm + SwiGLU to enhance performance	Strong performance with fewer params	Expensive to train, and run	Natural language generation tasks,

LLMs

Model	Reference	Category	Key Attention Mechanism	Dataset	Metric and results	Key Idea / Innovation	Advantages	Limitations	Application
SENet	[78]	Vision	Channel Attention	ImageNet [200]	Top-1 acc: 81.3 %	Channel-wise attention	Boosts channel importance	Adds small overhead	Image Classification
Convolutional Block Attention Module (CBAM)	[79]	Vision	Channel & Spatial Attention	ImageNet	Top-1 acc (ResNet101 + CBAM): 78.49%	Channel + spatial attention	Combines spatial & channel	Adds computation	Image classification, object detection
Residual Spatial-Temporal Attention Network (R-STAN)	[128]	Vision	Spatial & Temporal Attention	UCF101	Acc: 92.7 %	Residual spatial + temporal attention	Captures spatio-temporal feature	High memory	Action Recognition in video
				HMDB51	Acc: 64.4 %				
Efficient Channel Attention (ECA)	[87]	Vision	Channel Attention	ImageNet	Top-1 acc: (ResNet + ECA): 78.65%	Lightweight channel attention	Efficient, low overhead	Slightly less expressive	Image Classification
ViT	[22]	Vision	Patch-based MHA	ImageNet	Top-1 Acc: 79.8 %	Transformer on image patches	Competitive with CNNs at scale	Require large scale data for pretraining	Image classification
DETR	[197]	Vision	Multi-Head Attention (MHA)+ Cross attention	COCO 2017 [344]	AP:42.0 (ResNet-50 backbones), AP:43.5 (ResNet-101 backbones)	Use CNN backbone+ Transformer Encoder + Transformer Decoder+ feed-forward network (FFN)	No hand-designed components, Unified Architecture, Global Context	Slow Training Convergence, High Computational Cost	Object Detection
DeepViT	[198]	Vision	Patch-based MHA	ImageNet	Top-1 Acc: 80.9 %	Deeply stacked ViT	Improved representation	Training instability	Image classification
DeiT	[23]	Vision	Patch-based MHA + distillation	ImageNet	Top-1 Acc: 81.8 %	Use distillation token for Data-efficient ViT	Works with small datasets	Processes fixed-size patches	Image classification

Model	Reference	Category	Key Attention Mechanism	Dataset	Metric and results	Key Idea / Innovation	Advantages	Limitations	Application
T2T-ViT	[199]	Vision	Patch-based MHA+ Progressive tokenization	ImageNet	Top-1 Acc: 83.3%	Tokens-to-Token structure	Reduces token redundancy	Complexity	Image classification
CrossViT	[202]	Vision	MHA + Cross Attention	ImageNet	Top -1 Acc: 82.8 %	Multi-scale patch tokens + Cross-attention	Multi-scale fusion	Complexity	Image classification
PVT	[203]	Vision	Spatial-Reduction Attention (SRA) (MHA with Spatial Reduction)	ImageNet	Top-1 Acc: 81.7 %	Pyramid Structure ViT	Multi-scale feature extraction	Complexity	Image classification, Object detection, Semantic segmentation
				COCO 2017	AP: 40.4 (RetinaNet framework [345])				
				ADE20K [346]	mIoU: 42.1%				
Swin Transformer	[204]	Vision	Shifted window MHA	ImageNet	Top-1 Acc: 83.3%	Shifted window multi-head attention	Efficient, hierarchical	Still complex	Image classification, Object detection, semantic segmentation
				ImageNet-22k [347]	Top-1 Acc: 86.4 %				
				COCO 2017	AP: 50.5 (Cascade Mask R-CNN framework [348])				
				ADE20K	mIoU: 51.6% (UperNet [349])				
TNT	[206]	Vision	Patch-based MHA	ImageNet	Top -1 Acc: 82.9 %	Dual-transformer (Inner transformer+ Outer transformer)	Local + global features	Higher compute	Image classification

Model	Reference	Category	Key Attention Mechanism	Dataset	Metric and results	Key Idea / Innovation	Advantages	Limitations	Application
MViT	[208]	Vision	Multi-Head Pooling Attention (MHPA)	Kinetics-400	Top-1 Acc: 81.2 %	Multiscale ViT	Efficient multi-scale features	Complex	Video Recognition, Image Recognition
				Kinetics-600	Top-1 Acc: 84.1 %				
				SSv2 [350]	Top-1 Acc: 68.7 %				
				ImageNet	Top-1 Acc: 84.8 %				
ViViT	[209]	Vision	MHA + Spatial & Temporal attention	Kinetics-400 [351]	Top-1 Acc: 81.7 %	Spatial - temporal attention on video patches	Captures video dynamics	High memory/computation	Video classification
				Kinetics-600 [351]	Top-1 Acc: 82.9 %				
DAT	[210]	Vision	Deformable Multi-Head Attention (DMHA)	ImageNet	Top-1 Acc: 84.0 %	Deformable self-attention	Focuses on salient regions	Complex	Image classification, Object detection, semantic segmentation
				COCO 2017	RetinaNet: 45.6 AP Cascade Mask R-CNN: 51.3 AP				
				ADE20K	mIoU: 50.5% (UperNet)				
Spikformer	[216]	Vision	Spiking Self-Attention (SSA)	ImageNet	Top-1 Acc: 74.81 %	Use of Spiking Neural Networks (SNNs) in Transformers	Energy Efficiency	Need Neuromorphic hardware	Neuromorphic vision
				CIFAR100	Top-1 Acc: 78.61%				
				CIFAR10-DVS [352]	Acc: 80.9%				
				DVS128	Acc: 98.3%				

Notes:

The Bilingual Evaluation Understudy (BLEU) [353] is a commonly used metric for evaluating the quality of text produced by machines, especially in machine translation. The BLEU analyzes the overlap of n-grams (word sequences) to determine how closely the candidate text resembles a reference text. Three main factors go into its calculation: (1) n-gram accuracy between the reference and candidate texts; (2) a shortness penalty (BP) to lessen the impact of overly long or short phrases; and (3) clipping to take repetitive word occurrences into consideration. The precision is determined by dividing the number of n-grams in the candidate text by the

number of matching n-grams. In order to prevent overcounting of repeated n-grams in the candidate, BLEU employs clipping and counts the maximum frequency of each n-gram match across all references. When calculating modified precision, the original count is replaced by the clipped count, which is the minimum between the maximum reference count and the candidate count. The purpose of BP is to lessen the impact of sentence length on the BLEU score. When the candidate sentence is shorter than the reference sentence, BP is applied [354].

The General Language Understanding Evaluation (GLUE) [334] benchmark is a collection of nine datasets designed to evaluate natural language understanding systems. It encompasses a variety of tasks across different datasets, including: Corpus of Linguistic Acceptability (CoLA) [355] for sentence acceptability; Stanford Sentiment Treebank (SST-2) [340] for sentiment analysis; Microsoft Research Paraphrase Corpus (MRPC) [356], Quora Question Pairs (QQP), and Semantic Textual Similarity Benchmark (STS-B) [357] for sentence similarity; Multi-Genre Natural Language Inference Corpus (MNLI) [358], Stanford Question Answering Dataset (QNLI) [335], and Recognizing Textual Entailment (RTE) (created by combination of RTE1 [359], RTE2, RTE3 [360], and RTE5 [361]) for question answering, Winograd Schema Challenge (WNLI) [362] used for coreference resolution. The overall GLUE score is calculated by averaging the performance across all nine datasets.

7. Research Directions and Future Aspects

In recent years, attention mechanisms have become a central innovation in deep learning, shaping much of the field's progress through profound breakthroughs. The framework discussed in this study offers an integrated perspective on their core principles and functionality. Despite these advances, many research avenues remain underexplored, providing ample opportunity for future discoveries and contributions. Closing these gaps could elevate the state of the art and encourage wider integration of advanced attention-based strategies in deep learning practice. This section highlights critical issues and challenges that warrant further investigation and development.

7.1. Computational Complexity

Attention-based architectures, particularly large-scale models such as Transformers, face a major limitation that their computational and memory requirements grow quadratically with sequence length due to the self-attention mechanism $(O(n^2))$. This quadratic scaling quickly becomes impractical for very long inputs. A significant strand of current research therefore aims to design methods that approximate or restructure attention to achieve linear complexity, while still preserving model accuracy and effectiveness.

The design of the attention distribution function is a key factor in shaping the computational demands of attention mechanisms. Some research [363, 364] indicates that refining this function can lead to more efficient attention operations, reducing overall computational cost without compromising performance.

Choromanski et al. [180] introduced Performers, a Transformer variant designed to approximate full-rank softmax attention with theoretical accuracy guarantees while reducing computational requirements from quadratic to linear in both time and memory, as discussed in Section 4.10.

Wang et al. [181] demonstrated that self-attention can be efficiently approximated through a low-rank matrix, enabling a reformulation of the mechanism that scales linearly rather than quadratically in both time and memory. Building on this idea, they introduced Linformer, a model that delivers accuracy comparable to conventional Transformers while achieving substantial gains in computational and memory efficiency, as discussed in Section 4.11.

Dao et al. [133] proposed FlashAttention, an optimization that accelerates attention computations while also reducing memory overhead. A key contribution of this work is the emphasis on IO awareness, which accounts for the cost of reading and writing across the different levels of GPU memory, as discussed in Section 3.14.

7.2. Generalization

Achieving strong generalization in attention models continues to be a major challenge. Many current approaches are designed for narrow applications and tend to deliver strong performance only within those limited contexts. For example, channel and spatial attention mechanisms often yield excellent results in classification tasks, since attention methods are typically optimized for high-level processing. However, their effectiveness decreases when applied to low-level vision problems. A promising research direction is to investigate how attention strategies developed in one domain can be successfully transferred and adapted to other domains [365].

