• REVIEW •

An overview of domain-specific foundation model: key technologies, applications and challenges

Haolong CHEN[1,2], Hanzhi CHEN[1], Zijian ZHAO[1,5], Kaifeng HAN[3*],
Guangxu ZHU[1,2*], Yichen ZHAO[4], Ying DU[3], Wei Xu[6,7] & Qingjiang SHI[1,8]

[1]*Shenzhen Research Institute of Big Data, Shenzhen 518172, China;*
[2]*School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen 518172, China;*
[3]*China Academy of Information and Communications Technology, Beijing 100191, China;*
[4]*China Mobile Group Device Co., Ltd., Beijing 100033, China;*
[5]*School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China;*
[6]*National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China;*
[7]*Purple Mountain Laboratories, Nanjing 211111, China;*
[8]*School of Software Engineering, Tongji University, Shanghai 201804, China*

Abstract   The impressive performance of ChatGPT and other foundation-model-based products in human language understanding has prompted both academia and industry to explore how these models can be tailored for specific industries and application scenarios. This process, known as the customization of domain-specific foundation models, addresses the limitations of general-purpose models, which may not fully capture the unique patterns and requirements of domain-specific data. Despite its importance, there is a notable lack of comprehensive overview papers on building domain-specific foundation models, while numerous resources exist for general-purpose models. To bridge this gap, this article provides a timely and thorough overview of the methodology for customizing domain-specific foundation models. It introduces basic concepts, outlines the general architecture, and surveys key methods for constructing domain-specific models. Furthermore, the article discusses various domains that can benefit from these specialized models and highlights the challenges ahead. Through this overview, we aim to offer valuable guidance and reference for researchers and practitioners from diverse fields to develop their own customized foundation models.

Keywords   artificial intelligence; domain-specific foundation model; multi-modality foundation model; pre-training foundation model; fine-tuning

1 Introduction

ChatGPT has redefined people's understanding of artificial intelligence with its outstanding performance. As its core technology, the Large Language Model (LLM) has become an essential tool for researchers and practitioners across various fields to improve their workflows. General-purpose foundation models are usually trained on large public datasets, enabling them to learn and address a wide range of common problems. However, these datasets cannot fully encompass all the specialized knowledge and technical details of certain specific domains. As a result, although general-purpose foundation models possess broad general knowledge, they lack the necessary depth to meet the complex needs of some specific fields [1]. Therefore, constructing domain-specific foundation models tailored to the needs of particular industries has become particularly important. Domain-specific foundation models, also known as industry-specific foundation models, are developed using data and applications specific to a particular field. Compared to general-purpose foundation models, they are trained with a large amount of domain-specific data, enabling them to more accurately understand and generate domain-specific professional content.

With the widespreading of ChatGPT-like products, the scope of the "foundation model" is gradually expanding. It is thus necessary to first make a clear definition of the foundation model discussed in this article so as to lay the foundation for the subsequent discussion on the customization of domain-specific

arXiv:2409.04267v1 [cs.AI] 6 Sep 2024

Table 1 Examples of foundation models with two different types

Types of foundation models	Examples
Single-modality foundation models	VGG [2], ResNet [3], GPT-1 [4], GPT-2 [5], GPT-3 [6], GPT-3.5 turbo, BERT [7], GLM [8,9], LLaMA [10], LLaMA-2 [11], iGPT [12], LVM [13], SAM [14], BART [15], T5 [16], Time-LLM [17], UniTS [18], ST-LLM [19]
Multi-modality foundation models	CoDi [20], CoDi-2 [21], Claude-3, GPT-4 [22], LLaVA [23], BriVL [24], ImageBind [25], NExT-GPT [26]

foundation models. The foundation models mentioned in this article are neural network models that consists of at least one of the five modules in a general multi-modality foundation models (which will be detailed later). These models also exhibit the following characteristics:

• **Big data**: utilizing a large volume of data covering various scenarios for model training to provide ample knowledge for the model.

• **Large parameters**: the model possesses a huge number of parameters, sufficient to embed the knowledge implied by the big data into the model's parameters.

• **Versatility**: the model's input data format and data processing workflow can adapt to different requirements across various task scenarios.

• **Generalization**: the model exhibits a certain level of generalization, allowing it to perform well even in unknown data domains.

Based on the number of modalities a foundation model can process, they can be classified into single-modality foundation models and multi-modality foundation models, as shown in Table 1.

In the process of constructing domain-specific foundation models, a series of challenges will arise, especially in the stages of data acquisition and preprocessing. For example, the domain-specific data required may not be open-source or readily accessible, as it often has a high degree of confidentiality. Additionally, the modalities of domain-specific data might differ from those used for training general-purpose foundation models, making it difficult to adapt existing models to process this data. Furthermore, the environments where the domain-specific data collected may differ significantly from those for the pre-training datasets, resulting in domain-specific knowledge that pre-trained models are not familiar with. In general, constructing a domain-specific foundation model is challenging and costly, with significant implications for technical security, but it is expected to yield high economic benefits. Therefore, it is essential to thoroughly review and explore the methodologies for constructing these models, providing guideline for both researchers and practitioners.

It is noteworthy that previous review articles have predominantly focused on the development of general-purpose foundation models. Recently, while several review articles have begun to explore the domain-specific adaptation of foundation models, there is a significant gap in the literature for a comprehensive exploration of adaptation strategies that are applicable to foundation models of all modalities, extending beyond language, vision, or any single modality, to various application domains. We summarize representative surveys or review articles on foundation models in Table 2. This paper aims to provide researchers and practitioners interested in building domain-specific foundation model applications with methodological references by introducing the key methodologies therein. Also, practical examples and future directions will be discussed.

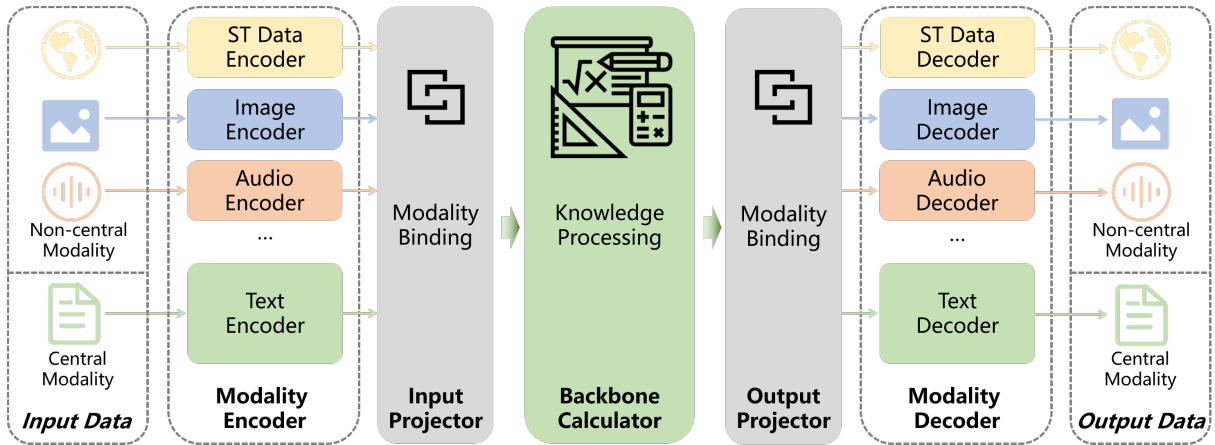## 2 Preliminaries of multi-modality foundation models

In this section, we will elaborate on the preliminary technological basis for customizing domain-specific foundation models. We begin with the architecture of foundation models, detailing all functional modules. Then, from four perspectives - feature extraction, modality alignment, scaling laws, and emergent phenomena - we will explain the foundational technologies that enable each module to contribute to the high performance of foundation models.

