Transformer 发展综述

李金莹

河海大学 计算机与软件学院,江苏 南京 211100

摘 要: Transformer 模型以其核心的注意力机制和并行计算能力,克服了传统循环神经网络和卷积神经网络在处理大规模数据和复杂任务时的局限,已成为现代人工智能的重要基础架构,广泛应用于自然语言处理,计算机视觉和多模态任务等领域,推动了多个学科的技术变革.因此,对 Transformer 的发展进行梳理是有必要的.本文系统梳理了 Transformer 模型的架构及其在不同领域的典型应用.首先,分析 Transformer 的基本结构,实现原理及其优缺点.接着,对 GPT,BERT 和 BART 三个 NLP 经典模型的设计进行了分析.在计算机视觉领域,分析了 ViT 和 Swin Transformer 模型,随后针对多模态任务,探讨了 CLIP 模型如何通过联合建模文本与图像实现跨模态语义理解.此外,讨论了 Transformer 模型的实际应用场景及未来发展方向.最后,本文通过实验提供了相关模型的直观评估与分析.

关键词: Transformer;自注意力;自然语言处理;计算机视觉;深度学习

Survey of Transformers

LI Jin-Ying

College of Computer Science and Software Engineering, Hohai University, Nanjing 211100, China

Abstract: The Transformer model, with its core attention mechanism and parallel computation capabilities, overcomes the limitations of traditional Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) in handling large-scale data and complex tasks. It has become an essential foundational architecture in modern artificial intelligence, widely applied in natural language processing, computer vision, and multimodal tasks, driving technological advancements across multiple disciplines. Therefore, it is necessary to introduce the development of Transformer. This paper systematically reviews the architecture of the Transformer model and its typical applications in various fields. First, it analyzes the basic structure and implementation principles of the Transformer, discussing its advantages and disadvantages. Next, the paper examines the design of three classic NLP models: GPT, BERT, and BART. In the computer vision field, the ViT and Swin Transformer models are analyzed. Then, the paper explores how the CLIP model achieves cross-modal semantic understanding by jointly modeling text and images in multimodal tasks. Additionally, the paper discusses the practical applications of Transformer models and their future development directions. Finally, experiments on related models are provided, offering a clear evaluation and analysis of their performance.

Key words: transformer; self-attention; natural language processing; computer vision; deep learning

1 introduction

Language Processing (NLP) Natural Language Processing (NLP) is an important branch of artificial intelligence, aiming to enable computers to understand, generate, and process human language. As a bridge connecting human intelligence and machine intelligence, NLP involves tasks such as language modeling, semantic analysis, machine translation, and text generation, and is widely used in scenarios such as search engines, intelligent assistants, and machine translation.

In the field of NLP, effectively modeling the contextual dependencies and semantic information of language has long been a core challenge. In 2014, Sutskever et al. proposed a Sequence-to-sequence learning (Seq2Seq) model [1] for machine translation tasks. The model consists of an encoder and a decoder and uses a long short-term memory network (LSTM). Since a single context vector is difficult to contain all the information, the Sequence-to-sequence learning model has some difficulty in processing long texts. Therefore, Bahdanau et al. proposed an attention model [2] for each hidden state in the encoder. Weights are calculated for all hidden states, and all hidden states are weighted during decoding, instead of the last hidden state, which improves the quality of the Seq2Seq model when processing long texts. Recurrent Neural Networks (RNNs) with added attention perform better. The next question is whether a neural network model can be directly implemented based on attention. In 2017, Vaswani et al. proposed the Transformer [3] model, which is directly based on the self - attention structure and the feed -forward neural network . It is implemented using FFNN, which allows it to compute the entire sequence in parallel without recursively processing it step by step.

In recent years, The Transformer model has made significant progress in the field of deep learning, and has been widely used in many fields such as natural language processing and computer vision, leading to a revolution in model architecture design. With its powerful parallel computing capabilities and global feature modeling abilities, the Transformer overcomes the limitations of traditional recurrent neural networks and convolutional neural networks (CNNs), becoming a core component of modern artificial intelligence systems. From language translation to image classification and multimodal tasks, the Transformer demonstrates outstanding performance and versatility across various fields. As research deepens, its variants and applications continue to emerge, further driving technological innovation in various fields and highlighting its crucial role as a foundational architecture.

Against this backdrop, it is of great significance to review the research on Transformers. The Transformer architecture, centered on the attention mechanism, overcomes the limitations of traditional methods when handling large-scale data and has become a foundational model in many fields. However, the flexibility and scalability of the Transformer have led to a wide variety of variants and applications, making it crucial to understand the evolutionary logic and innovative aspects of these models. A review of the basic principles and variants of the Transformer architecture helps to summarize current research findings, identify potential problems, and provide inspiration for future model design and applications.

Section 1 serves as an introduction, Section 2 analyzes the structure and implementation principle of Transformer and summarizes its advantages and disadvantages. Section 3 summarizes the research work on Transformer models in the NLP field and analyzes GPT [4]. BERT [5], BART [6] three typical models. Section 4 provides an overview of the research work on Transformer models in the field of computer vision, and analyzes ViT [7] and Swin [7]. The Transformer [8] model is an important model. Section 5 provides an overview of the use of the Transformer model in multimodal applications and analyzes the CLIP [9] model. Section 6 introduces the application of the Transformer model and discusses its future development direction. Section 7 presents experiments on related models and analyzes and summarizes them. Chapter 8 summarizes the entire text.

2 Overview of the Transformer Model

2.1 Transformer structure

The Transformer model is a type of Seq2Seq model, consisting of an encoder and a decoder. The encoder and decoder are composed of multiple layers of encoders and decoders, respectively ; this paper uses a six-layer encoder and a six-layer decoder. For each part of the model, the layers have the same structure, but different layers do not share parameters.

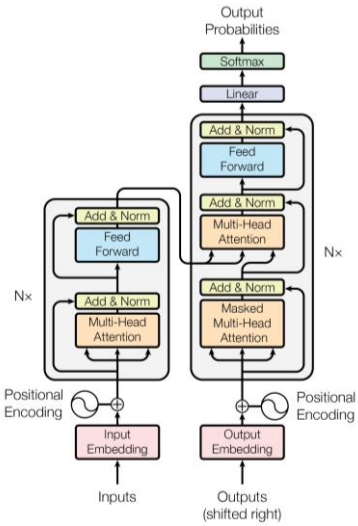


Fig. 1 The Transformer model architecture

Figure 1 Transformer model architecture

The Transformer model architecture is shown in Figure 1. Each encoder consists of a multi- head attention layer. The decoder consists of a multi-head attention layer (masked attention layer) and a feedforward neural network layer. Multi - Head Attention), and multi - head attention layers associated with the encoder (Encoder-Decoder). It consists of attention and feedforward neural network layers. Each sub-layer of the encoder and decoder is followed by a residual connection. Connection and layer-normalization are used to regularize vectors.