7.3. Multi-Modal Applications

In recent years, multimodal applications have expanded rapidly, with attention mechanisms playing a central role in advancing multimodal learning. Multimodality is especially important for tackling complex tasks, as it enables models to capture synergistic relationships across different inputs, even when they are not temporally aligned. This has facilitated the development of increasingly advanced applications, including Human Communication Comprehension (HCC) [366],Visual Question Answering (VQA) [367],

emotion recognition [368], Text-to-image generation [369], and image/video captioning [370, 371].

Despite these advances, most existing work has concentrated on pairs of modalities, such as visual-textual [372], image and sound data [373], image and infrared data [284]. Architectures capable of efficiently handling more than two modalities remain relatively underexplored. There is still not enough work on multimodal learning that examines voice data, RGBD images, images from monocular cameras, and data from sensors, including accelerometers, gyroscopes, GPS, RADAR, and biological sensors.

Attention models hold promise for predicting relationships across labels, actions, and attributes within a unified framework. One potential direction is to integrate these attention mechanisms into graph attention networks, thereby producing models better suited to heterogeneous multimodal inputs.

7.4. Interpretability

Attention modules process very high-dimensional features, particularly in deep networks with many layers, which makes interpreting attention patterns and visualizing individual weights extremely difficult. Although several visualization strategies have been proposed, none provide a universally reliable solution across all modules. Moreover, different methods often emphasize distinct aspects of attention, which can sometimes yield inconsistent or even contradictory interpretations. A major challenge is to design visualization techniques that capture essential information without oversimplifying or distorting the underlying patterns. Understanding how attention weights relate to model interpretability remains an open area of research [374]. Future work could investigate the distribution of attention in existing models and explore how modifications to these distributions might yield more transparent explanations of model predictions.

7.5. Integration with External Memory

Applications such as Question Answering (QA) and chatbots require the capability to retrieve and learn from information stored in a database of facts. In these settings, the network receives both a knowledge base and a query, where only certain facts are directly relevant to the query. Combining attention mechanisms with external memory forms a key concept at the intersection of sequence modeling, reasoning, and memory-augmented neural networks. External memory is used to store the fact database, while attention selectively identifies and focuses on the most relevant pieces of information. This integration enables models to better manage long-term dependencies, reasoning, and compositional tasks compared to using attention or memory alone.

Although integrating attention with external memory is a powerful approach, it faces significant challenges. These include scalability concerns due to costly large-scale memory access, the difficulty of accurate information retrieval, complexities in memory management such as forgetting, updating, and compression, issues of training stability, the challenge of balancing generalization with consistency across parametric and non-parametric knowledge, and gaps in evaluation caused by the lack of standardized benchmarks. Over time, multiple models and techniques have been developed to explicitly or implicitly merge attention with external memory, particularly within Large Language Models (LLMs). Examples include Retrieval-Augmented Generation (RAG) [375], Retrieval-Enhanced Transformer (RETRO) [376], Perceiver IO [377], and Mamba [378].

7.6. Attention for Reinforcement Learning (RL)

Reinforcement learning (RL) is a branch of machine learning concerned with sequential decision-making, where the aim is to learn mappings from states to actions that maximize long-term reward [21]. An RL agent typically operates with a well-defined objective and relies on four central elements: the policy, reward, value, and environment models. The policy specifies the agent's course of action, while the environment provides evaluative feedback in the form of rewards. In fact, the environment model captures the dynamics of the surroundings, enabling the agent to interpret its context more effectively and improve decision-making. This framework allows the agent to train efficiently over many cycles without requiring explicit background information [379].

A valuable line of future research is the application of attention mechanisms within reinforcement learning and multi-agent settings. Such integration has the potential to enhance how information is processed and to support more effective decision-making.

One of the persistent difficulties in reinforcement learning is the Credit Assignment Problem (CAP), which arises when agents must connect their actions with delayed or long-term consequences. Overcoming this challenge is critical for applying RL effectively in real-world environments, where feedback is often noisy, arrives with delays, or lacks clear causal information [380]. Incorporating attention mechanisms has been suggested as a promising direction for advancing solutions to the credit assignment problem [381-383].

Multi-Agent Reinforcement Learning (MARL) enables agents to learn through trial and error how to optimize cumulative rewards. Accurately characterizing and modeling the dynamics of multi-agent systems is vital for real-world applications, including autonomous driving and cooperative gaming. Recent work has shown that attention mechanisms can be employed [384-386] to better represent interactions among agents. In particular, attention has been used to capture how multi-agent behaviors emerge, as well as to identify distinct agent groups and the ways in which they coordinate with one another.

7.7. Auto Learning Attention

Neural architecture search (NAS) has recently demonstrated superior performance over manually designed architectures across a variety of tasks [387]. An important question is whether NAS can also be leveraged to discover effective architectures for higher-order attention mechanisms. To explore this, Ma et al. [388] introduced Higher-Order Group Attention (HOGA), a module designed through automated search. HOGA is represented as a Directed Acyclic Graph (DAG), in which groups correspond to nodes and edges denote heterogeneous attention operations. This formulation makes it possible to capture higher-order attention patterns, while a differentiable search process is employed to efficiently determine the most suitable configuration.

8. Conclusion

This study offeres a comprehensive analysis of attention mechanisms, beginning with an overview of their foundational concepts and extending to a review of diverse variants, including Hierarchical, Bidirectional, Multi-Head, Multi-query, Group-query, Graph, Channel, Spatial, Channel-Spatial, Temporal, Spatial-Temporal, Cross, Axial and Flash Attention. Recognizing the pivotal role of the Transformer architecture in advancing deep learning, we examined its core framework and surveyed a wide range of Transformer variants including BERT, GPT, Transformer XL, XLNet, BART, Fast Transformer, T5, Longformer, BIGBIRD, Performer, Linformer, Reformer, RoFormer, ALiBi, Switch Transformer, LLaMA, ViT, DETR, DeepViT, DeiT, T2T ViT, CrossViT, PVT, Swin Transformer, TNT, MViT, ViViT, DAT, and Spiking Transformer, each developed to address specific domain challenges. In addition, we emphasized the applications of attention mechanisms and Transformer-based approaches across a wide range of domains including natural language processing, computer vision, recommender systems, and sensor data analysis. Lastly, we offered a comparative overview of these models, outlining their fundamental ideas, outcomes, advantages, and drawbacks. Through this survey, we underscored how attention-based models, particularly those built upon the Transformer architecture, have become foundational across diverse application domains. The increasing diversity of attention mechanisms and Transformer variants highlights their adaptability and effectiveness in capturing complex dependencies and contextual relationships within data.

References

[1] X. Li *et al.*, "Deep learning attention mechanism in medical image analysis: Basics and beyonds," *International Journal of Network Dynamics and Intelligence,* pp. 93-116, 2023.

[2] F. M. Shiri, T. Perumal, N. Mustapha, R. Mohamed, M. A. B. Ahmadon, and S. Yamaguchi, "Measuring Student Satisfaction Based on Analysis of Physical Parameters in Smart Classroom," in *2024 12th International Conference on Information and Education Technology (ICIET)*, 2024: IEEE, pp. 18-23, doi: https://doi.org/10.1109/ICIET60671.2024.10542750.

[3] H. R. Taheri, M. Shakeri, and S. A. Zahedi, "Enhancing electrical conductivity and mechanical strength of gas diffusion layers through multi-objective optimization," *Journal of Power Sources,* vol. 658, p. 238358, 2025/12/01/ 2025, doi: https://doi.org/10.1016/j.jpowsour.2025.238358.

[4] L. Alzubaidi *et al.*, "A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications," *J. Big. Data.,* vol. 10, no. 1, p. 46, 2023, doi: https://doi.org/10.1186/s40537-023-00727-2.

[5] F. M. P. Shiri, T. Perumal, N. Mustapha, R. Mohamed, M. A. Ahmadon, and S. Yamaguchi, "A Survey on Multi-Resident Activity Recognition in Smart Environments," *Evolution of Information, Communication and Computing System,* pp. 12-27, 2023.

[6] A. Hernández and J. M. Amigó, "Attention mechanisms and their applications to complex systems," *Entropy,* vol. 23, no. 3, p. 283, 2021.

[7] G. W. Lindsay, "Attention in psychology, neuroscience, and machine learning," *Frontiers in computational neuroscience,* vol. 14, p. 516985, 2020.

[8] M. Hassanin, S. Anwar, I. Radwan, F. S. Khan, and A. Mian, "Visual attention methods in deep learning: An in-depth survey," *Information Fusion,* vol. 108, p. 102417, 2024.

[9] Z. Niu, G. Zhong, and H. Yu, "A review on the attention mechanism of deep learning," *Neurocomputing,* vol. 452, pp. 48-62, 2021.

[10] H. Larochelle and G. E. Hinton, "Learning to combine foveal glimpses with a third-order Boltzmann machine," *Advances in neural information processing systems,* vol. 23, 2010.

[11] L. Bazzani, H. Larochelle, V. Murino, J.-a. Ting, and N. D. Freitas, "Learning attentional policies for tracking and recognition in video with deep networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 937-944.

[12] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," *Advances in neural information processing systems,* vol. 27, 2014.

[13] M. F. Stollenga, J. Masci, F. Gomez, and J. Schmidhuber, "Deep networks with internal selective attention through feedback connections," *Advances in neural information processing systems,* vol. 27, 2014.

[14] C. Tang, N. Srivastava, and R. R. Salakhutdinov, "Learning generative models with visual attention," *Advances in Neural Information Processing Systems,* vol. 27, 2014.

[15] X. Yang, "An overview of the attention mechanisms in computer vision," in *Journal of physics: Conference series*, 2020, vol. 1693, no. 1: IOP Publishing, p. 012173.

[16] G. Brauwers and F. Frasincar, "A general survey on attention mechanisms in deep learning," *IEEE Transactions on Knowledge and Data Engineering,* vol. 35, no. 4, pp. 3279-3298, 2021.

[17] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473,* 2014.

[18] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025,* 2015.

[19] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," *arXiv preprint arXiv:1601.06733,* 2016.