2.1 Architecture of foundation models

According to state-of-the-art research on foundation models, it is widely considered that multi-modality foundation models can encompass all functionalities and structures of single-modality foundation models. Essentially, a single-modality foundation model implements only a subset of the functionalities of a multi-modality foundation model.

Table 2   Summary of representative surveys or review articles on foundation models

Article	Modality	Domain Adaptation	Domain	Year
Zhao *et al.* [27]	Language	No	General	2023
Liu *et al.* [28]	Graph	No	General	2023
Mao *et al.* [29]	Graph	No	General	2024
Du *et al.* [30]	Vision-Language	No	General	2022
Yin *et al.* [31]	Multi-modality-Language	No	General	2023
Yeh *et al.* [32]	Time-series Data	No	General	2023
Jin *et al.* [33]	Time-series & Spatio-temporal Data	No	General	2023
Cao *et al.* [34]	Multi-modality	No	General	2023
Zhou *et al.* [35]	Multi-modality	No	General	2023
Zhang *et al.* [36]	Multi-modality-Language	No	General	2024
Lai *et al.* [37]	Language	Yes	Law	2023
Ahn *et al.* [38]	Language	Yes	Mathematics	2024
Li *et al.* [39]	Language	Yes	Finance	2023
Thirunavukarasu *et al.* [40]	Language	Yes	Medicine	2023
Kazerouni *et al.* [41]	Vision	Yes	Medicine	2023
Shahriar [42]	Multi-modality	Yes	Arts	2022
Hou *et al.* [43]	Language	Yes	Software Engineering	2023
Bariah *et al.* [44]	Language	Yes	Telecommunications	2023
Zhou *et al.* [45]	Language	Yes	Telecommunications	2024
Cui *et al.* [46]	Multi-modality	Yes	Autonomous Driving	2024
Yang *et al.* [47]	Language	Yes	General	2024
Ours	Multi-modality	Yes	General	-

Figure 1   The framework of multi-modality foundation models.

The five-module framework proposed for multi-modality foundation models in the paper [36] effectively encompasses the architecture of multi-modality foundation models with language as the central modality. However, the recent emergence of non-language foundation models, including vision foundation models [13,14], graph foundation models [28,29], time series foundation models and spatio-temporal (ST) foundation models [32,33], indicates that the backbone of foundation models is expanding beyond language modalities. Therefore, we propose that the structure of multi-modality foundation models can be divided into the following five modules: Modality Encoder, Input Projector, Backbone Calculator, Output Projector, and Modality Decoder. Figure 1 illustrates the framework of a multi-modality foundation model with language as the central modality.

For multi-modality foundation models, we define the set of all input modalities as M. Generally, a multi-modality foundation model has a central modality C. Using modality alignment techniques, the multi-modality foundation model projects all the modalities it can handle onto this central modality. Below, we define the five modules of the multi-modality foundation model and the input and output data for each module, laying down the foundation for describing the architecture of foundation models in this

paper.

Modality Encoder (ME) encodes the data D_X of an input modality X into a feature vector F_X:

$$F_X = \text{ME}_X(D_X), \quad X \in M. \tag{1}$$

Input Projector (IP) projects the feature vector F_X of a modality X into the feature vector F_C of the central modality C:

$$F_C = \text{IP}_{XC}(F_X), \quad X, C \in M. \tag{2}$$

Backbone Calculator (BC) performs operations on the feature vector F_C of the central modality C, yielding results such as inference and generation $\hat{F}_C$:

$$\hat{F}_C = \text{BC}_C(F_C), \quad C \in M. \tag{3}$$

Output Projector (OP) projects the feature vector $\hat{F}_C$ of the central modality C into the feature vector $\hat{F}_X$ of a modality X:

$$\hat{F}_X = \text{OP}_{CX}(\hat{F}_C), \quad X, C \in M. \tag{4}$$

Modality Decoder (MD) decodes the feature vector $\hat{F}_X$ of the output modality X back into the original data format, resulting in the decoded data $\hat{D}_X$:

$$\hat{D}_X = \text{MD}_X(\hat{F}_X), \quad X \in M. \tag{5}$$

According to the above definition, building a domain-specific foundation model entails choosing the necessary modules — some of which may be optional — and assembling a model tailored to the specific domain requirements, followed by training it with data pertinent to that domain.

2.2 Feature extraction

Feature extraction concerns extracting representative features from raw data. In the field of machine learning, particularly deep learning, feature extraction is a crucial step. As raw data often contains a significant amount of redundant and noisy information, feature extraction, which maps the data into a more information-dense feature space, enables the models to understand data structure and patterns more effectively.

In deep learning, neural networks can perform end-to-end feature extraction from raw data. However, this manner often requires large amounts of data and computational resources to ensure good performance and generalization. In each layer of neural networks, it transfers the output of previous layer to a new vector space. This structure allows for a flexible definition of the output dimensions of each layer without explicitly specifying the transformation. Leveraging these favorable characteristics, autoencoder learns an effective representation of the data by minimizing the reconstruction error between the input vector and the reconstructed vector. Autoencoder first compresses the input data into a low-dimensional feature vector and then projects them back into the original data space by a decoder, as illustrated in Figure 2. An important design principle behind the modality encoder and modality decoder metioned before, is pairing of these components as autoencoders for training. As a variant of autoencoders, variational autoencoder (VAE) aims to both minimize the reconstruction error and maximum the likelihood of the input data to learn the distribution of the compressed vectors. For example, in the image modality, VQGAN [48] is constructed in this manner. It has also been widely applied in subsequent generative models [49, 50].

2.3 Modality alignment

In the workflow of a single-modality foundation model, the architecture excludes input and output projectors, reflecting the absence of a requirement for cross-modality data processing. In contrast, multi-modality foundation models must accommodate the processing of various data modalities, including both the primary and ancillary modalities. To achieve data conversion between modalities through input projectors and output projectors, the key is to apply modality alignment. The goal of modality alignment is to process the feature vectors of different modalities into a common feature space with the same dimension by using a loss function to characterize the correlation between feature vectors. Ideally, modality alignment should ensure that raw data of different modalities carrying the same semantic information are
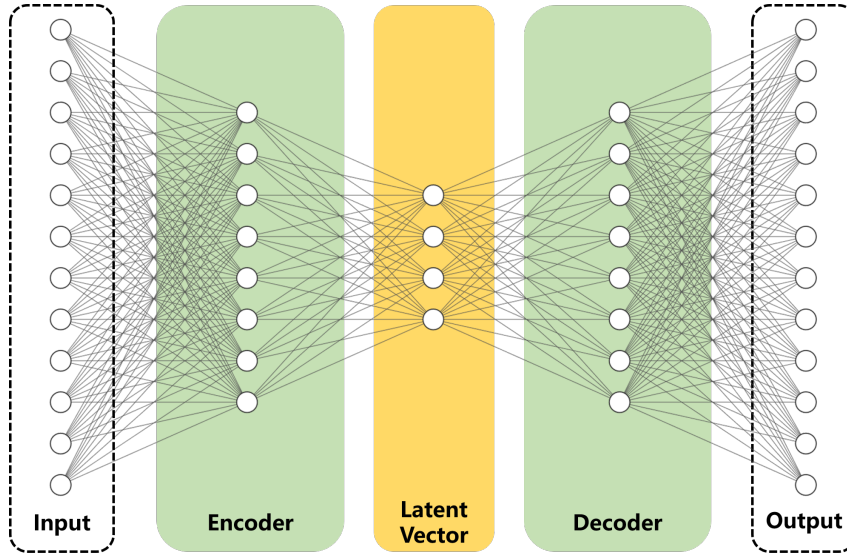
Figure 2   The structure of autoencoder.

represented as the same point in the target feature space, which facilitates the realization of cross-modality information transfer.