2.2 Implementation principle of Transformer

The implementation details of Transformer include three main parts: the representation of input and output text, the encoder, and the decoder.

For the input and output text, it is first transformed into a vector sequence through an embedding algorithm. The Transformer model appends position vectors to the text, which can provide the model with more meaningful information, such as location information and distance information. The vector obtained from embedding and the position vector are added together to obtain the final vector representation.

The design of positional encoding :

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

Fig. 2 Positional Encoding

Figure 2 Location coding method

The position encoding method is shown in Figure 2, where *pos* is the position. *d_{model}* is the dimension of the vector. *i represents the i-th dimension* of the vector , and the position vector of the d- *model dimension at the p _{os}* position can be obtained by the formula . In this way, the generation of position vectors can be extended to unknown sequence lengths.

During the encoding phase, the input to the first encoder layer is the processed vector, and its output is then fed into the next encoder layer.

The self-attention mechanism of the encoder:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Fig. 3 Attention function

Figure 3 Self-attention mechanism

The self-attention mechanism is shown in Figure 3. The vectors are ***Q*** , ***K*** , ***and V , where Q*** represents the query. ***K*** and ***V*** are key -value pairs. *d* represents the vector dimension. The calculation logic of attention can be described as two parts: the calculation of Query and Key is related, also known as the calculation of Attention score; the Value is weighted and summed according to the Softmax regularization result of Attention score, thus obtaining the result of the current self-attention layer .

Encoder multi-head self-attention mechanism design:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Fig . 4 Multi Head function

Figure 4. Multi-head self-attention mechanism

The multi-head self-attention mechanism is shown in Figure 4. *h* represents the number of attention heads.

A single-group self-attention outputs only a single vector, which is likely dominated by a single piece of information. To enable the model to focus on more useful information, the Transformer proposes a multi-head attention mechanism, which further improves the self-attention mechanism by setting multiple attention heads . The weights of each attention head are randomly initialized, but after training, the weights of each attention head can map the input vector to the corresponding representation space, allowing the model to focus on information at different locations.

Since the feedforward neural network layer only accepts one matrix, multiple matrices are concatenated to obtain a large matrix, which is then multiplied by another weight matrix ***W*** to map to the required dimension. This matrix contains all the information from the attention heads and is then input into the feedforward neural network layer.

Each time step in the decoding phase has an output. The output of the current time step of the decoder will be used as the input of the next decoder step. This process is repeated until the output ends.

The decoder employs a masked multi-head attention layer design: the attention in this layer is only allowed to see information in the output sequence that is earlier than the current position. This is achieved by masking the

information of positions after the current position after calculating the attention score, setting the attention score to -inf, and then performing Softmax regularization.

The decoder employs a multi-head attention layer design related to the encoder: the query matrix in the input is constructed from the output of the previous layer, while the key and value matrices come from the output of the last encoder layer. This helps the decoder focus its attention on the appropriate positions in the input vector sequence.

Each decoder and encoder has a position-wise feed-forward neural network layer :

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

Fig. 5 FFN function

Figure 5. Position-based feedforward neural network layer

The feedforward neural network is shown in Figure 5, where the input x is the vector of the output of the previous layer. ***W1*** , ***W2*** is the weight matrix, and b1 and *b2* are *bias* vectors . The parameters are different for each feedforward neural network layer. FFN consists of two fully connected layers, with the ReLU activation function used in between . The first layer expands the dimension, for example, from 512 to 2048, and then applies the activation function; the second layer compresses the vector back to the original dimension.

The decoder outputs a vector, which needs to be converted into words using linear and softmax layers. The linear layer is a standard fully connected neural network that maps the decoder's output vector to a larger vector space, where each number represents the score of the corresponding word. The softmax layer then performs regularization, transforming the scores into probabilities, and finally selects the word with the highest probability as the output for the current time step.

2.3 Analysis of Transformer

The Transformer's basic architecture is centered on a self-attention mechanism, freeing it from the dependence of recurrent neural networks on sequence computation. By introducing a multi-head attention mechanism, it can capture global dependencies between different positions in the input sequence, thereby improving feature extraction capabilities. In the input and output text processing stages , the positional encoding design allows the model to learn the sequential information of the input sequence without explicit sequence connection structures. Furthermore, the Transformer's parallelization design significantly improves training efficiency, making it particularly suitable for pre-training on large-scale datasets . The Transformer's modular structure is easy to extend and transfer, and it has become a foundational model in fields such as natural language processing, computer vision, and multimodal learning.

Despite its many advantages, the Transformer still has areas for improvement. When sequences are long, the computational and memory overhead of the self-attention mechanism increases dramatically, limiting its application on resource-constrained devices. The Transformer is heavily reliant on large-scale data, and its performance on small datasets may be inferior to traditional methods. Furthermore, the lack of explicit modeling of local context in the attention computation of each layer in the Transformer may lead to insufficient representation of local features in certain tasks. To address these issues, numerous improvement schemes have been proposed in recent years, such as sparse attention mechanisms , attempting to maintain or even enhance model performance while improving efficiency.

3 NLP Research Based on Transformer Architecture

3.1 NLP Overview

Language Processing (NLP) Natural Language Processing (NLP) is a discipline within the fields of linguistics and machine learning. NLP studies information related to natural language, including how to understand and process natural language to interact more effectively with humans. The focus of NLP research is on moving beyond understanding individual words or phrases to understanding the context of those words, i.e., sentences and paragraphs.

The main tasks of NLP research include:

(1) Text understanding includes part-of-speech tagging, syntactic analysis, semantic analysis, and sentiment analysis. It can perform functions such as extracting information from text, answering text-related questions, summarizing text, identifying grammatical errors, and detecting spam.

(2) Text generation: It can automatically generate text based on prompts, such as sentence completion, text paragraph generation, and machine translation.

(3) Speech processing and generation: Implement the reading aloud of a given text and recognize a speech segment to convert it into text.

Natural language primarily refers to human language, that is, the language humans use in daily life, and is therefore easy for humans to understand. However, NLP aims to enable machines to understand human language. The information contained in text, such as actions and emotions, needs to be converted into a form that allows machine learning, which is a challenge in NLP research.