[20] A. Vaswani *et al.*, "Attention is all you need," in *31st int. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 5998–6008.

[21] F. M. Shiri, T. Perumal, N. Mustapha, and R. Mohamed, "A Comprehensive Overview and Comparative Analysis on Deep Learning Models," *Journal on Artificial Intelligence,* vol. 6, no. 1, pp. 301-360, 2024, doi: https://doi.org/10.32604/jai.2024.054314.

[22] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929,* 2020.

[23] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*, 2021: PMLR, pp. 10347-10357.

[24] Y. Wang, Y. Deng, Y. Zheng, P. Chattopadhyay, and L. Wang, "Vision transformers for image classification: A comparative survey," *Technologies,* vol. 13, no. 1, p. 32, 2025.

[25] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "Vivit: A video vision transformer," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 6836-6846.

[26] Z. Liu *et al.*, "Video swin transformer," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 3202-3211.

[27] Y. Liu *et al.*, "Lovit: Long video transformer for surgical phase recognition," *Medical Image Analysis,* vol. 99, p. 103366, 2025.

[28] F. Sun *et al.*, "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proceedings of the 28th ACM international conference on information and knowledge management*, 2019, pp. 1441-1450.

[29] J. Zou, E. Kanoulas, P. Ren, Z. Ren, A. Sun, and C. Long, "Improving conversational recommender systems via transformer-based sequential modelling," in *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, 2022, pp. 2319-2324.

[30] J. W. Kim, A. U. Khan, and I. Banerjee, "Systematic review of hybrid vision transformer architectures for radiological image analysis," *Journal of Imaging Informatics in Medicine,* pp. 1-15, 2025.

[31] H. Kheddar, "Transformers and large language models for efficient intrusion detection systems: A comprehensive survey," *Information Fusion,* vol. 124, p. 103347, 2025.

[32] H. Lv, J. Chen, T. Pan, T. Zhang, Y. Feng, and S. Liu, "Attention mechanism in intelligent fault diagnosis of machinery: A review of technique and application," *Measurement,* vol. 199, p. 111594, 2022.

[33] I. A. Albadarneh, B. H. Hammo, and O. S. Al-Kadi, "Attention-based transformer models for image captioning across languages: An in-depth survey and evaluation," *Computer Science Review,* vol. 58, p. 100766, 2025.

[34] Y. O. Sharrab, H. Attar, M. A. H. Eljinini, Y. Al-Omary, and W. A. E. Al-Momani, "Advancements in speech recognition: A systematic review of deep learning transformer models, trends, innovations, and future directions," *IEEE Access,* vol. 13, pp. 46925-46940, 2025.

[35] U. Hussan, H. Wang, J. Peng, H. Jiang, and H. Rasheed, "Transformer-based renewable energy forecasting: A comprehensive review," *Renewable and Sustainable Energy Reviews,* vol. 226, p. 116356, 2026.

[36] M.-H. Guo *et al.*, "Attention mechanisms in computer vision: A survey," *Computational visual media,* vol. 8, no. 3, pp. 331-368, 2022.

[37] D. Soydaner, "Attention mechanism in neural networks: where it comes and where it goes," *Neural Computing and Applications,* vol. 34, no. 16, pp. 13371-13385, 2022.

[38] S. Chaudhari, V. Mithal, G. Polatkan, and R. Ramanath, "An attentive survey of attention models," *ACM Transactions on Intelligent Systems and Technology (TIST),* vol. 12, no. 5, pp. 1-32, 2021.

[39] A. de Santana Correia and E. L. Colombini, "Attention, please! A survey of neural attention models in deep learning," *Artificial Intelligence Review,* vol. 55, no. 8, pp. 6037-6124, 2022.

[40] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259,* 2014.

[41] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850,* 2013.

[42] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016, pp. 1480-1489.

[43] F. Mortezapour Shiri, S. Yamaguchi, and M. A. B. Ahmadon, "A Deep Learning Model Based on Bidirectional Temporal Convolutional Network (Bi-TCN) for Predicting Employee Attrition," *Applied Sciences,* vol. 15, no. 6, p. 2984, 2025, doi: https://doi.org/10.3390/app15062984.

[44] H. Ying *et al.*, "Sequential recommender system based on hierarchical attention network," in *IJCAI international joint conference on artificial intelligence*, 2018.

[45] Y. Cheng, F. Huang, L. Zhou, C. Jin, Y. Zhang, and T. Zhang, "A hierarchical multimodal attention-based neural network for image captioning," in *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, 2017, pp. 889-892.

[46] Q. Wang and A. B. Chan, "Gated hierarchical attention for image captioning," in *Asian Conference on Computer Vision*, 2018: Springer, pp. 21-37.

[47] W. Wang, Z. Chen, and H. Hu, "Hierarchical attention network for image captioning," in *Proceedings of the AAAI conference on artificial intelligence*, 2019, vol. 33, no. 01, pp. 8957-8964.

[48] S. Yan, Y. Xie, F. Wu, J. S. Smith, W. Lu, and B. Zhang, "Image captioning via hierarchical attention mechanism and policy gradient optimization," *Signal Processing,* vol. 167, p. 107329, 2020.

[49] Y. Ming, N. Hu, C. Fan, F. Feng, J. Zhou, and H. Yu, "Visuals to text: A comprehensive review on automatic image captioning," *IEEE/CAA Journal of Automatica Sinica,* vol. 9, no. 8, pp. 1339-1365, 2022.

[50] C. Liu, Y. Liu, Y. Yan, and J. Wang, "An intrusion detection model with hierarchical attention mechanism," *IEEE Access,* vol. 8, pp. 67542-67554, 2020.

[51] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," *arXiv preprint arXiv:1611.01603,* 2016.

[52] L. Liu, J. Liu, and J. Han, "Multi-head or single-head? an empirical comparison for transformer training," *arXiv preprint arXiv:2106.09650,* 2021.

[53] J. Li, Z. Tu, B. Yang, M. R. Lyu, and T. Zhang, "Multi-head attention with disagreement regularization," *arXiv preprint arXiv:1810.10183,* 2018.

[54] P. Liang, B. Taskar, and D. Klein, "Alignment by agreement," in *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, 2006, pp. 104-111.

[55] Y. Cheng *et al.*, "Agreement-based joint training for bidirectional attention-based neural machine translation," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016, pp. 2761-2767.

[56] J.-B. Cordonnier, A. Loukas, and M. Jaggi, "Multi-head attention: Collaborate instead of concatenate," *arXiv preprint arXiv:2006.16362,* 2020.

[57] N. Shazeer, "Fast transformer decoding: One write-head is all you need," *arXiv preprint arXiv:1911.02150,* 2019.

[58] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai, "Gqa: Training generalized multi-query transformer models from multi-head checkpoints," *arXiv preprint arXiv:2305.13245,* 2023.

[59] R. Pope *et al.*, "Efficiently scaling transformer inference," *Proceedings of Machine Learning and Systems,* vol. 5, pp. 606-624, 2023.

[60] Y. Chen, C. Zhang, X. Gao, R. D. Mullins, G. A. Constantinides, and Y. Zhao, "Optimised grouped-query attention mechanism for transformers," *arXiv preprint arXiv:2406.14963,* 2024.

[61] V. Joshi, P. Laddha, S. Sinha, O. J. Omer, and S. Subramoney, "QCQA: Quality and Capacity-aware grouped Query Attention," *arXiv preprint arXiv:2406.10247,* 2024.

[62] S. S. Chinnakonduru and A. Mohapatra, "Weighted Grouped Query Attention in Transformers," *arXiv preprint arXiv:2407.10855,* 2024.

[63] Z. Khan, M. Khaquan, O. Tafveez, B. Samiwala, and A. A. Raza, "Beyond Uniform Query Distribution: Key-Driven Grouped Query Attention," *arXiv preprint arXiv:2408.08454,* 2024.

[64] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems,* vol. 32, no. 1, pp. 4-24, 2020.

[65] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE international joint conference on neural networks, 2005.*, 2005, vol. 2: IEEE, pp. 729-734.

[66] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks,* vol. 20, no. 1, pp. 61-80, 2008.

[67] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493,* 2015.

[68] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555,* 2014.

[69] F. M. Shiri, T. Perumal, N. Mustapha, R. Mohamed, M. A. B. Ahmadon, and S. Yamaguchi, "Recognition of Student Engagement and Affective States Using ConvNeXtlarge and Ensemble GRU in E-Learning," in *2024 12th Int. Conf. Inf. edu. Technol. (ICIET)*, Yamaguchi, Japan, 18-20 March 2024: IEEE, pp. 30-34, doi: https://doi.org/10.1109/ICIET60671.2024.10542707.

[70] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903,* 2017.

[71] A. G. Vrahatis, K. Lazaros, and S. Kotsiantis, "Graph attention networks: a comprehensive review of methods and applications," *Future Internet,* vol. 16, no. 9, p. 318, 2024.

[72] Y. Sun and J. Han, "Mining heterogeneous information networks: a structural analysis approach," *ACM SIGKDD explorations newsletter,* vol. 14, no. 2, pp. 20-28, 2013.

[73] X. Wang *et al.*, "Heterogeneous graph attention network," in *The world wide web conference*, 2019, pp. 2022-2032.

[74] K. Wang, W. Shen, Y. Yang, X. Quan, and R. Wang, "Relational graph attention network for aspect-based sentiment analysis," *arXiv preprint arXiv:2004.12362,* 2020.

[75] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?," *arXiv preprint arXiv:2105.14491,* 2021.

[76] M. Yu, J. Shi, C. Xue, X. Hao, and G. Yan, "A review of single image super-resolution reconstruction based on deep learning," *Multimedia Tools and Applications,* vol. 83, no. 18, pp. 55921-55962, 2024.

[77] A. Ahmadyfard, "A Comprehensive Survey of Channel Attention Mechanisms in Single Image Super-Resolution," *J. Electrical Systems,* vol. 20, no. 3, pp. 9571-9583, 2024.