There are two main implementation methods for modality alignment: the fusion encoder architecture and the dual encoder architecture [30]:

1. **Fusion encoder**: Fusion encoder encodes the multi-modality data by the self-attention or cross-attention mechanism from transformer [51]. The attention mechanism can be represented as follows:

$$Attenton(Q, K, V) = \text{softmax}(\frac{Q^T K}{\sqrt{d_k}})V, \tag{6}$$

where query Q, key K, and value V are all intermediate features based on the input feature, and d_k refers to the dimension of Q and K. Self-attention-based mechanisms require concatenating the feature vectors of the main and auxiliary modalities as input to the transformer to generate Q, K, and V, allowing the model to automatically focus on the features of different modalities and achieve cross-modality information fusion. For example, VL-BERT [52] concatenates the feature vectors of text and images, and then uses the self-attention mechanism of transformer to aggregate and align language-visual features. On the other hand, methods based on cross-attention mechanism calculate Q, K, and V separately for the feature vectors of the two modalities, thereby achieving cross-modality information fusion. For example, DiT [50] captures the relationship between text and images by self-attention mechanism and then realizes the image generation controlled by text. We illustrates the two fusion encoder architectures in Figure 3.

2. **Dual encoder**: As a multi-modality learning strategy, dual encoder trains a dedicated encoder for each modality independently. The core idea of this architecture is to guide the learning process of the two encoders in synchronization through semantic similarity metrics by leverage contrastive learning methods. Then the output feature vectors of different encoders can be projected into the same vector space. Specifically, this model is based on the hypothesis that if the feature vectors output by the two encoders belong to the same feature space, the feature vectors with paired labels should be closer in the vector space, and vice versa. Through this alignment method, we can expect that the output results of encoders for different modalities describing similar objects or scenes will be close enough. And even in an ideal state, they will converge to the same point in the feature space. The key to achieving this goal is to construct a reasonable model architecture and thoroughly train it on a large-scale dataset. Figure 4 shows the processing logic of the dual encoder.

However, the cost of one-to-one aligning each pair of modalities would be very high. Besides, obtaining a dataset with each pair of modalities aligned is also challenging. To this end, some researchers have proposed the bridging alignment (also known as binding alignment) strategy. This
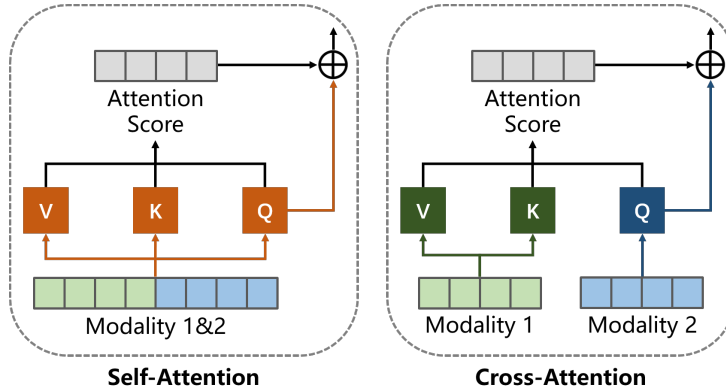
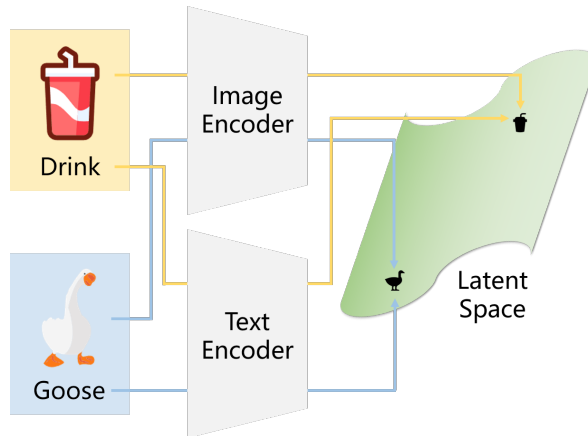**Figure 3** The structure of fusion encoder.

Figure 4 The structure of dual encoder.

method matches all other modalities with a central modality, thereby achieving the alignment of all modalities in the semantic space. For example, ImageBind [25] takes image as the central modality, while CoDi [20] takes text as the central modality. In this way, they effectively simplify the training process of multi-modality alignment and improve the practicality and efficiency of the model. In contrastive learning, the class imbalance in long-tailed datasets can lead to reduced effectiveness, as most contrastive pairs consist of major classes instead of tail classes, making the model ineffective for the minor classes. Therefore, [53] proposed a method to solve this problem.

2.4   Scaling law

Scaling law refers to the mathematical pattern about how the performance of a system changes as the scale of the system increases. In the field of AI, especially in the research and application of foundation models, scaling law describes a series of rules and phenomena about how model performance changes as the model scale expands, including the number of parameters, dataset size, computational resources. It uses quantitative analysis methods to reveal the intrinsic mechanism of the performance improvement of foundation models.

[54] discussed how the inductive biases of different models affect the relationship between model scale expansion and performance. They found that model architecture is indeed one of the key factors affecting the benefits of model expansion. They also pointed out that although the standard transformer architecture may not always achieve the best performance, it does exhibit the best scalability. In the fields of computer vision [55] and natural language process [56], models based on the transformer architecture have shown an exponential relationship between model scale and model performance.

Besides, [57] examined the impact of the number of downstream tasks and model scale on the performance of instruction fine-tuning. They fine-tuned the models on a wide variety of tasks by a multi-task

joint training method. As a result, the language model could learn a broader language representation and knowledge, thereby enhancing its generalization ability on unseen tasks. During the joint training process, knowledge transfer between different tasks is promoted through parameter sharing, significantly improving the model's generalization ability and performance. In addition, joint training also reduces the time and computational resources required to train each task individually, improving training efficiency. This phenomenon of model performance improving as task diversity increases is a manifestation of the scaling law. [58] verified the phenomenon of model performance increasing with the number of tasks on the large-scale benchmark OPT-IML Bench. Additionally, there are also studies that have separately provided the model performance of natural language models [56] and autoregressive generative models of various modalities [59] at different scales.

Although there is no unified form for the quantitative representation of the scaling law, it can be generally represented as an exponential relationship between the model loss function and the model's parameters, dataset size, and computational resources. We use the loss function $L(\cdot)$ to characterize the model's performance, where a smaller loss function value represents better model performance. Eq. (7) describes the performance of a model with a given number of parameters trained to convergence on a sufficiently large dataset, where $L(N)$ is the loss function, N is the number of trainable parameters of the model, N_c is a constant, and α_N is the power law exponent. Eq. (8) provides the performance of a suitably sized model trained on a sufficiently large dataset under a given computational resource constraint, where $L(C)$ is the loss function, C is the given computational resource, C_c is a constant, and α_C is the power law exponent. Eq. (9) describes the performance of a foundation model trained with early stopping strategy on a given dataset size, where $L(D)$ is the loss function, D is the dataset size (in tokens), D_c is a constant, and α_D is the power law exponent.

$$L(N) \propto \left(\frac{N_c}{N}\right)^{\alpha_N}, \tag{7}$$

$$L(C) \propto \left(\frac{C_c}{C}\right)^{\alpha_C}, \tag{8}$$

$$L(D) \propto \left(\frac{D_c}{D}\right)^{\alpha_D}. \tag{9}$$

According to the above equations, when not constrained by other conditions, the model's loss function decreases exponentially as the number of parameters, computational resources, and training data volume increase. This means that by increasing the model's parameters, investing more in computational resources, and expanding the training data volume, the model's performance can also be exponentially improved.

2.5 Emergent phenomenon

The scaling law reveals that the scaling up of model size can lead to incremental improvements in model performance. On the contrary, the emergent phenomenon refers to the new properties exhibited by the model after reaching a critical point of scale expansion, one of which is a significant improvement in model performance [60].