The development of Natural Language Processing (NLP) can be broadly divided into three stages: rule-based, statistical, and machine learning-based. Traditional methods process natural language based on syntactic-semantic rules, primarily through matching. However, this approach is overly complex, requiring the formulation of numerous rules, which can lead to inconsistencies. Subsequently, statistical methods emerged, evaluating the validity of sentences based on their probability of occurrence. Some machine learning methods have adopted statistical approaches. Currently, NLP increasingly utilizes deep learning methods, employing deep neural networks to process large-scale real-world text. Before the advent of the Transformer model, the primary technology used in NLP was the RNN.

3.2 NLP The Development and Classification of Transformer Models

The Transformer architecture was proposed in June 2017, and its initial research direction was translation tasks. Subsequently, a series of Transformer-based models emerged, including GPT , BERT, BART, T5 [10].

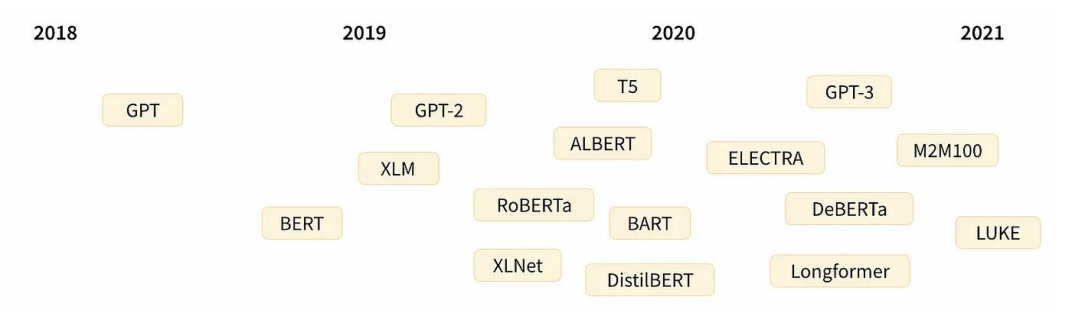


Fig. 6 Models since 2018 based on the original Transformer

Figure 6. Models based on the Transformer architecture since 2018

Figure 6 shows some important Transformer models since 2018. These models can be divided into three categories:

(1) Encoder -Only models are primarily used for text information extraction tasks, such as text classification and sentiment analysis; BERT is a representative example .

(2) Decoder -Only models are typically used for sequence generation tasks, such as text generation and machine translation; GPT is a representative example.

(3) The Encoder-Decoder model, also known as the sequence -to-sequence model , is used for complex tasks such as summarizing from input text , machine translation , and question answering . Examples include BART and T5 .

This paper will further analyze the representative GPT of each of the three types of models. BERT, BART will be introduced.

3.3 GPT

Traditional natural language processing methods have shortcomings in terms of relying on labeled data and task-specific architecture design. One is the scarcity of labeled data, as acquiring large-scale labeled data is time-consuming and expensive; the other is poor transferability, as traditional models are difficult to transfer from one task to another. GPT (Generative Pre-trained Transformer) proposes a semi-supervised approach that aims to enhance the model's generalization ability by combining unsupervised pre - training with supervised fine -tuning to learn general language representations from large-scale unlabeled data.

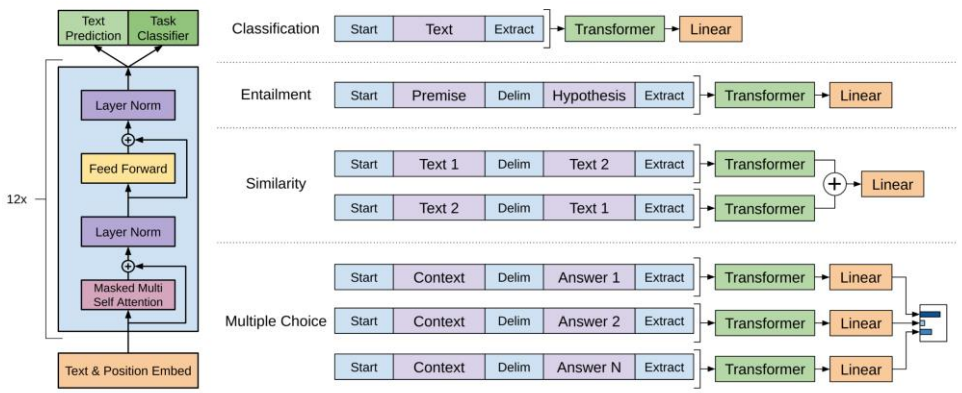


Fig. 7 GPT model

Figure 7 GPT model

GPT uses the decoder part of Transformer , and the training process consists of two stages: a pre-training stage and a fine-tuning stage. Figure 7 shows the architecture of GPT on the left and the fine-tuning for different tasks on the right.

$$L_1(\mathcal{U}) = \sum_i \log P(u_i|u_{i-k}, \ldots, u_{i-1}; \Theta)$$

$$h_0 = UW_e + W_p$$
$$h_l = \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n]$$
$$P(u) = \texttt{softmax}(h_n W_e^T)$$
$$P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y).$$

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m).$$

Fig. 8 Functions from GPT

Formulas in Figure 8 GPT

the pre-training phase, GPT is trained on a large-scale unlabeled text, with the goal of learning the distribution of text sequences. The formula in the first row of Figure 8 learns the distribution of text sequences by maximizing the log-likelihood function of language modeling, where *U* represents the expected word sequence during training. *k* represents the size of the context window, which is the first *k* words that each word depends on. *P* represents the conditional probability of the model predicting the current word based on the previous k words .

The formulas in rows 2-4 of Figure 8 illustrate the construction of the multi-layered Transformer decoder used by GPT. The input vector *h* ₀ consists of two parts. *U* represents the context vector sequence, ***We W*** _{*p*} represents the word embedding matrix and the position embedding matrix. Subsequently, each layer updates the hidden state through a Transformer block. The input of the *l- th* layer is *h* _{*l* -1} , and the output is *h* _{*l* -1}. Finally , the probability distribution *P is calculated using the hidden state h* _{*n*} output from the last layer and the weight matrix ***W _{*** *e*} .

- tuning phase, the model is adapted to a specific supervised task. Given a labeled dataset *C* , where each sample contains an input sequence { *x1* , … , *xm* } *and a label y, the goal is to maximize the log* - likelihood *L2* , the activation value of the last Transformer block ' and the parameter *Wy* to predict *y* , as shown in the formulas in rows 5 and 6 of Figure 8.

GPT's innovation lies in capturing the contextual dependencies of natural language through the use of the Transformer framework; it significantly improves the performance on small data tasks by using unsupervised pre-training combined with supervised fine-tuning; and because the pre-training stage does not depend on specific tasks, the model can generalize to a variety of tasks, eliminating the need to design new architectures for different tasks compared to traditional methods.