[78] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern. Recognit.*, 2018, pp. 7132-7141.

[79] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3-19.

[80] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 286-301.

[81] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng, "A^ 2-nets: Double attention networks," *Advances in neural information processing systems,* vol. 31, 2018.

[82] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang, "Second-order attention network for single image super-resolution," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 11065-11074.

[83] Z. Gao, J. Xie, Q. Wang, and P. Li, "Global second-order pooling convolutional networks," in *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, 2019, pp. 3024-3033.

[84] J. Fu *et al.*, "Dual attention network for scene segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3146-3154.

[85] Z. Yang, L. Zhu, Y. Wu, and Y. Yang, "Gated channel transformation for visual recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11794-11803.

[86] H. Lee, H.-E. Kim, and H. Nam, "Srm: A style-based recalibration module for convolutional neural networks," in *Proceedings of the IEEE/CVF International conference on computer vision*, 2019, pp. 1854-1862.

[87] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, "ECA-Net: Efficient channel attention for deep convolutional neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11534-11542.
[88] B. Niu *et al.*, "Single image super-resolution via a holistic attention network," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*, 2020: Springer, pp. 191-207.
[89] Z. Qin, P. Zhang, F. Wu, and X. Li, "Fcanet: Frequency channel attention networks," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 783-792.
[90] H. Zhang *et al.*, "Resnest: Split-attention networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 2736-2746.
[91] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, 2014: Springer, pp. 818-833.
[92] P. Li, J. Xie, Q. Wang, and W. Zuo, "Is second-order information helpful for large-scale visual recognition?," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2070-2078.
[93] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear CNN models for fine-grained visual recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1449-1457.
[94] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," *arXiv preprint arXiv:1612.03928,* 2016.
[95] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "Ccnet: Criss-cross attention for semantic segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 603-612.
[96] X. Hu, Z. Zhang, Z. Jiang, S. Chaudhuri, Z. Yang, and R. Nevatia, "SPAN: Spatial pyramid attention network for image manipulation localization," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*, 2020: Springer, pp. 312-328.
[97] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146,* 2016.
[98] Y. Wu, W. AbdAlmageed, and P. Natarajan, "Mantra-net: Manipulation tracing network for detection and localization of image forgeries with anomalous features," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 9543-9552.
[99] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis, "Learning rich features for image manipulation detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1053-1061.
[100] B. Bayar and M. C. Stamm, "A deep learning approach to universal image manipulation detection using a new convolutional layer," in *Proceedings of the 4th ACM workshop on information hiding and multimedia security*, 2016, pp. 5-10.
[101] Z. Meng, J. Ma, and X. Yuan, "End-to-end low cost compressive spectral imaging with spatial-spectral self-attention," in *European conference on computer vision*, 2020: Springer, pp. 187-204.
[102] L. Chen *et al.*, "Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5659-5667.
[103] F. Wang *et al.*, "Residual attention network for image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3156-3164.
[104] A. G. Roy, N. Navab, and C. Wachinger, "Concurrent spatial and channel 'squeeze & excitation'in fully convolutional networks," in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part I*, 2018: Springer, pp. 421-429.
[105] J. Park, S. Woo, J.-Y. Lee, and I. S. Kweon, "Bam: Bottleneck attention module," *arXiv preprint arXiv:1807.06514,* 2018.
[106] L. Mou *et al.*, "CS-Net: Channel and spatial attention network for curvilinear structure segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2019: Springer, pp. 721-730.
[107] Y. Huang, D. Kang, W. Jia, L. Liu, and X. He, "Channelized axial attention–considering channel relation within spatial attention for semantic segmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022, vol. 36, no. 1, pp. 1016-1025.
[108] Z. Zhang, C. Lan, W. Zeng, X. Jin, and Z. Chen, "Relation-aware global attention for person re-identification," in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2020, pp. 3186-3195.
[109] Q. Hou, D. Zhou, and J. Feng, "Coordinate attention for efficient mobile network design," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 13713-13722.
[110] C. H. Song, H. J. Han, and Y. Avrithis, "All the attention you need: Global-local, spatial-channel attention for image retrieval," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2022, pp. 2754-2763.
[111] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.
[112] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," *Advances in neural information processing systems,* vol. 28, 2015.
[113] Q. Liu *et al.*, "FUSCANet: Enhancing Skin Disease Classification through Feature Fusion and Spatial-Channel Attention Mechanisms," *IEEE Access,* 2025.
[114] J. Li, J. Wang, Q. Tian, W. Gao, and S. Zhang, "Global-local temporal representations for video person re-identification," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3958-3967.
[115] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122,* 2015.
[116] Z. Liu, L. Wang, W. Wu, C. Qian, and T. Lu, "Tam: Temporal adaptive module for video recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 13708-13718.
[117] D. Cui, C. Xin, L. Wu, and X. Wang, "ConvTransformer Attention Network for temporal action detection," *Knowledge-Based Systems,* vol. 300, p. 112264, 2024.
[118] H. Chen and Z. Shi, "A spatial-temporal attention-based method and a new dataset for remote sensing image change detection," *Remote sensing,* vol. 12, no. 10, p. 1662, 2020.
[119] C. Tian and W. K. Chan, "Spatial-temporal attention wavenet: A deep learning framework for traffic prediction considering spatial-temporal dependencies," *IET Intelligent Transport Systems,* vol. 15, no. 4, pp. 549-561, 2021.

[120] C.-T. Lin, Y.-K. Wang, P.-L. Huang, Y. Shi, and Y.-C. Chang, "Spatial-temporal attention-based convolutional network with text and numerical information for stock price prediction," *Neural Computing and Applications,* vol. 34, no. 17, pp. 14387-14395, 2022.
[121] J. Fan, K. Zhang, Y. Huang, Y. Zhu, and B. Chen, "Parallel spatio-temporal attention-based TCN for multivariate time series prediction," *Neural Computing and Applications,* vol. 35, no. 18, pp. 13109-13118, 2023.
[122] M. Qiao *et al.*, "KSTAGE: A knowledge-guided spatial-temporal attention graph learning network for crop yield prediction," *Information Sciences,* vol. 619, pp. 19-37, 2023.
[123] M. Awais *et al.*, "Short-term photovoltaic energy generation for solar powered high efficiency irrigation systems using LSTM with Spatio-temporal attention mechanism," *Scientific Reports,* vol. 14, no. 1, p. 10042, 2024.
[124] X. Chen, Y. Hu, F. Dong, K. Chen, and H. Xia, "A multi-graph spatial-temporal attention network for air-quality prediction," *Process Safety and Environmental Protection,* vol. 181, pp. 442-451, 2024.
[125] Y. Pan *et al.*, "Spatial–temporal attention network for depression recognition from facial videos," *Expert systems with applications,* vol. 237, p. 121410, 2024.
[126] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, "An end-to-end spatio-temporal attention model for human action recognition from skeleton data," in *Proceedings of the AAAI conference on artificial intelligence*, 2017, vol. 31, no. 1.
[127] C. Yan *et al.*, "STAT: Spatial-temporal attention mechanism for video captioning," *IEEE transactions on multimedia,* vol. 22, no. 1, pp. 229-241, 2019.
[128] Q. Liu, X. Che, and M. Bie, "R-STAN: Residual spatial-temporal attention network for action recognition," *IEEE Access,* vol. 7, pp. 82246-82255, 2019.
[129] K.-H. Lee, X. Chen, G. Hua, H. Hu, and X. He, "Stacked cross attention for image-text matching," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 201-216.
[130] P. O. Pinheiro and R. Collobert, "From image-level to pixel-level labeling with convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1713-1721.
[131] R. Hou, H. Chang, B. Ma, S. Shan, and X. Chen, "Cross attention network for few-shot classification," *Advances in neural information processing systems,* vol. 32, 2019.
[132] J. Ho, N. Kalchbrenner, D. Weissenborn, and T. Salimans, "Axial attention in multidimensional transformers," *arXiv preprint arXiv:1912.12180,* 2019.
[133] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," *Advances in Neural Information Processing Systems,* vol. 35, pp. 16344-16359, 2022.
[134] Z. Jia, M. Maggioni, B. Staiger, and D. P. Scarpazza, "Dissecting the NVIDIA volta GPU architecture via microbenchmarking," *arXiv preprint arXiv:1804.06826,* 2018.
[135] R. Dolbeau, "Theoretical peak FLOPS per instruction set: A tutorial," *The Journal of Supercomputing,* vol. 74, no. 3, pp. 1341-1377, 2018.
[136] R. Garg and L. Hendren, "A portable and high-performance general matrix-multiply (GEMM) library for GPUs and single-chip CPU/GPU systems," in *2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 2014: IEEE, pp. 672-680.
[137] T. Dao, "Flashattention-2: Faster attention with better parallelism and work partitioning," *arXiv preprint arXiv:2307.08691,* 2023.
[138] J. Shah, G. Bikshandi, Y. Zhang, V. Thakkar, P. Ramani, and T. Dao, "Flashattention-3: Fast and accurate attention with asynchrony and low-precision," *Advances in Neural Information Processing Systems,* vol. 37, pp. 68658-68685, 2024.
[139] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan, "Training deep neural networks with 8-bit floating point numbers," *Advances in neural information processing systems,* vol. 31, 2018.
[140] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern. Recognit.*, 2016, pp. 770-778.
[141] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450,* 2016.
[142] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," *ACM Comput. Surv.,* vol. 55, no. 6, pp. 1-28, 2022, doi: https://doi.org/10.24963/ijcai.2023/764.
[143] Y. Liu and L. Wu, "Intrusion Detection Model Based on Improved Transformer," *Appl. Sci.,* vol. 13, no. 10, p. 6251, 2023, doi: https://doi.org/10.3390/app13106251.
[144] F. M. Shiri, T. Perumal, N. Mustapha, and R. Mohamed, "Deep Learning and Federated Learning in Human Activity Recognition with Sensor Data: A Comprehensive Review," *Computer Modeling in Engineering \& Sciences,* vol. 145, no. 2, 2025, doi: https://doi.org/10.32604/cmes.2025.071858.
[145] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805,* 2018.
[146] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144,* 2016.
[147] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942,* 2019.
[148] X. Jiao *et al.*, "Tinybert: Distilling bert for natural language understanding," *arXiv preprint arXiv:1909.10351,* 2019.
[149] S. Goyal, A. R. Choudhury, S. Raje, V. Chakaravarthy, Y. Sabharwal, and A. Verma, "Power-bert: Accelerating bert inference via progressive word-vector elimination," in *International Conference on Machine Learning*, 2020: PMLR, pp. 3690-3699.
[150] Y. Liu *et al.*, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692,* 2019.
[151] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909,* 2015.
[152] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "Spanbert: Improving pre-training by representing and predicting spans," *Transactions of the association for computational linguistics,* vol. 8, pp. 64-77, 2020.