The emergent phenomenon essentially reveals the source of the superior performance of foundation models. In the field of deep learning, especially in the domain of LLM, the emergent phenomenon has been widely observed. For example, models like LLaMA have demonstrated exceptional comprehension, generation capabilities, and even a certain degree of logical reasoning ability, which small language models cannot achieve. As the model scale increases, the model can have more parameters and a more complex structure, allowing it to capture the complex features and patterns in the data. Foundation models often exhibit strong generalization capabilities, which is mainly because their abundant parameters can store rich knowledge, enabling them to make accurate inferences and predictions on unseen data. It can provide them adaptability and universality to different tasks, and even allows the model to truly learn the underlying principles and reasoning methods in the data. [60] suggested that the emergence point of the emergent phenomenon in LLMs can be influenced by the task and the prompting method used. Specifically, the use of a chain-of-thought prompting approach can significantly enhance the LLMs' ability to handle complex reasoning tasks [61], thereby bringing forward the emergence point of the emergent phenomenon.

Table 3 Customization methods of domain-specific foundation models

Customization Method	Customization Degree	Difficulty	Flexibility	Computation
Based on general-purpose foundation models	Low, only the input method of domain knowledge is customized	Low	Low	Low
Based on pre-trained modules	Medium, some modules can be customized	Medium	Medium	Medium
Without pre-trained modules	High, each module can be customized	High	High	High

When building domain-specific foundation models, developers must select an appropriate model scale, considering both the requirements of potential downstream tasks and user prompting patterns. An increase in the number of model parameters raises the demand for computational resources and the risk of overfitting. Consequently, there is a limit to the extent that parameters can be increased. The trade-offs highlighted by this phenomenon are crucial for model deployment, offering significant insights for both model design and application.

3 Key technologies for building domain-specific foundation models

This section delves into the technical pathways toward customizing domain-specific foundation models. We will provide a detailed explanation of how to flexibly select and combine the appropriate modules from five key components — modality encoders, input projectors, backbone calculators, output projectors, and modality decoders based on the specific requirements of different domains. Additionally, we will analyze concrete cases to help readers better understand and apply the methodologies discussed in this section.

We can categorize the customization of domain-specific foundation models into three levels, ranging from low to high level of customization (i.e., from high to low reliance on general-purpose foundation models or pre-trained modules):

1. Domain-specific enhancement based on general-purpose foundation models.

2. Customization of the foundation model based on pre-trained modules.

3. Construction of the foundation model without pre-trained modules.

Table 3 summarizes the characteristics of these three domain-specific foundation model customization methods.

3.1 Domain-specific enhancement based on general-purpose foundation models

General-purpose foundation models offer comprehensive capabilities, making them suitable for a wide array of task scenarios. When such a general-purpose foundation model can fully handle the required data modalities, it renders unnecessary any modifications to its underlying architecture for model developers. Instead, they can focus on implementing domain-specific enhancements.
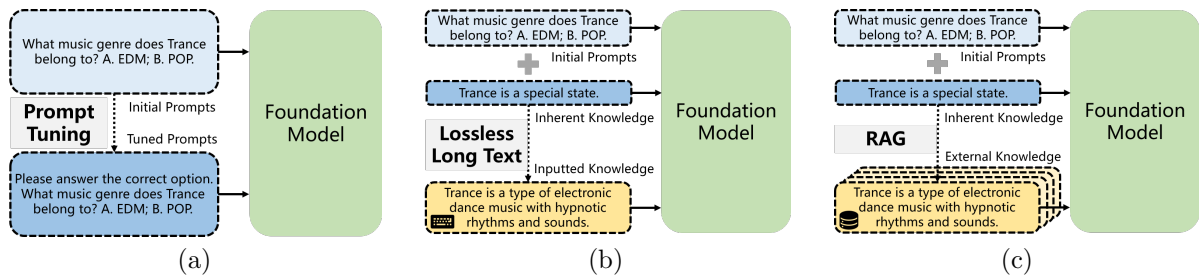
Depending on whether the domain-specific enhancement requires altering the parameters of the foundation model, we can further divide this into two types: plug-and-play domain-specific enhancement and fine-tuning-based domain-specific enhancement. Table 4 categorizes and summarizes the methods of domain-specific enhancement based on a full-architecture general-purpose foundation model.

3.1.1 *Plug-and-play domain-specific enhancement*

The generality, generalization capabilities, and reasoning abilities of general-purpose foundation models enable them to serve as the foundation for domain-specific models. To achieve plug-and-play domain enhancement without modifying the parameters of the foundation model, two approaches can be utilized: **leveraging existing knowledge** or **embedding new knowledge**. The first approach aims to leverage domain knowledge already stored within the general-purpose foundation model, as illustrated in Figure 5 (a). The second approach, embedding new knowledge, involves equipping the foundation model with the ability to handle domain tasks by introducing domain-specific knowledge. This can be further divided into two methods: embedding knowledge through prompts and embedding knowledge via an external knowledge base. These methods are depicted in Figure 5 (b) and (c) respectively. The following sections will provide a detailed explanation of these techniques.

Table 4   Specific-domain enhancement with the entire general-purpose foundation models

Customization Method			Parameter Modification	Addition of New Modules	Domain Knowledge Provider	Specific Techniques
Plug -and -play	Utilizing Existing Knowledge	Hard Prompting	No	No	Deployer	PET [62]
		Soft Prompting	No	Yes	Deployer	Prefix Tuning [63], P-tuning [64]
	Adding New Knowledge	Prompts	No	No	User	LongRoPE [65], Transformer-XL [66]
		External Knowledge Base	No	Yes	Deployer	RAG [67], GraphRAG [68], RankRAG [69]
Fine -tuning -based	Adapter-Based		Yes	Yes	Deployer	Adapter [70], AdapterFusion [71], IA3 [72], Model Reprogramming [73]
	Low-Rank Matrix Decomposition		Yes	Yes	Deployer	LoRA [74], LoHa [75], LoKr [76]
	Full Parameter Fine-tuning		Yes	No	Deployer	PEFT [77]

Figure 5   Plug-and-play domain-specific enhancement. (a) Invoking existing knowledge; (b) Knowledge embedding by prompts; (c) Knowledge embedding by external knowledge base.

1. **Invoking existing knowledge for domain enhancement**: During the training process, a general-purpose foundation model may have already enclosed domain knowledge. Prompt tuning improves prompts to better invoke the model's inherent domain knowledge, where "tuning" refers to the optimization of prompts. Specifically, it involves inserting a carefully crafted prompt into the input data as context to improve the generated output. These carefully crafted prompts can be natural language descriptions, examples, rules, or other text or embedding vectors that guide the model to understand the task requirements. When generating outputs, the model will consider these carefully crafted prompts to produce task-relevant results. Prompt tuning can be categorized into hard prompts and soft prompts:

   (a) **Hard prompts**: Hard prompt methods are common techniques in natural language processing (NLP). They guide the language model's output using interpretable and reusable handcrafted words and tokens. Hard prompts are typically manually designed and tailored for specific tasks, making them difficult to modify. PET (Pattern Exploiting Training) [62] is a classic hard prompt learning method that models questions as cloze tasks and optimizes the final output words. This method trains the model on a small amount of supervised data and performs ensembling predictions on unsupervised data to guide the model.

   (b) **Soft prompts**: Designing hard prompts requires experimental exploration and expertise, and manually designed prompts may not align well with model's data processing methods. To simplify this process and enhance the flexibility of prompt tuning, researchers proposed soft prompt-based tuning methods. Prefix Tuning [63] is a form of soft prompt tuning that adapts to specific downstream tasks by adding learnable prefix vectors (soft prompts) to the beginning of the input sequence. These prefix vectors, as part of the input, guide the model's output to meet task requirements. The advantage of prefix tuning is that it only updates these prefix vectors rather than the model's parameters, significantly reducing computational and storage resource demands while retaining the rich knowledge learned by the pre-trained model.