GPT also has areas for improvement. It only uses left-to-right context and cannot fully utilize global context information; large-scale pre-training requires significant computational resources and has high computational costs.

Based on GPT-1, GPT-2 [11] and GPT-3 [12] were developed .

3.4 BERT

Before BERT, most natural language processing models (such as GPT and traditional language models) only used unidirectional or shallow context modeling. For example, GPT can only model from left to right, making it difficult to capture the complete context. BERT aims to learn information representations containing rich global contextual semantics through bidirectional modeling, making it suitable for various NLP tasks such as text classification, question answering, and recognition.

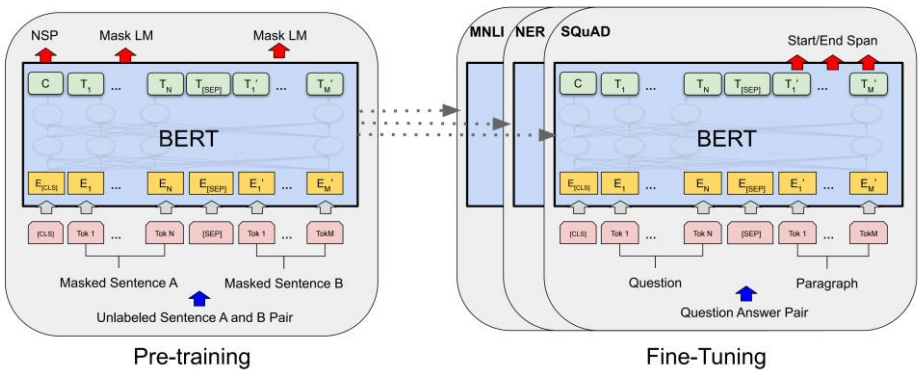


Fig. 9 BERT model

Figure 9 BERT model

As shown in Figure 9, BERT combines large-scale unsupervised pre-training with task-specific supervised fine-tuning and uses the encoder part of the Transformer to achieve full context modeling.

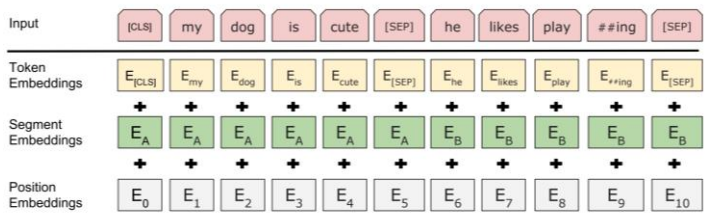


Fig. 10 BERT input representation

Figure 10 Input representation of BERT

The input representation of BERT consists of three parts: token embedding, segmentation embedding (which divides different sentences), and position embedding , as shown in Figure 10.

The pre-training phase includes two tasks: Masked Language Model (Mask) LM) and Next Sentence Prediction (NSP). Mask LM randomly masks some words and then predicts those masked words; NSP is used to determine whether two sentences are adjacent.

During the fine-tuning phase, for each task, simply insert the task-specific inputs and outputs into BERT and fine-tune all parameters end-to-end .

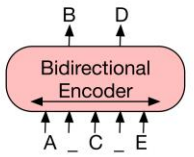
BERT's innovation lies in bidirectional contextual modeling. Through the Mach LM task, BERT learns global dependencies between words; further, by combining it with the NSP task, it enhances the model's performance in long text comprehension and inter-sentence relationship judgment.

The drawbacks of BERT are that pre-training is costly and requires a lot of computing resources; the inference time of the bidirectional architecture is also relatively long.

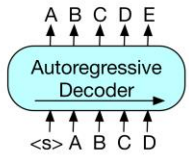
3.5 BART

Traditional generative models have limited performance in complex natural language generation tasks, while existing bidirectional models (such as BERT), although excellent in understanding tasks, cannot be directly applied to generation tasks. BART combines the advantages of BERT and GPT, enabling it to handle generation tasks such

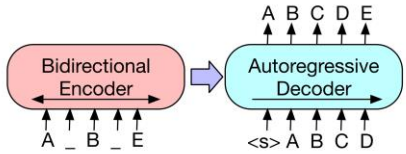
as text summarization and translation, generating high-quality natural language output, as well as understanding tasks such as classification and question answering, demonstrating excellent performance in context understanding.



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.



(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

Fig. 11 A schematic comparison of BART with BERT and GPT

Figure 11. A brief comparison of BART with BERT and GPT.

BART's architecture is based on the Transformer, specifically consisting of an encoder, similar to BERT, used to encode the input text; and a decoder, similar to GPT, used to progressively generate the output text. This design allows BART to simultaneously possess bidirectional understanding and generation capabilities. Figure 11 shows a simplified comparison of BART with BERT and GPT.

In the preprocessing stage, BART uses diverse noiseing techniques to make the model focus more on semantic understanding of the text. As shown in Figure 12, BART uses five noiseing methods for training, namely, randomly replacing individual words with tokens. Masking), randomly deleting individual words (Token) Deletion, randomizing the order of a text sequence (Sentence Permutation), preserving the order of the text but randomly selecting the starting point (Document Rotation), and replacing consecutive words with a mask or randomly generating a mask (Text Infilling).

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D},\ \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D} \tag{1}$$

$$\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1 \ldots L \tag{2}$$

$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \qquad \ell = 1 \ldots L \tag{3}$$

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0) \tag{4}$$

Fig. 15 Equations from ViT

Figure 15 Equations in ViT

Figure 15 illustrates some of the methods used by ViT . The first is patch embedding : the standard Transformer receives a one-dimensional vector sequence, while an image is two-dimensional. For the height, width, and number of channels, H , ... W , An image of size C needs to be segmented into N *non-overlapping* blocks of size $P \times P$, where N *is related* to H , W , and P. Each block is flattened into a one-dimensional vector and mapped to an

embedding space of fixed dimension D through a linear transformation, resulting in an embedding sequence *z0* . An additional BERT-like [class] is added to this sequence . The extra-learnable [ckass] embedding of the token is used as the representation *y* of the image when the Transformer outputs this embedding .

Positional Encoding : Transformers lack position awareness, so explicit positional encoding is required. Vit incorporates positional embeddings into block embeddings to preserve positional information, and then uses the resulting embedding vectors as input to the Transformer.

A standard Transformer encoder includes a multi-head self-attention mechanism (MSA) and a feedforward neural network (MLP) module. Layer normalization (LN) is used before each module; residual connections are used after each module. Connection).