[153] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108,* 2019.

[154] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531,* 2015.

[155] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," *arXiv preprint arXiv:2006.03654,* 2020.

[156] Y. Yang, M. C. S. Uy, and A. Huang, "Finbert: A pretrained language model for financial communications," *arXiv preprint arXiv:2006.08097,* 2020.

[157] S. Ji, T. Zhang, L. Ansari, J. Fu, P. Tiwari, and E. Cambria, "Mentalbert: Publicly available pretrained language models for mental healthcare," *arXiv preprint arXiv:2110.15621,* 2021.

[158] A. Adhikari, A. Ram, R. Tang, and J. Lin, "Docbert: Bert for document classification," *arXiv preprint arXiv:1904.08398,* 2019.

[159] F. Souza, R. Nogueira, and R. Lotufo, "BERTimbau: pretrained BERT models for Brazilian Portuguese," in *Brazilian conference on intelligent systems*, 2020: Springer, pp. 403-417.

[160] N. Webersinke, M. Kraus, J. A. Bingler, and M. Leippold, "Climatebert: A pretrained language model for climate-related text," *arXiv preprint arXiv:2110.12010,* 2021.

[161] P. Prakash, S. Chilukuri, N. Ranade, and S. Viswanathan, "RareBERT: transformer architecture for rare disease patient identification using administrative claims," in *Proceedings of the AAAI conference on artificial intelligence*, 2021, vol. 35, no. 1, pp. 453-460.

[162] J. Lee *et al.*, "BioBERT: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics,* vol. 36, no. 4, pp. 1234-1240, 2020.

[163] H. Niu *et al.*, "EHR-BERT: A BERT-based model for effective anomaly detection in electronic health records," *Journal of Biomedical Informatics,* vol. 150, p. 104605, 2024.

[164] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[165] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog,* vol. 1, no. 8, p. 9, 2019.

[166] T. Brown *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems,* vol. 33, pp. 1877-1901, 2020.

[167] J. Achiam *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774,* 2023.

[168] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," in *Proceedings of the 57th annual meeting of the association for computational linguistics*, 2019, pp. 2978-2988.

[169] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," in *33rd Conf. Neural Inf. Process. Syst.*, Vancouver, Canada, Dec. 2019, pp. 5754–5764.

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

[171] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415,* 2016.

[172] C. Raffel *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research,* vol. 21, no. 140, pp. 1-67, 2020.

[173] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research,* vol. 15, no. 1, pp. 1929-1958, 2014.

[174] M. Wang, P. Xie, Y. Du, and X. Hu, "T5-based model for abstractive summarization: A semi-supervised learning approach with consistency loss functions," *Applied Sciences,* vol. 13, no. 12, p. 7111, 2023.

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

[176] A. Van Den Oord *et al.*, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499,* vol. 12, p. 1, 2016.

[177] M. Zaheer *et al.*, "Big bird: Transformers for longer sequences," *Advances in neural information processing systems,* vol. 33, pp. 17283-17297, 2020.

[178] D. A. Spielman and S.-H. Teng, "Spectral sparsification of graphs," *SIAM Journal on Computing,* vol. 40, no. 4, pp. 981-1025, 2011.

[179] K. M. Hermann *et al.*, "Teaching machines to read and comprehend," *Advances in neural information processing systems,* vol. 28, 2015.

[180] K. Choromanski *et al.*, "Rethinking attention with performers," *arXiv preprint arXiv:2009.14794,* 2020.

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

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

[183] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, "RoFormer: Enhanced Transformer with Rotary Position Embedding," *arXiv preprint arXiv:2104.09864,* 2021.

[184] B. Heo, S. Park, D. Han, and S. Yun, "Rotary position embedding for vision transformer," in *European Conference on Computer Vision*, 2024: Springer, pp. 289-305.

[185] O. Press, N. A. Smith, and M. Lewis, "Train short, test long: Attention with linear biases enables input length extrapolation," *arXiv preprint arXiv:2108.12409,* 2021.

[186] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," *arXiv preprint arXiv:1803.02155,* 2018.

[187] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *J. Mach. Learn. Res. ,* vol. 23, no. 120, pp. 1-39, 2022.

[188] Z. Chen, Y. Deng, Y. Wu, Q. Gu, and Y. Li, "Towards understanding the mixture-of-experts layer in deep learning," *Advances in neural information processing systems,* vol. 35, pp. 23049-23062, 2022.

[189] N. Shazeer *et al.*, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint arXiv:1701.06538,* 2017.

[190] H. Touvron *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971,* 2023.

[191] B. Zhang and R. Sennrich, "Root mean square layer normalization," *Advances in neural information processing systems,* vol. 32, 2019.

[192] A. Chowdhery *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research,* vol. 24, no. 240, pp. 1-113, 2023.
[193] N. Shazeer, "Glu variants improve transformer," *arXiv preprint arXiv:2002.05202,* 2020.
[194] S. Black *et al.*, "Gpt-neox-20b: An open-source autoregressive language model," *arXiv preprint arXiv:2204.06745,* 2022.
[195] M. N. Rabe and C. Staats, "Self-attention does not need $ O (n^ 2) $ memory," *arXiv preprint arXiv:2112.05682,* 2021.
[196] Q. Wang *et al.*, "Learning deep transformer models for machine translation," *arXiv preprint arXiv:1906.01787,* 2019.
[197] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*, 2020: Springer, pp. 213-229.
[198] D. Zhou *et al.*, "Deepvit: Towards deeper vision transformer," *arXiv preprint arXiv:2103.11886,* 2021.
[199] L. Yuan *et al.*, "Tokens-to-token vit: Training vision transformers from scratch on imagenet," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.* , 2021, pp. 558-567.
[200] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, 2009: Ieee, pp. 248-255.
[201] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861,* 2017.
[202] C.-F. R. Chen, Q. Fan, and R. Panda, "Crossvit: Cross-attention multi-scale vision transformer for image classification," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 357-366.
[203] W. Wang *et al.*, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern. Recognit.*, 2021, pp. 568-578.
[204] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.* , 2021, pp. 10012-10022.
[205] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556,* 2014.
[206] K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," *Adv. Neural. Inf. Process. Syst.,* vol. 34, pp. 15908-15919, 2021.
[207] K. Han, J. Guo, Y. Tang, and Y. Wang, "Pyramidtnt: Improved transformer-in-transformer baselines with pyramid architecture," *arXiv preprint arXiv:2201.00978,* 2022.
[208] H. Fan *et al.*, "Multiscale vision transformers," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 6824-6835.
[209] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid, "ViViT: A Video Vision Transformer," *arXiv preprint arXiv:2103.15691,* 2021.
[210] Z. Xia, X. Pan, S. Song, L. E. Li, and G. Huang, "Vision transformer with deformable attention," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 4794-4803.
[211] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature,* vol. 575, no. 7784, pp. 607-617, 2019.
[212] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in neuroscience,* vol. 12, p. 331, 2018.
[213] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 2661-2671.
[214] J. C. Whittington, J. Warren, and T. E. Behrens, "Relating transformers to models and neural representations of the hippocampal formation," *arXiv preprint arXiv:2112.04035,* 2021.
[215] C. Caucheteux and J.-R. King, "Brains and algorithms partially converge in natural language processing," *Communications biology,* vol. 5, no. 1, p. 134, 2022.
[216] Z. Zhou *et al.*, "Spikformer: When spiking neural network meets transformer," *arXiv preprint arXiv:2209.15425,* 2022.
[217] X. Chu *et al.*, "Twins: Revisiting the design of spatial attention in vision transformers," *Advances in neural information processing systems,* vol. 34, pp. 9355-9366, 2021.
[218] M. Yao *et al.*, "Spike-driven transformer," *Advances in neural information processing systems,* vol. 36, pp. 64043-64058, 2023.
[219] Y. Hu, L. Deng, Y. Wu, M. Yao, and G. Li, "Advancing spiking neural networks toward deep residual learning," *IEEE transactions on neural networks and learning systems,* vol. 36, no. 2, pp. 2353-2367, 2024.
[220] C. Zhou *et al.*, "Spikingformer: Spike-driven residual learning for transformer-based spiking neural network," *arXiv preprint arXiv:2304.11954,* 2023.
[221] C. Zhou *et al.*, "Qkformer: Hierarchical spiking transformer using qk attention," *Advances in Neural Information Processing Systems,* vol. 37, pp. 13074-13098, 2024.
[222] M. Yao *et al.*, "Spike-driven transformer v2: Meta spiking neural network architecture inspiring the design of next-generation neuromorphic chips," *arXiv preprint arXiv:2404.03663,* 2024.
[223] X. Shi, Z. Hao, and Z. Yu, "Spikingresformer: Bridging resnet and vision transformer in spiking neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024, pp. 5610-5619.
[224] Y. Fang, Z. Wang, L. Zhang, J. Cao, H. Chen, and R. Xu, "Spiking wavelet transformer," in *European conference on computer vision*, 2024: Springer, pp. 19-37.
[225] Z. Lei *et al.*, "Spike2former: Efficient spiking transformer for high-performance image segmentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025, vol. 39, no. 2, pp. 1364-1372.
[226] D. Lee, Y. Li, Y. Kim, S. Xiao, and P. Panda, "Spiking transformer with spatial-temporal attention," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 13948-13958.