Building on prefix tuning, researchers introduced the P-tuning method [64]. P-tuning replaces fixed or manually designed words and tokens with learnable soft prompts. Its core idea is to treat prompts as part of the model that can be learned, allowing the model to learn not only how to respond to given tasks but also how to generate the best prompts. These soft prompts are typically a series of embedding vectors processed alongside the actual text input. Through end-to-end training, the model automatically learns to adjust these embedding vectors for better task performance. P-tuning combines the parameter efficiency of prefix tuning with the flexibility of traditional hard prompt tuning. Soft prompts give the model greater freedom to generate answers, potentially producing more diverse outputs but also increasing the risk of generating inaccurate or irrelevant responses.

2. **Knowledge embedding for domain enhancement**: When the existing knowledge of a general-purpose foundation model is insufficient to solve domain tasks, introducing new knowledge by embedding additional background information can achieve higher quality output. This method is known as knowledge embedding for domain enhancement.

   (a) **Knowledge embedding by prompts**: Prompts, serving as a direct interface between the user and the large language model, can be used to incorporate domain knowledge. However, the method of knowledge embedding through prompts has a significant limitation: the amount of embedded domain knowledge is restricted by the maximum prompt length of the model. The limitation on the model's ability to process longer text inputs stems from three core issues of the Transformer architecture:

   • **Limitations of positional encoding**: Transformer models typically generate fixed-length positional encodings using sine and cosine functions, where each position in the sequence is uniquely encoded. However, when the sequence length exceeds the maximum length used during training, the model cannot effectively handle the additional text because it cannot generate valid encodings for the new positions.

   • **Resource consumption of the attention mechanism**: The attention mechanism is the core of Transformer models, allowing the model to compute attention weights for each element in the sequence. However, as the sequence length increases, the computational complexity and memory requirements of this mechanism grow quadratically, leading to significant resource consumption.

   • **Long-distance dependency issue**: When handling long sequences, the Transformer needs to span a large number of input tokens, which often results in problems such as gradient vanishing or exploding. This makes it difficult for the model to capture dependencies between elements that are far apart in the sequence.

   To address the above issues, Lossless Long Text technology has emerged. It aims to enhance the model's ability to process long texts that exceed its input length limitations, allowing users to input a large amount of domain knowledge directly through prompts into the large language model as contextual information for domain enhancement. Lossless Long Text technology expands the long text input capability of large language models in two directions: extrapolation and interpolation:

   i. **Extrapolation**: Extrapolation involves extending the model's context window to handle new texts that exceed the length of the training data. This typically involves improving the positional encoding mechanism so the model can understand and process longer sequences. Longformer [78] extends the ability to handle long texts effectively by combining local and global attention mechanisms; BigBird [79] uses sparse attention mechanisms and reversible layers to extrapolate the model's long-sequence processing capabilities; LongRoPE [65] improves positional encoding by introducing rotational transformations in self-attention, allowing the model to handle long-distance dependencies and support inputs up to two million tokens without impacting computational efficiency.

   ii. **Interpolation**: Interpolation refers to enhancing the model's ability to process long texts within its existing sequence length capacity by adjusting and optimizing the attention mechanism. This typically involves improvements to the attention mechanism so the model can more effectively handle long-distance information. The BERT model [7] enhances text understanding through pre-training with a bidirectional Transformer. XLNet [80]

improves long text processing by using permutation language modeling and generalized autoregressive pre-training to enhance the model's internal representations. Transformer-XL [66] is an improved Transformer model that introduces a recurrence mechanism to address the issue of gradient vanishing in long text processing. This allows the model to retain information from previous sequences while processing the current sequence, thereby better understanding and generating long text content.

(b) **Knowledge embedding by external knowledge base**: In practical applications, users may not be able to provide sufficient domain knowledge to enhance a general-purpose foundation model. To address this issue, model deployers can augment the general-purpose foundation model with a dedicated domain knowledge base. This method allows the general-purpose foundation model to reference this external knowledge base when generating answers or performing tasks, thereby obtaining the necessary domain information and context to provide more accurate and targeted responses or solutions. Retrieval-Augmented Generation (RAG) [1, 67, 81] technology was developed to achieve this purpose. RAG technology aims to enhance the language model's generation capabilities by leveraging an external document base without retraining the model. It is particularly suitable for tasks requiring a customizable dynamic knowledge base, such as question answering, text summarization, and fact-checking. The core of RAG technology is the integration of a retrieval component that can quickly find information relevant to the current task in a large document database during the generation process. Once the relevant documents are retrieved, this information is used as additional contextual information to aid the generation process. The advantage of RAG technology is that it combines the generation capabilities of large language models with the knowledge provided by external retrieval systems, without requiring domain knowledge themselves. Furthermore, because the external knowledge base can be replaced as needed, RAG technology offers high flexibility and adaptability.

In RAG, the two core challenges are how to retrieve useful information from the database and how to organize that information to augment the generation of foundation models. For the retrieval task, the earliest and most naive approach uses sparse retrieval that directly matches data based on raw data like BM25 [82]. Inspired by the field of information retrieval, dense retrieval like DPR [83] has been proposed, which projects raw data into a high-dimensional space, potentially helping to capture semantic similarity better. However, a significant challenge is that even if the retrieved data is highly related to the original input in the semantic space, we cannot guarantee that it will help the model generation, due to issues like polysemy. To address this problem, some researchers have introduced other technologies like knowledge graphs to aid the retrieval process [68]. [84] showed that adding noise can help enhance the model performance compared to traditional dense retrieval methods. Additionally, the organization of the retrieved information can also directly influence the output quality. For example, providing a ranking of the retrieved data [69] can improve model performance.

While the aforementioned technologies were initially proposed to enhance large language models in specific domains, their applications are not limited to language models. With the development of foundation model domains, these technologies are expected to be extended to foundation models of other modalities.

3.1.2 *Domain-specific enhancement based on fine-tuning*

When plug-and-play domain enhancement techniques are difficult to implement or require embedding too much domain knowledge into the general-purpose foundation model, or when deep modifications to the general-purpose foundation model are necessary, we can turn to the strategy of domain enhancement based on fine-tuning. This strategy aims to customize the required domain-specific foundation model while preserving the pre-trained knowledge of the general-purpose foundation model as much as possible by specific domain enhancement [85].

Fine-tuning techniques can be divided into three main types: adapter-based fine-tuning, low-rank matrix decomposition-based fine-tuning, and full-parameter fine-tuning. Figure 6 (a), (b) and (c) illustrate these three technical pathways respectively. In following sections, this paper will elaborate on these techniques, sorted from low to high resource requirements and complexity needed for fine-tuning.
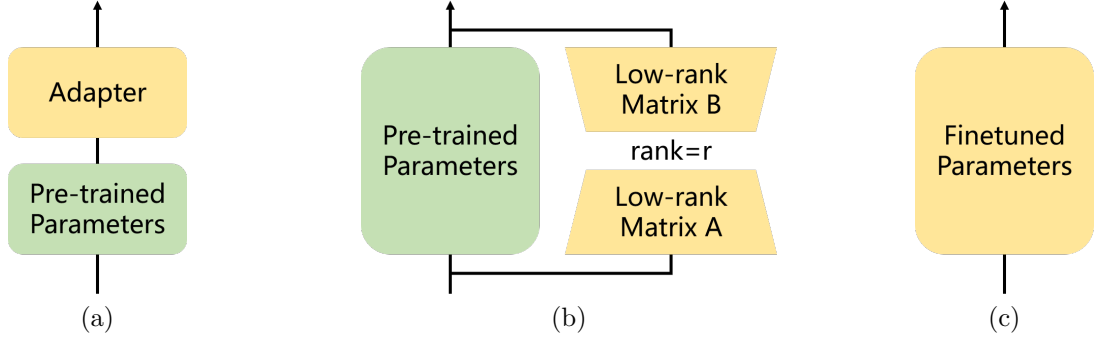
Figure 6   Fine-tuning-based domain-specific enhancement. (a) Adapter-based tuning; (b) Low-rank-decomposition-based tuning; (c) Full fine-tuning.