ViT are: it uses a self-attention mechanism to model global dependencies, resulting in strong global modeling capabilities; it uses a standard Transformer architecture , making it highly versatile and transferable to various vision tasks; and it can fully utilize large-scale image data through pre-training , especially excelling in high-resource scenarios.

ViT also has some shortcomings: its performance depends on large-scale data and is only average on small datasets; the computational complexity of the self-attention mechanism is quadratic , which is inefficient for high-resolution image processing; and the self-attention mechanism is global and cannot perceive local details well.

3.6 Swin Transformer

In ViT , the image is divided into a series of fixed-size patches, and these patches are then processed using a self-attention mechanism. While this method effectively captures global information, it suffers from two problems in practical applications: First, ViT performs global self-attention calculations directly on the entire image, which is computationally intensive, especially in high-resolution images, leading to excessive consumption of computational resources; secondly, ViT performs poorly in modeling the local structure and detailed features of an image because the contextual information of each image patch depends on the global image patch.

To overcome these problems, Swin Transformer proposes a novel design that utilizes a shifted window-based self-attention mechanism. Window -based self-attention is used to process image patches locally, reducing computational complexity. At the same time, the hierarchical structure gradually extracts features from local to global features, enhancing the model's performance in visual tasks.

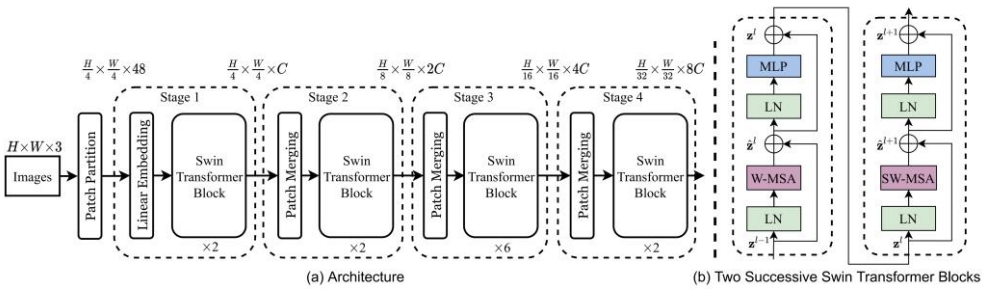


Fig. 16 The architecture of Swin Transformer

Figure 16 Architecture of the Swin Transformer

The implementation of Swin Transformer consists of three parts, as shown in Figure 16.

First, the input image is segmented into a series of non-overlapping patches. (Partition), where each block is a fixed-size window.

Then , use Swing Transformer Block is processed. Swin Transformer Block contains two types of MSA:

W-MSA (Window-based Multi-head Self Attention). SW-MSA (Shifted) (two types of Swin based Multi-head Self Attention) Transformer Blocks are adjacent. W-MSA calculates the attention weights within each window, while SW-MSA aggregates information from different windows through a sliding window.

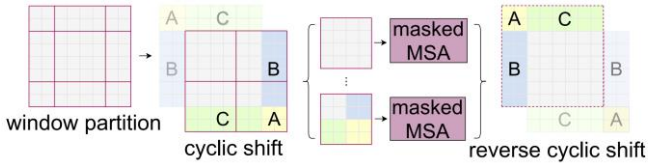


Fig. 17 An approach for self-attention in shifted window partitoning

Figure 17 Segmentation method of sliding window in self-attention mechanism

After the window slides, both the number of windows and the size of the edge windows change. To address this, the authors propose a batch computation method, as shown in Figure 17, which uses a mask to restrict the changes while keeping the number of windows and the window size constant.

Swin Transformer After block processing, downsampling operation (Patch) is performed. (Merging), increasing the window size.

The Swin Transformer is the first to propose limiting the computational scope of each self-attention through window partitioning, which makes computation more efficient. As the network depth increases, the window size can gradually increase, thereby capturing richer global information. This method not only solves the computational problem of ViT in visual tasks but also achieves better results in visual tasks such as object detection and semantic segmentation. Swin Transformer is used as a backbone for general vision tasks.

The Swin Transformer also has some shortcomings. When the input image resolution is high, the receptive field of local regions is limited, and the area that can be covered by window sliding is small, which is not conducive to processing. In addition, the Swin Transformer is quite sensitive to the setting of hyperparameters, and parameters such as window size and number of layers need to be carefully adjusted.

4 Overview of CLIP, a multimodal model based on the Transformer architecture

In traditional computer vision tasks, most models rely on specific datasets and fixed tasks, such as image classification on ImageNet. However, this approach limits the applicability of the models, which struggle to handle categories not seen during training, especially in open-world domains. Unlike machines, humans can flexibly describe and understand new concepts through natural language. Inspired by this, research has emerged that combines natural language and vision. CLIP (Contrastive Language–Image Pretraining) is a groundbreaking multimodal model proposed by OpenAI. Its core idea is to align images and text into the same embedding space through a contrastive learning mechanism, enabling the model to simultaneously understand linguistic descriptions and visual content and establish a connection between the two. CLIP is trained using large-scale image-text pairs, enabling the model to perform open-domain image recognition and understanding using natural language without requiring extensive task-specific annotations.

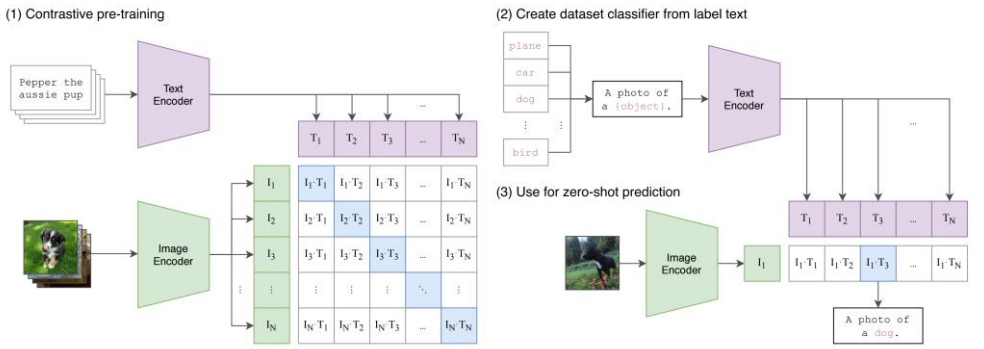


Fig. 18 Summary of CLIP

Figure 18 Overview of CLIP

Figure 18 provides an overview of CLIP. CLIP consists of two independent encoders: a visual encoder, based on ResNet or ViT , for processing images, mapping the input image to a fixed-size image feature vector; and a text encoder, based on Transformer, for processing natural language text, encoding the input text sequence into a fixed-size text feature vector.