[227] R. Patil and V. Gudivada, "A review of current trends, techniques, and challenges in large language models (llms)," *Applied Sciences,* vol. 14, no. 5, p. 2074, 2024.
[228] M. India, P. Safari, and J. Hernando, "Self multi-head attention for speaker recognition," *arXiv preprint arXiv:1906.09890,* 2019.
[229] Z. Lin *et al.*, "A structured self-attentive sentence embedding," *arXiv preprint arXiv:1703.03130,* 2017.
[230] S. Kardakis, I. Perikos, F. Grivokostopoulou, and I. Hatzilygeroudis, "Examining attention mechanisms in deep learning models for sentiment analysis," *Applied Sciences,* vol. 11, no. 9, p. 3883, 2021.
[231] N. Du *et al.*, "Glam: Efficient scaling of language models with mixture-of-experts," in *International conference on machine learning*, 2022: PMLR, pp. 5547-5569.
[232] J. W. Rae *et al.*, "Scaling language models: Methods, analysis & insights from training gopher," *arXiv preprint arXiv:2112.11446,* 2021.
[233] J. Hoffmann *et al.*, "Training compute-optimal large language models," *arXiv preprint arXiv:2203.15556,* 2022.
[234] R. Thoppilan *et al.*, "Lamda: Language models for dialog applications," *arXiv preprint arXiv:2201.08239,* 2022.
[235] H. Touvron *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288,* 2023.
[236] Y. Sun *et al.*, "Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation," *arXiv preprint arXiv:2107.02137,* 2021.
[237] M. Mager *et al.*, "GPT-too: A language-model-first approach for AMR-to-text generation," *arXiv preprint arXiv:2005.09123,* 2020.
[238] G. Bao, Y. Zhang, Z. Teng, B. Chen, and W. Luo, "G-transformer for document-level machine translation," *arXiv preprint arXiv:2105.14761,* 2021.
[239] H.-I. Liu and W.-L. Chen, "Re-transformer: a self-attention based model for machine translation," *Procedia Computer Science,* vol. 189, pp. 3-10, 2021.
[240] M. Gheini, X. Ren, and J. May, "Cross-attention is all you need: Adapting pretrained transformers for machine translation," *arXiv preprint arXiv:2104.08771,* 2021.
[241] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and Ł. Kaiser, "Universal transformers," *arXiv preprint arXiv:1807.03819,* 2018.
[242] K. Nassiri and M. Akhloufi, "Transformer models used for text-based question answering systems," *Applied Intelligence,* vol. 53, no. 9, pp. 10602-10635, 2023.
[243] W. Wang, M. Yan, and C. Wu, "Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering," *arXiv preprint arXiv:1811.11934,* 2018.
[244] C. Xing, Y. Wu, W. Wu, Y. Huang, and M. Zhou, "Hierarchical recurrent attention network for response generation," in *Proceedings of the AAAI conference on artificial intelligence*, 2018, vol. 32, no. 1.
[245] T. Yang, L. Hu, C. Shi, H. Ji, X. Li, and L. Nie, "HGAT: Heterogeneous graph attention networks for semi-supervised short text classification," *ACM Transactions on Information Systems (TOIS),* vol. 39, no. 3, pp. 1-29, 2021.
[246] Z. Dan, Y. Zhao, X. Bi, L. Wu, and Q. Ji, "Multi-task transformer with adaptive cross-entropy loss for multi-dialect speech recognition," *Entropy,* vol. 24, no. 10, p. 1429, 2022.
[247] C. Wang *et al.*, "Arobert: An asr robust pre-trained language model for spoken language understanding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing,* vol. 30, pp. 1207-1218, 2022.
[248] S. Kumar and A. Solanki, "An abstractive text summarization technique using transformer model with self-attention mechanism," *Neural Computing and Applications,* vol. 35, no. 25, pp. 18603-18622, 2023.
[249] P. J. Goutom, N. Baruah, and P. Sonowal, "Attention-based Transformer for Assamese Abstractive Text Summarization," *Procedia Computer Science,* vol. 235, pp. 1097-1104, 2024.
[250] T. Ma, Q. Pan, H. Rong, Y. Qian, Y. Tian, and N. Al-Nabhan, "T-bertsum: Topic-aware text summarization based on bert," *IEEE Transactions on Computational Social Systems,* vol. 9, no. 3, pp. 879-890, 2021.
[251] S. Abdel-Salam and A. Rafea, "Performance study on extractive text summarization using BERT models," *Information,* vol. 13, no. 2, p. 67, 2022.
[252] W. Ai, Y. Wei, H. Shao, Y. Shou, T. Meng, and K. Li, "Edge-enhanced minimum-margin graph attention network for short text classification," *Expert Systems with Applications,* vol. 251, p. 124069, 2024.
[253] S. K. Prabhakar and D.-O. Won, "Medical text classification using hybrid deep learning models with multihead attention," *Computational intelligence and neuroscience,* vol. 2021, no. 1, p. 9425655, 2021.
[254] B. Peng, T. Zhang, K. Han, Z. Zhang, Y. Ma, and M. Ma, "BVMHA: Text classification model with variable multihead hybrid attention based on BERT," *Journal of Intelligent & Fuzzy Systems,* vol. 46, no. 1, pp. 1443-1454, 2024.
[255] X. Zhang, L. Yu, and S. Tian, "BGAT: Aspect-based sentiment analysis based on bidirectional GRU and graph attention network," *Journal of Intelligent & Fuzzy Systems,* vol. 44, no. 2, pp. 3115-3126, 2023.
[256] X. Zhou, T. Zhang, C. Cheng, and S. Song, "Dynamic multichannel fusion mechanism based on a graph attention network and BERT for aspect-based sentiment classification," *Applied Intelligence,* vol. 53, no. 6, pp. 6800-6813, 2023.
[257] A. S. Talaat, "Sentiment analysis classification system using hybrid BERT models," *Journal of Big Data,* vol. 10, no. 1, p. 110, 2023.
[258] N. J. Prottasha *et al.*, "Transfer learning for sentiment analysis using BERT based supervised fine-tuning," *Sensors,* vol. 22, no. 11, p. 4157, 2022.
[259] K. L. Tan, C. P. Lee, K. S. M. Anbananthen, and K. M. Lim, "RoBERTa-LSTM: a hybrid model for sentiment analysis with transformer and recurrent neural network," *Ieee Access,* vol. 10, pp. 21517-21525, 2022.
[260] A. Bello, S.-C. Ng, and M.-F. Leung, "A BERT framework to sentiment analysis of tweets," *Sensors,* vol. 23, no. 1, p. 506, 2023.
[261] A. Sinha and J. Dolz, "Multi-scale self-guided attention for medical image segmentation," *IEEE journal of biomedical and health informatics,* vol. 25, no. 1, pp. 121-130, 2020.
[262] I. J. Goodfellow *et al.*, "Generative adversarial nets," *Advances in neural information processing systems,* vol. 27, 2014.
[263] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International conference on machine learning*, 2019: PMLR, pp. 7354-7363.
[264] W. Tong, W. Chen, W. Han, X. Li, and L. Wang, "Channel-attention-based DenseNet network for remote sensing image scene classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing,* vol. 13, pp. 4121-4132, 2020.