1. **Adapter-based fine-tuning**: Adapter-based fine-tuning [70] is a method that inserts small trainable adapter modules into pre-trained models, aiming to efficiently adapt the model to specific downstream tasks. During fine-tuning, only parameters of adapter modules are updated, while the original parameters of the pre-trained model remain unchanged, reducing computational resources and storage requirements while retaining the rich knowledge learned by the model during pre-training. AdapterFusion [71] is an extension of adapter-based fine-tuning that allows the model to learn multiple tasks or adapt to various data distributions simultaneously by fusing multiple adapter modules, with each adapter module focusing on capturing task-specific features. Infused Adapter by Inhibiting and Amplifying Inner Activations (IA3) [72] scales the activation layers by injecting learned vectors into the attention and feed-forward modules of the Transformer architecture. Since these learned vectors are only trainable parameters during fine-tuning, IA3 significantly reduces the number of trainable parameters compared to traditional adapter-based fine-tuning, thereby reducing training costs and improving training efficiency. Additionally, IA3 does not introduce inference latency because its adapter weights can be merged with the foundation model while maintaining the flexibility and adaptability of the model, allowing customized fine-tuning for different tasks and datasets. [73] proposes a method called Model Reprogramming that converts both the input and output domains of a foundation model by adding two adapter layers, enabling more flexible adaptation functionality.

2. **Low-rank matrix decomposition based fine-tuning**: Fine-tuning based on low-rank matrix decomposition reduces the number of parameters that need to be updated by decomposing the weight matrices in the pre-trained model into the product of low-rank matrices. Low-rank matrix decomposition can capture the most important information in the weight matrices while keeping the original pre-trained parameters unchanged during fine-tuning, updating only low-rank decomposition matrices, thus reducing the computational and storage requirements during fine-tuning. This method improves fine-tuning efficiency while maintaining or approaching the performance of full-parameter fine-tuning.

   (a) **Low-Rank Adaptation (LoRA)**: Low-Rank Adaptation (LoRA) [74] achieves efficient fine-tuning performance by decomposing model parameters into the product of low-rank matrices using singular value decomposition (SVD). The principle of LoRA can be represented by the following equation:

   $$W_{\text{new}} = W_{\text{old}} + \Delta W, \tag{10}$$

   where W_{old} is the original weight matrix, and ΔW is the low-rank update matrix, which can be constructed by selecting the singular vectors corresponding to smaller singular values of W_{old}. The singular value decomposition of W_{old} is given by:

   $$\Delta W = U\Sigma V^T. \tag{11}$$

   Here, U and V are matrices obtained from the SVD of W_{old}, and Σ is a diagonal matrix containing the most important singular values of W_{old}. By updating only the parameters in U, Σ, and V, LoRA efficiently fine-tunes the model.

One limitation of LoRA is that it typically applies the same low-rank structure to all layers, ignoring the varying importance of different layers and parameters for downstream tasks. Adaptive Low-Rank Adaptation (AdaLoRA) [86] is an improved method based on LoRA, which adaptively determines which layer parameters need to be updated. By employing adaptive learning rates and task-specific parameter adjustment strategies, AdaLoRA enables the model to automatically adjust the intensity and scope of fine-tuning according to the specific requirements of the task.

Researchers have also found that LoRA's continuous pre-training performance is unsatisfactory on some large-scale datasets. Thus, Layerwise Importance Sampled AdamW (LISA) [87] strategy is proposed, where the weight norm distribution of different layers exhibits uncommon skewness. LISA adopts importance sampling strategy by randomly activating different layers in the foundation model for optimization. Specifically, LISA consistently updates the bottom embedding layers and the top linear head while randomly updating a small number of intermediate self-attention layers. This method, with memory consumption comparable to LoRA, outperforms LoRA and even full-parameter fine-tuning in various downstream fine-tuning tasks.

(b) **Low-Rank Hadamard Product (LoHa)**: Low-Rank Hadamard Product (LoHa) [75] updates the model's weights by introducing the Hadamard product of low-rank matrices. The principle of LoHa can be represented by the following equation:

$$W_{\text{new}} = W_{\text{old}} \odot \Delta W, \tag{12}$$

where W_{old} is the original weight matrix, and ΔW is the update matrix approximated by a low-rank matrix. The update matrix ΔW can be further decomposed into the Hadamard product of two low-rank matrices:

$$\Delta W = L_1 \odot L_2. \tag{13}$$

Here, L_1 and L_2 are two low-rank matrices that adjust elements of the original weight matrix by learning key information extracted from the input data. By updating only parameters in L_1 and L_2, LoHa achieves efficient fine-tuning of the model while maintaining its adaptability to new tasks.

(c) **Low-Rank Kronecker Product (LoKr)**: Low-Rank Kronecker Product (LoKr) [76] is another parameter-efficient fine-tuning method that emerged after LoHa. LoKr utilizes the properties of the Kronecker product to expand the dimensions of the weight matrix while keeping the increase in the number of parameters within a manageable range. The Kronecker product allows the model to learn complex interactions across different dimensions, which is particularly useful for capturing high-order relationships in the input data. The updating process of LoKr can be represented as:

$$W_{\text{new}} = W_{\text{old}} + \Delta W, \tag{14}$$

where ΔW is the Kronecker product of two low-rank matrices L_1 and L_2:

$$\Delta W = L_1 \otimes L_2. \tag{15}$$

In this formula, L_1 and L_2 are two low-rank matrices, and they compute a large matrix that is updated during the fine-tuning process through Kronecker product. LoKr is particularly suitable for tasks that require increasing the model's dimensions to capture more complex relationships, while maintaining similar parameter efficiency to LoHa. However, LoKr may require more complex mathematical operations to handle the Kronecker product, and in some cases, its computational cost may be higher than that of LoHa.

3. **Full fine-tuning**: Full fine-tuning is not constrained by pre-training tasks or data distributions, making it flexible to adapt to various downstream tasks. It allows the model to be directly optimized end-to-end on the data of the final task without the need for additional adapter modules. However, because it requires updating all parameters in the model, full fine-tuning demands significant computational resources and longer training times. Moreover, foundation models have a vast number of

parameters, and insufficient fine-tuning data may lead to overfitting. Additionally, the intermediate variables generated during full fine-tuning consume a large amount of GPU memory. Therefore, researchers have proposed many parameter-efficient fine-tuning methods mentioned earlier, which can reduce resource consumption and training time while maintaining performance [77].

3.2 Customization of the foundation model based on pre-trained modules

Foundation models may contain millions or even billions of parameters, and through transfer learning, it's possible to reduce parts of the model that need training, thereby significantly lowering training costs. This approach is known as the customization of the foundation model based on pre-trained modules. The essence of transfer learning lies in utilizing the knowledge embedded in model parameters during the pre-training process when constructing a new model. As mentioned earlier, in the architecture of foundation models, there are typically five main modules: modality encoders, backbone calculators, modality decoders, input projectors, and output projectors. Among them, the modality encoder, backbone calculator, and modality decoder carry a large amount of knowledge as they are directly involved in the encoding, processing, and decoding of data. In contrast, the input projector and output projector themselves carry less model knowledge. In some cases, they may not even have explicit models responsible for these functions, or these modules are only trained when building new foundation models. Therefore, when customizing foundation models, we generally do not choose to transfer the input projector and output projector.