CLIP uses contrastive pre-training . For a batch of training samples, it first calculates the feature representation of each image and the feature representation of each text, and then uses cosine similarity to calculate the similarity between the image and the text. For correct pairings, it aims to maximize the cosine similarity; for incorrect pairings, it aims to minimize the cosine similarity.

Zero -shot prediction

CLIP can directly handle unseen tasks without additional fine-tuning, and its powerful zero-shot learning capability makes it perform well on a variety of open- domain tasks . At the same time, since the training data comes from a large number of Internet image-text pairs, CLIP can capture rich semantic information and is more adaptable to different tasks.

CLIP's shortcomings include: CLIP cannot handle relatively abstract logical reasoning tasks; CLIP makes predictions by finding the best-matching existing label rather than predicting new labels, which is not automated enough; and it is limited by computing resources. CLIP cannot perform generative networks for image-text pairs.

OpenAI also proposed DALL-E [18] for generating images from text, which uses CLIP to identify a large number of alternative results.

5 Applications and development directions of the Transformer model

As a deep learning architecture, the Transformer model has been continuously developed since it was proposed. Figure 19 shows the growth of Transformer and ViT references [19] . The Transformer model is widely used in various tasks in the real world.

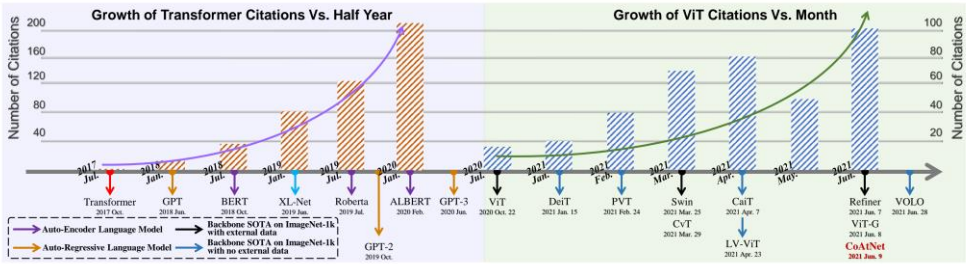


Fig. 19 Growth of Transformer and ViT citations

Figure 19 Growth of Transformer and Vit references

NLP tasks require semantic understanding, generation, or analysis of text. These tasks typically require models to handle long text sequences and capture contextual information, such as syntax, syntactic structure, and semantic relationships. Applications of the Transformer model in NLP include:

Machine translation: For example, Google Translate enables accurate translation between different languages.

Text generation: such as ChatGPT, can generate coherent and meaningful long texts.

Sentiment analysis: Analyzing the sentiment of comments, articles, or social media content.

Information extraction: Extracting structured information, such as entities and relationships, from unstructured text.

Computer vision tasks require models to process and understand patterns, structure, and semantic information in visual data. Traditionally, these tasks rely on convolutional neural networks, but Transformer models are becoming increasingly important, with applications including:

Image classification: dividing images into different categories, such as analyzing X-ray images in the medical field.

Object detection: Identifying and locating specific objects in an image, such as pedestrians or vehicles in an autonomous vehicle.

Video analytics: classifying motion, behavior, or events in a video.

The Transformer model has also been applied to other fields, such as image generation in multimodal domains and bioinformatics.

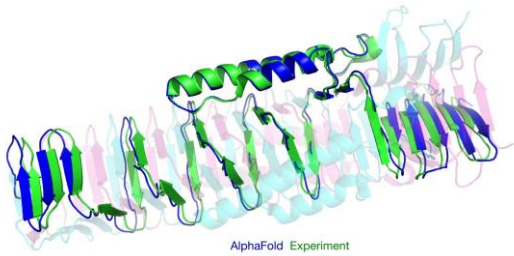


Fig. 2 0 AlphaFold 2 Experiment

Figure 2.0 AlphaFold2 experimental results

Bioinformatics tasks require extracting structural or functional information from complex biological sequences, such as DNA, RNA, or proteins. Transformer excels in modeling sequence dependencies, and its application is the AlphaFold2 [20] model based on sequence inference of protein three-dimensional structure, as

shown in Figure 20.

With technological advancements, the Transformer model will be further expanded to a wider range of application scenarios. The future development directions of the Transformer model include: optimizing computational efficiency, especially when processing long sequences and large-scale data, reducing computational resource consumption, and improving inference speed; secondly, enhancing the model's scalability and adaptability, exploring more lightweight Transformer variants to adapt to scenarios with limited computational resources; and further developing cross-modal learning, improving the deep fusion between different modalities such as text, images, and audio, and promoting the development of multimodal tasks.

6 Experiments on the Transformer model

The purpose of this experiment is to explore the performance and applicability of the Transformer model in different tasks. The experiments include BERT masked language modeling, BART text summarization, CLIP zero-shot image classification, and SwinIR [21] image restoration. These tasks cover multiple fields such as natural language processing, computer vision, and multimodal processing. Through the experiments, we hope to analyze the performance of the Transformer architecture in different task scenarios and provide a basis for further optimization of the model.

The experimental data includes text and images. The text is the Chinese and English abstracts of this paper, and the images are photos taken on campus and pictures of polar bears and pandas from the internet.

The basic process of the experiment involves calling the model's API on a cloud server to write an .ipynb file, inputting data, running the program, and generating results and charts.

Figure 21 shows a code writing process for CLIP zero-shot image classification experiments and analysis.

```
def query(data):
    with open(data["image_path"], "rb") as f:
        img = f.read()
    payload={
        "parameters": data["parameters"],
        "inputs": base64.b64encode(img).decode("utf-8")
    }
    response = requests.post(API_URL, headers=headers, json=payload)
    return response.json()

output = query({
    "image_path": "/content/20240316.jpg",
    "parameters": {"candidate_labels": ["cat", "koala","bird","polarbear","panda"]},
})
#print(output)
#Left
image_path = '/content/20240316.jpg'
image_name = 'polarbear.jpg'
img = mpimg.imread(image_path)
fig, axs = plt.subplots(1, 2, figsize=(15, 6))
axs[0].imshow(img)
axs[0].axis('off')
axs[0].text(0.5, -0.1, image_name, ha='center', va='bottom', transform=axs[0].transAxes)
```

Fig. 2 1 Code

Figure 2.1 Code

The BERT masked language modeling task uses a BERT -based- Chinese model .