[265] A. Gomaa and O. M. Saad, "Residual Channel-attention (RCA) network for remote sensing image scene classification," *Multimedia Tools and Applications,* pp. 1-25, 2025.
[266] Z. Lu, B. Xu, L. Sun, T. Zhan, and S. Tang, "3-D channel and spatial attention based multiscale spatial–spectral residual network for hyperspectral image classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing,* vol. 13, pp. 4311-4324, 2020.
[267] J. Guo, N. Jia, and J. Bai, "Transformer based on channel-spatial attention for accurate classification of scenes in remote sensing image," *Scientific Reports,* vol. 12, no. 1, p. 15473, 2022.
[268] C. Zhuang, X. Yuan, L. Gu, Z. Wei, Y. Fan, and X. Guo, "Frequency Regulated Channel-Spatial Attention module for improved image classification," *Expert Systems with Applications,* vol. 260, p. 125463, 2025.
[269] Y. Bazi, L. Bashmal, M. M. A. Rahhal, R. A. Dayil, and N. A. Ajlan, "Vision transformers for remote sensing image classification," *Remote Sensing,* vol. 13, no. 3, p. 516, 2021.
[270] O. N. Manzari, H. Ahmadabadi, H. Kashiani, S. B. Shokouhi, and A. Ayatollahi, "MedViT: a robust vision transformer for generalized medical image classification," *Computers in biology and medicine,* vol. 157, p. 106791, 2023.
[271] S. Regmi, A. Subedi, N. K. Tomar, U. Bagci, and D. Jha, "Vision transformer for efficient chest x-ray and gastrointestinal image classification," in *Medical Imaging 2025: Computer-Aided Diagnosis*, 2025, vol. 13407: SPIE, pp. 912-923.
[272] O. Petit, N. Thome, C. Rambour, L. Themyr, T. Collins, and L. Soler, "U-net transformer: Self and cross attention for medical image segmentation," in *International Workshop on Machine Learning in Medical Imaging*, 2021: Springer, pp. 267-276.
[273] E. K. Aghdam, R. Azad, M. Zarvani, and D. Merhof, "Attention swin u-net: Cross-contextual attention mechanism for skin lesion segmentation," in *2023 IEEE 20th International Symposium on Biomedical Imaging (ISBI)*, 2023: IEEE, pp. 1-5.
[274] G. C. Ates, P. Mohan, and E. Celik, "Dual cross-attention for medical image segmentation," *Engineering Applications of Artificial Intelligence,* vol. 126, p. 107139, 2023.
[275] A. Ramachandran and S. K. KS, "Tiny Criss-Cross Network for segmenting paddy panicles using aerial images," *Computers and Electrical Engineering,* vol. 108, p. 108728, 2023.
[276] G. Sun *et al.*, "DA-TransUNet: integrating spatial and channel dual attention with transformer U-net for medical image segmentation," *Frontiers in Bioengineering and Biotechnology,* vol. 12, p. 1398237, 2024.
[277] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for semantic segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 7262-7272.
[278] J. Chen *et al.*, "Transunet: Transformers make strong encoders for medical image segmentation," *arXiv preprint arXiv:2102.04306,* 2021.
[279] J. Jain, J. Li, M. T. Chiu, A. Hassani, N. Orlov, and H. Shi, "Oneformer: One transformer to rule universal image segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 2989-2998.
[280] Z. Zhang, X. Lu, G. Cao, Y. Yang, L. Jiao, and F. Liu, "ViT-YOLO: Transformer-based YOLO for object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 2799-2808.
[281] Y. Li *et al.*, "Mvitv2: Improved multiscale vision transformers for classification and detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 4804-4814.
[282] L. Yu *et al.*, "SpikingViT: A multiscale spiking vision transformer model for event-based object detection," *IEEE Transactions on Cognitive and Developmental Systems,* vol. 17, no. 1, pp. 130-146, 2024.
[283] H. Lu, X. Chen, G. Zhang, Q. Zhou, Y. Ma, and Y. Zhao, "SCANet: Spatial-channel attention network for 3D object detection," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019: IEEE, pp. 1992-1996.
[284] Y. Cao, J. Bin, J. Hamari, E. Blasch, and Z. Liu, "Multimodal object detection by channel switching and spatial attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 403-411.
[285] J. Shen, Y. Chen, Y. Liu, X. Zuo, H. Fan, and W. Yang, "ICAFusion: Iterative cross-attention guided feature fusion for multispectral object detection," *Pattern Recognition,* vol. 145, p. 109913, 2024.
[286] Y. Pan, T. Yao, Y. Li, and T. Mei, "X-linear attention networks for image captioning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10971-10980.
[287] B. T. Nguyen, S. T. Nguyen, and A. H. Vo, "Channel and spatial attention mechanism for fashion image captioning," *International Journal of Electrical & Computer Engineering (2088-8708),* vol. 13, no. 5, 2023.
[288] L. Zheng, W. Xu, Z. Miao, X. Qiu, and S. Gong, "RESTHT: relation-enhanced spatial–temporal hierarchical transformer for video captioning," *The Visual Computer,* vol. 41, no. 1, pp. 591-604, 2025.
[289] L. Zhou, Y. Zhou, J. J. Corso, R. Socher, and C. Xiong, "End-to-end dense video captioning with masked transformer," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8739-8748.
[290] M. Cornia, M. Stefanini, L. Baraldi, and R. Cucchiara, "Meshed-memory transformer for image captioning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10578-10587.
[291] Y. Wang, J. Xu, and Y. Sun, "End-to-end transformer based model for image captioning," in *Proceedings of the AAAI conference on artificial intelligence*, 2022, vol. 36, no. 3, pp. 2585-2594.
[292] M. Ding *et al.*, "Cogview: Mastering text-to-image generation via transformers," *Advances in neural information processing systems,* vol. 34, pp. 19822-19835, 2021.
[293] B. Zhang *et al.*, "Styleswin: Transformer-based gan for high-resolution image generation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 11304-11314.
[294] H. Chang, H. Zhang, L. Jiang, C. Liu, and W. T. Freeman, "Maskgit: Masked generative image transformer," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 11315-11325.
[295] M. Ding, W. Zheng, W. Hong, and J. Tang, "Cogview2: Faster and better text-to-image generation via hierarchical transformers," *Advances in Neural Information Processing Systems,* vol. 35, pp. 16890-16902, 2022.
[296] H. Chang *et al.*, "Muse: Text-to-image generation via masked generative transformers," *arXiv preprint arXiv:2301.00704,* 2023.

[297] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, "Online multi-object tracking using CNN-based single object tracker with spatial-temporal attention mechanism," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 4836-4845.
[298] J. Zhu, H. Yang, N. Liu, M. Kim, W. Zhang, and M.-H. Yang, "Online multi-object tracking with dual matching attention networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 366-382.
[299] P. Chu, J. Wang, Q. You, H. Ling, and Z. Liu, "Transmot: Spatial-temporal graph transformer for multiple object tracking," in *Proceedings of the IEEE/CVF Winter Conference on applications of computer vision*, 2023, pp. 4870-4880.
[300] P. Sun *et al.*, "Transtrack: Multiple object tracking with transformer," *arXiv preprint arXiv:2012.15460,* 2020.
[301] F. Zeng, B. Dong, Y. Zhang, T. Wang, X. Zhang, and Y. Wei, "Motr: End-to-end multiple-object tracking with transformer," in *European conference on computer vision*, 2022: Springer, pp. 659-675.
[302] T. Meinhardt, A. Kirillov, L. Leal-Taixe, and C. Feichtenhofer, "Trackformer: Multi-object tracking with transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 8844-8854.
[303] G. Y. Gopal and M. A. Amer, "Separable self and mixed attention transformers for efficient object tracking," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2024, pp. 6708-6717.
[304] J. Li, X. Liu, W. Zhang, M. Zhang, J. Song, and N. Sebe, "Spatio-temporal attention networks for action recognition and detection," *IEEE Transactions on Multimedia,* vol. 22, no. 11, pp. 2990-3001, 2020.
[305] D. Ahn, S. Kim, H. Hong, and B. C. Ko, "Star-transformer: a spatio-temporal cross attention transformer for human action recognition," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2023, pp. 3330-3339.
[306] J. Yang, X. Dong, L. Liu, C. Zhang, J. Shen, and D. Yu, "Recurring the transformer for video action recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14063-14073.
[307] J. Chen and C. M. Ho, "Mm-vit: Multi-modal video transformer for compressed video action recognition," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2022, pp. 1910-1921.
[308] Z. Xing, Q. Dai, H. Hu, J. Chen, Z. Wu, and Y.-G. Jiang, "Svformer: Semi-supervised video transformer for action recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 18816-18826.
[309] L. Yu *et al.*, "Svformer: a direct training spiking transformer for efficient video action recognition," in *International Workshop on Human Brain and Artificial Intelligence*, 2024: Springer, pp. 161-180.
[310] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, "Session-based social recommendation via dynamic graph attention networks," in *Proceedings of the Twelfth ACM international conference on web search and data mining*, 2019, pp. 555-563.
[311] C. Xu *et al.*, "Graph contextualized self-attention network for session-based recommendation," in *IJCAI*, 2019, vol. 19, no. 2019, pp. 3940-3946.
[312] L. Wu, S. Li, C.-J. Hsieh, and J. Sharpnack, "SSE-PT: Sequential recommendation via personalized transformer," in *Proceedings of the 14th ACM conference on recommender systems*, 2020, pp. 328-337.
[313] J. Jiang *et al.*, "AdaMCT: adaptive mixture of CNN-transformer for sequential recommendation," in *Proceedings of the 32nd ACM international conference on information and knowledge management*, 2023, pp. 976-986.
[314] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "Kgat: Knowledge graph attention network for recommendation," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019, pp. 950-958.
[315] Z. Tao, Y. Wei, X. Wang, X. He, X. Huang, and T.-S. Chua, "Mgat: Multimodal graph attention network for recommendation," *Information Processing & Management,* vol. 57, no. 5, p. 102277, 2020.
[316] Y. Liu, S. Yang, Y. Xu, C. Miao, M. Wu, and J. Zhang, "Contextualized graph attention network for recommendation with item knowledge graph," *IEEE Transactions on knowledge and data engineering,* vol. 35, no. 1, pp. 181-195, 2021.
[317] A. Liang, "A Graph Attention-Based Recommendation Framework for Sparse User-Item Interactions," *Journal of Computer Science and Software Applications,* vol. 5, no. 4, 2025.
[318] Y. Shavit and I. Klein, "Boosting inertial-based human activity recognition with transformers," *IEEE Access,* vol. 9, pp. 53540-53547, 2021.
[319] I. Dirgová Luptáková, M. Kubovčík, and J. Pospíchal, "Wearable sensor-based human activity recognition with transformer model," *Sensors,* vol. 22, no. 5, p. 1911, 2022.
[320] S. Xiao, S. Wang, Z. Huang, Y. Wang, and H. Jiang, "Two-stream transformer network for sensor-based human activity recognition," *Neurocomputing,* vol. 512, pp. 253-268, 2022.
[321] W. Zhang, D. Yang, Y. Xu, X. Huang, J. Zhang, and M. Gidlund, "DeepHealth: A self-attention based method for instant intelligent predictive maintenance in industrial Internet of Things," *IEEE Transactions on Industrial Informatics,* vol. 17, no. 8, pp. 5461-5473, 2020.
[322] T. Zhou, G. Zhang, and Y. Cai, "Residual Self-Attention-Based Temporal Deep Model for Predicting Aircraft Engine Failure within a Specific Cycle," 2025.
[323] Q. Li, W. Lin, Z. Liu, and A. Prorok, "Message-aware graph attention networks for large-scale multi-robot path planning," *IEEE Robotics and Automation Letters,* vol. 6, no. 3, pp. 5533-5540, 2021.
[324] Z. Wang and M. Gombolay, "Learning scheduling policies for multi-robot coordination with graph attention networks," *IEEE Robotics and Automation Letters,* vol. 5, no. 3, pp. 4509-4516, 2020.
[325] Z. Wang, C. Liu, and M. Gombolay, "Heterogeneous graph attention networks for scalable multi-robot scheduling with temporospatial constraints," *Autonomous Robots,* vol. 46, no. 1, pp. 249-268, 2022.
[326] H. Kim, Y. Ohmura, and Y. Kuniyoshi, "Transformer-based deep imitation learning for dual-arm robot manipulation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021: IEEE, pp. 8965-8972.
[327] L. Chen *et al.*, "Transformer-based imitative reinforcement learning for multirobot path planning," *IEEE Transactions on Industrial Informatics,* vol. 19, no. 10, pp. 10233-10243, 2023.
[328] S. H. S. Hosseini, N. N. Joojili, and M. Ahmadi, "LLMT: A transformer-based multi-modal lower limb human motion prediction model for assistive robotics applications," *IEEE Access,* vol. 12, pp. 82730-82741, 2024.
[329] E. Alerskans, J. Nyborg, M. Birk, and E. Kaas, "A transformer neural network for predicting near-surface temperature," *Meteorological Applications,* vol. 29, no. 5, p. e2098, 2022.