Next, this paper will detail how to customize foundation models based on the pre-trained modality encoder, backbone calculator, and modality decoder. Through this approach, we can effectively leverage the knowledge of pre-trained models while reducing the computational resources demand, making the model more suitable for specific tasks and environments.

3.2.1 *Transfer learning based on pre-trained modality encoders*

General-purpose foundation models often adapt to the distribution characteristics of a vast dataset during training, internalizing ample domain knowledge in their model parameters. They are thus well-suited as feature extraction modules for customized foundation models. By utilizing the pre-trained model's pre-existing feature extraction module as the modality encoder for domain data, downstream task modules can be seamlessly integrated after this module to fulfill task requirements. The modality encoder encapsulates knowledge about crucial features of the data. There are two main approaches to transfer learning for obtaining modality encoders:

• **Transfer within the same modality across different data domains**: This approach involves transferring the knowledge of the modality encoder trained on the source data domain to the target data domain. It typically entails aligning the feature distribution of the source and target domain data. Fine-tuning the pre-trained modality encoder from the source domain on the target domain enables it to adapt to the characteristics of the new data. Specifically, this can be achieved by adjusting or adding layers to the encoder. Additionally, domain adaptation techniques such as domain adversarial training or domain-invariant feature extraction techniques can be employed to reduce the distribution discrepancy between the source and target domains.

• **Transfer across different modalities**: When it comes to multi-modality foundation models, there are often cases that lacking of corresponding pre-trained encoders for certain modalities. In such situations, a cross-modality transfer strategy can be utilized, adapting encoders of other modalities to process new data modalities. For instance, the authors of ImageBind treat depth and thermal imaging data as a form of single-channel images, thereby using image encoders to extract features for thermal data. Initializing the model with weights pre-trained on image datasets can lead to faster convergence compared to random initialization and can enhance generalization to some extent.

3.2.2 *Transfer learning based on pre-trained backbone calculator*

For multi-modality foundation models, the backbone calculator is the core computational component responsible for processing encoded feature vectors and performing tasks such as classification and generation. The backbone calculator of customized foundation models can be transferred from pre-trained models to leverage the complex feature processing and task execution capabilities learned from large-scale datasets. This approach avoids training the backbone calculator from scratch but still requires

constructing appropriate previous modules to encode the data into feature vectors that can be processed by the backbone calculator. For example, NExT-GPT [26] converts raw data from various modalities into language modality feature vectors before feeding them into a pre-trained LLM, allowing the model to process inputted tokens according to task requirements. There are two main approaches to transferring pre-trained backbone calculators:

• **Transfer of a single pre-trained backbone calculator**: Utilizing a general-purpose foundation model (e.g., LLaMA) as the backbone calculator to process data from the central modality. When transferring a pre-trained backbone calculator to a specific domain application, it typically requires fine-tuning to adapt to specific data characteristics and task requirements of that domain. This step can be performed on a limited domain dataset, fine-tuning the model parameters to optimize its processing capabilities for the domain-specific data.

• **Modular combination of multiple pre-trained backbone calculators**: Modular combination is a flexible design method in deep learning architectures that allow integrating multiple specialized pre-trained models into a unified framework based on task requirements. The Mixture of Experts (MoE) model [88] can serve as an effective mechanism to further optimize this modular combination. The MoE model introduces multiple expert networks and uses a gating mechanism and mixing strategy to dynamically select and combine outputs of these experts, enabling specialized processing for different tasks or data subsets.

The primary function of the gating mechanism is to determine how input data should be distributed among different experts. It generates a weight or score for each expert based on the characteristics of the input data, reflecting each expert's capability or suitability for processing the current input. The output of the gating mechanism is typically used to guide the mixing strategy, indicating the importance of each expert for the current input. The mixing strategy then combines the outputs of multiple experts according to specific rules to generate the final model output. This strategy can be simple, such as averaging or weighted averaging, or more complex, involving probability distributions of model outputs or other advanced methods.

For example, for complex tasks requiring both image recognition and language understanding, one expert network might excel at identifying edges in images, while another expert network might be adept at understanding semantic relationships in natural language. The gating mechanism of the MoE model can automatically adjust the participation level of each expert network based on the characteristics of the input data and the task requirement, which allows the model to flexibly invoke the most appropriate expert network when dealing with mixed visual and language inputs, achieving optimal performance. Moreover, MoE models are highly scalable. They can adapt to new task requirements or data types by adding new expert networks and updating the gating mechanism, making them suitable for constructing flexible customized foundation models.

3.2.3   *Transfer learning based on pre-trained modality decoders*

Modality decoders play a crucial role in multi-modality foundation pre-trained models. They are responsible for converting processed feature vectors back into the form of the original data. In generative tasks, such as converting text to images or audio to text, modality decoders need not only to accurately decode the feature vectors to reconstruct understandable original data but also to exhibit a certain level of creativity. Some pre-trained modality decoders can also understand and process multi-modality feature inputs. For instance, CoDi-2 can utilize both text and audio as conditions to control image generation. By transferring such pre-trained decoders, there is no need to train complex decoder structures from scratch, allowing them to be directly applied to image generation tasks.

Here are methods to effectively utilize pre-trained modality decoders for transfer learning:

1. **Fine-tuning pre-trained modality decoders**: Similar to modality encoders, modality decoders can be fine-tuned on a specific task's dataset to adapt to new task requirements. This process usually involves adjusting the last few layers of the decoder or adding new layers to better capture the data characteristics of the specific domain.

2. **Transferring cross-modality generative modality decoders**: In cross-modality generation tasks, pre-trained modality decoders can be directly used to generate data in the target modality. The conditional information is first encoded into feature vectors through a conditional encoder, and then combined with feature vectors of the original data to achieve conditional generation. The

prerequisite for this functionality is ensuring that input feature vectors can be correctly understood by the decoder, which may involve adjustments to the backbone calculator and output projector.

3.3 Construction of the foundation model without pre-trained modules

When it is not possible to construct a foundation model through transferring from pre-trained models, it becomes necessary to design and train the corresponding modules. We will first provide a general analysis about the architectures of single-modality and multi-modality foundation models, as the foundation for constructing each components.

Single-modality foundation models consist of three core modules: the modality encoder, the backbone calculator, and the modality decoder. For example, in LLaMA-2 [11], the modality encoder and decoder are specifically designed for the language modality, utilizing the byte pair encoding (BPE) algorithm for encoding and decoding functions. And the backbone calculator is a massive autoregressive Transformer model. By this way, LLaMA-2 achieves a complete processing workflow of "input raw text – input text feature vectors – output text feature vectors – output raw text". Additionally, Bai et al. [13] introduces the concept of visual sentences, and proposes a large visual model (LVM) that can autoregressively generate the required image based on visual sentences. It realizes in-context learning within the pure image modality, which enables the model to directly infer tasks from image modality prompts and generate corresponding results. This not only explores the potential of pure visual input but also provides a new perspective for constructing domain-specific foundation models — the central modality does not have to be limited to language but any modality widely used in a specific domain.

Multi-modality foundation models require the additional input projectors and output projectors to achieve modality alignment. For instance, CoDi-2 [21] first utilizes multiple modality encoders proposed in ImageBind [25] to process input data, aligning all corresponding modalities to the image modality. Then, the feature vectors of the image modality are transformed into the feature space of the language modality through a multi-layer perceptron (MLP). In detail, it uses the pre-trained autoregressive Transformer of the LLM LLaMA-2-7b-chat-hf as the backbone calculator's foundation. The image and audio features processed by the backbone calculator are converted back to the image domain via two MLPs. And they are then used as control vectors input of a Diffusion-based generative model to obtain the final image and text results. The training losses include text generation loss, modality conversion loss, and data generation loss. So that the multi-modality feature processing capability of the backbone calculator and the modality conversion capability of the two MLPs can be trained at the same time by an end-to-end way. The model's modality alignment is reflected in two terms. On one hand, the model aligns the feature vectors of multiple modalities to the image modality through the pre-trained modality encoders of ImageBind. On the other hand, it also converts between image feature vectors and text feature vectors via the MLP.