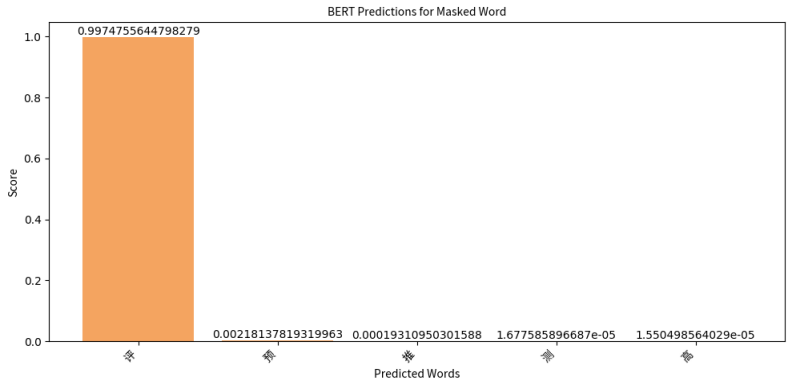


Fig. 2 2 Results o f BERT

Figure 2.2 BERT prediction results

The original text states: Finally, this paper provides an intuitive evaluation and analysis of the relevant models through experiments.

The input is: Finally, this paper provides an intuitive [MASK] estimation and analysis of the relevant models through experiments .

As shown in Figure 22, BERT predicts the mask content as "评" with a probability of 0.99 , which is very accurate.

The model used in BART's text summarization task is bart -large- cnn .

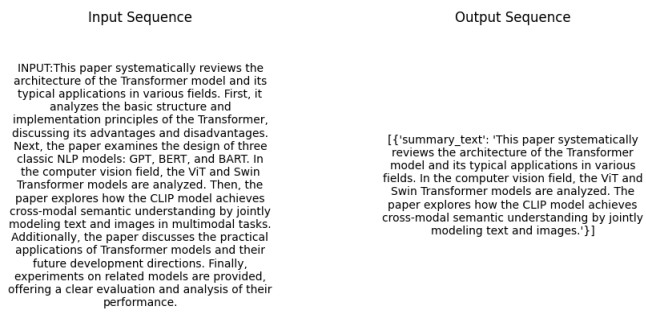


Fig. 2 3 Results o f B A RT

Figure 2.3 BART generation results

As shown in Figure 23, BART generates a text summary that provides a general overview of the text, but it loses information about the NLP domain model.

The CLIP model is used for zero-shot image classification. The model is clip-vit-large-patch14 , and the inputs are images of polar bears and pandas.

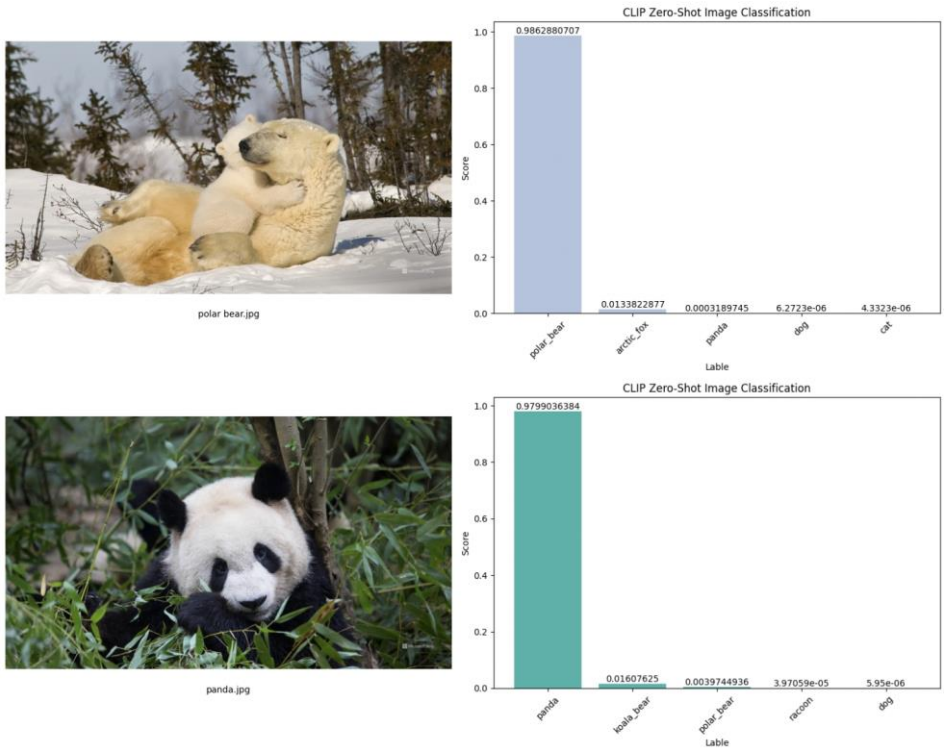


Fig. 2 4 Results o of CLIP about polar bear and panda image

Figure 2.4 CLIP classification results for polar bear and panda images.

As shown in Figure 24, the classification probabilities for polar bear and giant panda images are 0.986 and 0.980 , respectively , both of which are very accurate.

SwinIR is based on Swin The Transformer image restoration model takes a campus photo as input.