[330] T. Nguyen *et al.*, "Scaling transformer neural networks for skillful and reliable medium-range weather forecasting," *Advances in Neural Information Processing Systems,* vol. 37, pp. 68740-68771, 2024.
[331] W. Zheng *et al.*, "GRU–transformer: a novel hybrid model for predicting soil moisture content in root zones," *Agronomy,* vol. 14, no. 3, p. 432, 2024.
[332] Y. Liang, S. Ke, J. Zhang, X. Yi, and Y. Zheng, "Geoman: Multi-level attention networks for geo-sensory time series prediction," in *Ijcai*, 2018, vol. 2018, pp. 3428-3434.
[333] K. F. Niresi, M. Zhao, H. Bissig, H. Baumann, and O. Fink, "Spatial-temporal graph attention fuser for calibration in iot air pollution monitoring systems," in *2023 IEEE SENSORS*, 2023: IEEE, pp. 01-04.
[334] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461,* 2018.
[335] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250,* 2016.
[336] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860,* 2019.
[337] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *arXiv preprint arXiv:1609.07843,* 2016.
[338] C. Chelba *et al.*, "One billion word benchmark for measuring progress in statistical language modeling," *arXiv preprint arXiv:1312.3005,* 2013.
[339] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model," in *2012 IEEE Spoken Language Technology Workshop (SLT)*, 2012: IEEE, pp. 234-239.
[340] R. Socher *et al.*, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631-1642.
[341] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 2011, pp. 142-150.
[342] Z. Chen, H. Zhang, X. Zhang, and L. Zhao, "Quora question pairs," ed, 2018.
[343] A. Wang *et al.*, "Superglue: A stickier benchmark for general-purpose language understanding systems," *Advances in neural information processing systems,* vol. 32, 2019.
[344] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *European conference on computer vision*, 2014: Springer, pp. 740-755.
[345] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980-2988.
[346] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 633-641.
[347] L. Beyer, O. J. Hénaff, A. Kolesnikov, X. Zhai, and A. v. d. Oord, "Are we done with imagenet?," *arXiv preprint arXiv:2006.07159,* 2020.
[348] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6154-6162.
[349] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified perceptual parsing for scene understanding," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 418-434.
[350] R. Goyal *et al.*, "The" something something" video database for learning and evaluating visual common sense," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5842-5850.
[351] W. Kay *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950,* 2017.
[352] A. Amir *et al.*, "A low power, fully event-based gesture recognition system," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7243-7252.
[353] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311-318.
[354] S. Lee *et al.*, "A survey on evaluation metrics for machine translation," *Mathematics,* vol. 11, no. 4, p. 1006, 2023.
[355] A. Warstadt, A. Singh, and S. R. Bowman, "Neural network acceptability judgments," *Transactions of the Association for Computational Linguistics,* vol. 7, pp. 625-641, 2019.
[356] B. Dolan and C. Brockett, "Automatically constructing a corpus of sentential paraphrases," in *Third international workshop on paraphrasing (IWP2005)*, 2005.
[357] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia, "Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation," *arXiv preprint arXiv:1708.00055,* 2017.
[358] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," *arXiv preprint arXiv:1704.05426,* 2017.
[359] I. Dagan, O. Glickman, and B. Magnini, "The pascal recognising textual entailment challenge," in *Machine learning challenges workshop*, 2005: Springer, pp. 177-190.
[360] D. Giampiccolo, B. Magnini, I. Dagan, and W. B. Dolan, "The third pascal recognizing textual entailment challenge," in *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, 2007, pp. 1-9.
[361] L. Bentivogli, P. Clark, I. Dagan, and D. Giampiccolo, "The Fifth PASCAL Recognizing Textual Entailment Challenge," *TAC,* vol. 7, no. 8, p. 1, 2009.
[362] H. J. Levesque, E. Davis, and L. Morgenstern, "The Winograd schema challenge," *KR,* vol. 2012, no. 13th, p. 3, 2012.
[363] Y.-H. H. Tsai, S. Bai, M. Yamada, L.-P. Morency, and R. Salakhutdinov, "Transformer dissection: a unified understanding of transformer's attention via the lens of kernel," *arXiv preprint arXiv:1908.11775,* 2019.
[364] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *International conference on machine learning*, 2020: PMLR, pp. 5156-5165.

[365] P. Deora, R. Ghaderi, H. Taheri, and C. Thrampoulidis, "On the optimization and generalization of multi-head attention," *arXiv preprint arXiv:2310.12680,* 2023.
[366] A. Zadeh, P. P. Liang, S. Poria, P. Vij, E. Cambria, and L.-P. Morency, "Multi-attention recurrent network for human communication comprehension," in *Proceedings of the AAAI conference on artificial intelligence*, 2018, vol. 32, no. 1.
[367] S. Lu, M. Liu, L. Yin, Z. Yin, X. Liu, and W. Zheng, "The multi-modal fusion in visual question answering: a review of attention mechanisms," *PeerJ Computer Science,* vol. 9, p. e1400, 2023.
[368] S. Zhang *et al.*, "Multimodal emotion recognition based on audio and text by using hybrid attention networks," *Biomedical Signal Processing and Control,* vol. 85, p. 105052, 2023.
[369] H. Chefer, Y. Alaluf, Y. Vinker, L. Wolf, and D. Cohen-Or, "Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models," *ACM transactions on Graphics (TOG),* vol. 42, no. 4, pp. 1-10, 2023.
[370] K. Lin *et al.*, "Swinbert: End-to-end transformers with sparse attention for video captioning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 17949-17958.
[371] J. Zhang, Y. Xie, W. Ding, and Z. Wang, "Cross on cross attention: Deep fusion transformer for image captioning," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 33, no. 8, pp. 4257-4268, 2023.
[372] X. Song, D. Han, C. Chen, X. Shen, and H. Wu, "Vman: visual-modified attention network for multimodal paradigms," *The Visual Computer,* vol. 41, no. 4, pp. 2737-2754, 2025.
[373] D. Mamieva, A. B. Abdusalomov, A. Kutlimuratov, B. Muminov, and T. K. Whangbo, "Multimodal emotion detection via attention-based fusion of extracted facial and speech features," *Sensors,* vol. 23, no. 12, p. 5475, 2023.
[374] A. K. Mohankumar, P. Nema, S. Narasimhan, M. M. Khapra, B. V. Srinivasan, and B. Ravindran, "Towards transparent and explainable attention models," *arXiv preprint arXiv:2004.14243,* 2020.
[375] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems,* vol. 33, pp. 9459-9474, 2020.
[376] S. Borgeaud *et al.*, "Improving language models by retrieving from trillions of tokens," in *International conference on machine learning*, 2022: PMLR, pp. 2206-2240.
[377] A. Jaegle *et al.*, "Perceiver io: A general architecture for structured inputs & outputs," *arXiv preprint arXiv:2107.14795,* 2021.
[378] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," *arXiv preprint arXiv:2312.00752,* 2023.
[379] M. R. Rezaee, N. A. W. A. Hamid, and Z. Ismail, "Federated Multi-Agent Reinforcement Learning for UAV Collision Avoidance in Dense Smart City Environment," in *2024 12th International Japan-Africa Conference on Electronics, Communications, and Computations (JAC-ECC)*, 2024: IEEE, pp. 7-10.
[380] E. Pignatelli, J. Ferret, M. Geist, T. Mesnard, H. van Hasselt, and L. Toni, "A Survey of Temporal Credit Assignment in Deep Reinforcement Learning," *Transactions on Machine Learning Research*.
[381] J. Ferret, R. Marinier, M. Geist, and O. Pietquin, "Self-attentional credit assignment for transfer in reinforcement learning," *arXiv preprint arXiv:1907.08027,* 2019.
[382] W. Li, W. Liu, S. Shao, S. Huang, and A. Song, "Attention-based intrinsic reward mixing network for credit assignment in multiagent reinforcement learning," *IEEE Transactions on Games,* vol. 16, no. 2, pp. 270-281, 2023.
[383] W. Wei, H. Li, S. Zhou, B. Li, and X. Liu, "Attention With System Entropy for Optimizing Credit Assignment in Cooperative Multi-Agent Reinforcement Learning," *IEEE Transactions on Automation Science and Engineering,* 2025.
[384] Z. Pu, H. Wang, Z. Liu, J. Yi, and S. Wu, "Attention enhanced reinforcement learning for multi agent cooperation," *IEEE Transactions on neural networks and learning systems,* vol. 34, no. 11, pp. 8235-8249, 2022.
[385] J. Shao *et al.*, "Complementary attention for multi-agent reinforcement learning," in *International conference on machine learning*, 2023: PMLR, pp. 30776-30793.
[386] I. P. Gomes, C. Premebida, and D. F. Wolf, "Multi-agent interaction-aware behavior intention prediction using graph mixture of experts attention network on urban roads," *Expert Systems with Applications,* vol. 270, p. 126485, 2025.
[387] S. Salmani Pour Avval, N. D. Eskue, R. M. Groves, and V. Yaghoubi, "Systematic review on neural architecture search," *Artificial Intelligence Review,* vol. 58, no. 3, p. 73, 2025.
[388] B. Ma, J. Zhang, Y. Xia, and D. Tao, "Auto learning attention," *Advances in neural information processing systems,* vol. 33, pp. 1488-1500, 2020.