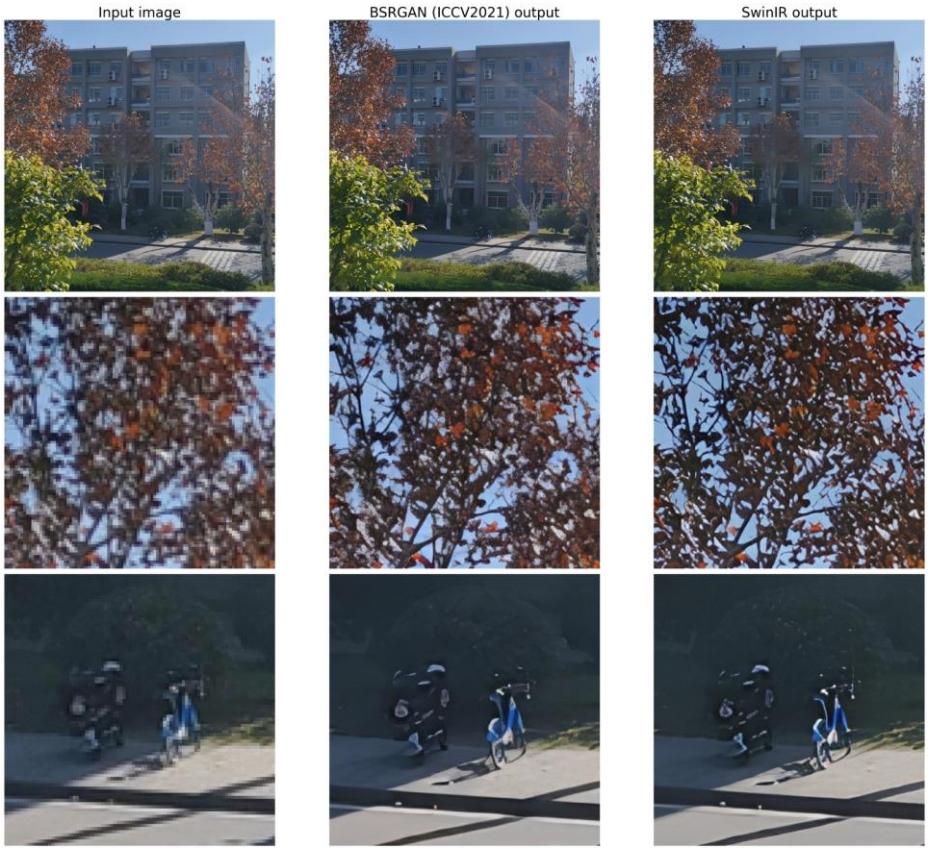


Fig. 2 5 Results o f BSRGAN and SwinIR

Figure 2.5 Image restoration results using BSRGAN and SwinIR .

Figure 25 shows the image restoration results of BSRGAN [22] and SwinIR , including the overall image and local magnification of leaves and non-motorized vehicles. It can be seen that SwinIR has richer and more accurate details, but both models lose information about dark bricks when processing building walls.

Overall, this experiment demonstrates the broad applicability of the Transformer architecture across multiple domains and tasks. Future research can further optimize the model and explore more innovative application scenarios to fully realize the potential of the Transformer architecture.

7 Summarize

This paper analyzes the design, advantages, and disadvantages of Transformer-related models, conducts experiments, and comprehensively discusses their application scenarios in different fields.

The advantages of the Transformer model are mainly reflected in its powerful parallel computing capabilities and global feature modeling capabilities. However, the Transformer also has the problem of high computational resource consumption, especially in the processing of long sequences, where the computational complexity is high.

In the field of natural language processing, Transformer-based models such as GPT, BERT, and BART have performed well, but they also have their own limitations. As an autoregressive generative model, the GPT model excels at text generation tasks and has strong generation capabilities, but it is relatively weak in context

understanding and struggles to effectively handle complex language reasoning. BERT, as a bidirectional encoding model , outperforms GPT in context understanding and is widely used in text classification and question answering tasks. However, its generation ability is weak and it cannot generate long texts as smoothly as GPT. The BART model combines the advantages of both GPT and BERT, making it suitable for tasks such as text generation and summarization. However, its training process is relatively complex and relies on a large amount of computing resources.

In the field of computer vision, ViT and Swin Transformer demonstrate the capabilities of Transformers in image processing. ViT directly applies the Transformer architecture to process images. By dividing images into fixed-size blocks and flattening them into sequences, it breaks the CNN's dependence on local features and successfully improves image classification performance. However, ViT requires a large amount of training data and performs worse than CNN on small datasets. The Swin Transformer effectively controls computational complexity and improves performance on visual tasks by introducing a sliding window mechanism, especially excelling in object detection and image segmentation. However, its local computation method makes the training and inference process of the model relatively complex.

In multimodal tasks, the CLIP model demonstrates the powerful capabilities of the Transformer in cross-modal tasks involving text and images. By jointly training images and text, CLIP can achieve cross-modal similarity calculation and performs well in tasks such as image search and image caption generation. However, the CLIP model has high requirements for the quality and diversity of training data and has certain limitations when dealing with highly complex tasks.

Experiments show that Transformer-based models perform differently on different tasks, and specific models should be configured appropriately according to the characteristics of the task.

In summary , Transformer and its variants have demonstrated powerful capabilities in multiple fields, but they still face challenges such as high computational resource consumption and strong data dependency.

References :

[1] Sutskever I. Sequence to Sequence Learning with Neural Networks[J]. arXiv preprint arXiv:1409.3215, 2014.

[2] Bahdanau D. Neural machine translation by jointly learning to align and translate[J]. arXiv preprint arXiv:1409.0473, 2014.

[3] Vaswani A. Attention is all you need[J]. Advances in Neural Information Processing Systems, 2017.

[4] Radford A. Improving language understanding by generative pre-training[J]. 2018.

[5] Devlin J. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[6] Lewis M. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension[J]. arXiv preprint arXiv:1910.13461, 2019.

[7] Dosovitskiy A. An image is worth 16x16 words: Transformers for image recognition at scale[J]. arXiv preprint arXiv:2010.11929, 2020.

[8] Liu Z, Lin Y, Cao Y, et al. Swin transformer: Hierarchical vision transformer using shifted windows[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2021: 10012-10022.

[9] Radford A, Kim JW, Hallacy C, et al. Learning transferable visual models from natural language supervision[C]//International conference on machine learning. PMLR, 2021: 8748-8763.

[10] Raffel C, Shazeer N, Roberts A, et al. Exploring the limits of transfer learning with a unified text-to-text transformer[J]. Journal of machine learning research, 2020, 21(140): 1-67.

[11] Radford A, Wu J, Child R, et al. Language models are unsupervised multitask learners[J]. OpenAI blog, 2019, 1(8): 9.

[12] Brown T B. Language models are few-shot learners[J]. arXiv preprint arXiv:2005.14165, 2020.

[13] Carion N, Massa F, Synnaeve G, et al. End-to-end object detection with transformers[C]//European conference on computer vision. Cham: Springer International Publishing, 2020: 213-229.

[14] Zheng S, Lu J, Zhao H, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021: 6881-6890.

[15] Meinhardt T, Kirillov A, Leal- Taixe L, et al. Trackformer : Multi-object tracking with transformers[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022: 8844-8854.

[16] Chen H, Wang Y, Guo T, et al. Pre-trained image processing transformer[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021: 12299-12310.

[17] Lin K, Li L, Lin CC, et al. Swinbert : End-to-end transformers with sparse attention for video captioning[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 17949-17958.

[18] Ramesh A, Pavlov M, Goh G, et al. Zero-shot text-to-image generation[C]//International conference on machine learning. Pmlr , 2021: 8821-8831.

[19] Liu Y, Zhang Y, Wang Y, et al. A survey of visual transformers[J]. IEEE Transactions on Neural Networks and Learning Systems, 2023.

[20] Jumper J, Evans R, Pritzel A, et al. Highly accurate protein structure prediction with AlphaFold[J]. nature, 2021, 596(7873): 583-589.

[21] Liang J, Cao J, Sun G, et al. Swinir : Image restoration using swin transformer[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2021: 1833-1844.

[22] Zhang K, Liang J, Van Gool L, et al. Designing a practical degradation model for deep blind image super-resolution[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021: 4791-4800.