In summary, constructing a foundation model begins with determining the data modalities and selecting the central modality. Next, the modality encoder and input projector are implemented to convert raw data from different modalities into central modality feature vectors, which will then be processed by the backbone calculator. Following this, output projector and modality decoder are designed to convert the feature vectors from the backbone calculator back into the original data forms of each modality. Once the model structure is constructed, the training process can begin. In the following, we will provide a detailed introduction to the implementation principles and construction methods of each module.

3.3.1 *Constructing modality encoders*

Constructing a modality encoder means designing a neural network capable of extracting feature vectors from data. The general steps for building a modality encoder are as follows:

1. **Preprocessing into suitable data structures**: Choose an appropriate data structure based on the characteristics of the data modality for subsequent model usage. For example, in audio processing, a common approach is to convert time-domain signals into spectrograms and then use neural network designed for images to extract features. And another method is to view audio as sequential vectors and use neural networks for sequence like time series or neural language to process it. When selecting the target data structure, researchers need to balance task requirements and processing difficulty, ensuring that the data structure both represents domain knowledge sufficiently and is suitable for downstream model processing. Additionally, since domain-specific foundation models

Table 5 Comparison between encoder-only structure, decoder-only structure and encoder-decoder structure

Model Architecture	Generative Capability	Understanding Capability	Computation	Examples
Encoder-only	Low	High	Low	BERT [7]
Decoder-only	High	Low	Low	GPT Series [4–6, 22], LLaMA Series [10, 11]
Encoder-decoder	High	High	High	BART [15], T5 [16]

      need to be functionally versatile, the compatibility of various task inputs should be considered when choosing the target data structure.

2. **Designing the network architecture**: Design the network architecture according to the characteristics of the input data structure. For example, Transformer architectures can be used for text data to capture long-range dependencies, while CNN-based or ViT-based models can be employed for image data to extract features.

3. **Training the modality encoder**: Pre-train the modality encoder using a dataset with sufficient quantity and variety of samples to learn the general features and distributions of the modality data. Pre-training is the process of infusing the model with knowledge. If the dataset size or diversity is inadequate, the model might not learn a complete representation of the modality data. One method to train a modality encoder is to combine the modality encoder and modality decoder into an autoencoder and perform unsupervised training by minimizing reconstruction error. Another training method is designing and training a model for a specific task by supervised training, and then transferring the upstream part of the model as the modality encoder. However, this method cannot get a compatible modality decoder, which might affect the design and functionality of subsequent modules. Therefore, when designing the modality encoder, it is essential to consider the consistency of the entire foundation model architecture.

3.3.2 *Constructing input projectors*

The role of input projectors is to project data from different modalities into a common feature space. As discussed in section 2.3, modality alignment can be achieved through either fusion encoders or dual encoders. When constructing input projectors, the key decision is whether to use a bridging strategy to integrate input vectors from different modalities or to use a fine-tuning approach to bring the projectors of different modalities closer together. These two strategies correspond to the concepts of fusion encoders and dual encoders, respectively. During training, using loss functions from multi-modality understanding tasks, such as multi-modality classification or generation task losses, can train the model's cross-modality projection capability. Additionally, an end-to-end training approach can optimize the overall performance of the foundation model and train cross-modality projections at the same time. As previously mentioned, the CoDi-2 model [21] utilizes ImageBind [25] encoders aligned by CLIP as the image and audio modality encoders and part of the input projector. It then incorporates an MLP as another part of the input projector. During the end-to-end training process of the foundation model, the MLP is optimized, so as to align the image and audio to text.

3.3.3 *Constructing the backbone calculator*

The backbone calculator is responsible for understanding and generating feature vectors of the central modality. Constructing a backbone calculator for a specific domain begins with identify the most common and information-rich data modality in that domain as the processed modality by the backbone calculator. The model architecture is then designed based on this modality. Currently, mainstream model architectures are based on Transformers. A complete Transformer model consists of an encoder and a decoder, where the encoder analyzes the input data to extract compact feature representations, and the decoder uses these feature representations to generate the output content. Since the structures of the encoder and decoder are different, generally, the encoder has stronger comprehension capabilities, while the decoder has more powerful generative capabilities. Transformer-based foundation model backbone calculators have three main architectural forms, including encoder-only architectures, decoder-only architectures, and encoder-decoder architectures. The characteristics of these three architectures are summarized in Table 5.
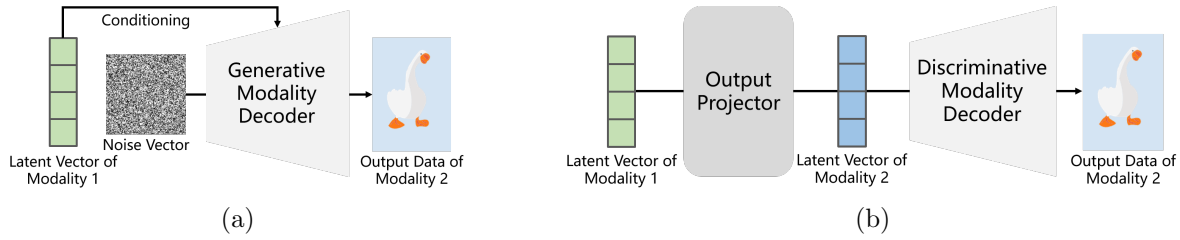
Figure 7   The workflow of output projector and modality decoder. (a) Generative modality decoder;(b) Discriminative modality decoder.

1. **Backbone processor based on encoder-only architecture**: The encoder-only architecture consists solely of the encoder part of the Transformer. It is typically used for tasks that require understanding input text rather than generating new text sequences, such as text classification and sentiment analysis. Due to containing only the encoder part, the encoder model structure is relatively simple but can only produce fixed-length outputs, resulting in limited generation capability. In terms of generation tasks, backbone processors based on encoder-only model can only handle tasks such as sequence completion. BERT [7] is a well-known example of an encoder-only model.

2. **Backbone processor based on decoder-only architecture**: In the decoder-only architecture, the decoder directly processes the input sequence and generates the output sequence without a separate encoder to compactly represent the input sequence. This reduces parameter count and computational overhead but also makes it more challenging for the model to understand input sequences, thus limiting its ability to handle long sequences. This architecture does not require explicit context representation when generating output sequences but captures information automatically within the sequence through self-attention mechanisms. Backbone processors based on decoder-only architecture typically employ autoregressive generation, meaning they generate words or characters one by one based on previous generated content to complete sequence text generation tasks. Models like the GPT series [4–6, 22] and the LLaMA series [10, 11] of foundation language models belong to the decoder-only architecture.

3. **Backbone processor based on encoder-decoder architecture**: The encoder-decoder architecture can simultaneously possess the understanding capability of an encoder and the generation capability of a decoder, but also results in higher parameter count and computational costs. Models like Meta's BART [15] and Google's T5 [16] adopt this architecture.

3.3.4   *Constructing output projectors and modality decoders*

Modality decoders come in two types: generative and discriminative. Generative modality decoders can produce high-quality data samples given conditional information. Discriminative modality decoders, on the other hand, excel in precise reconstruction of data samples based on input vectors. The type of modality decoders also affects the design of output projectors. Thus it requires a joint consideration of these two modules. Figure 7 (a) and (b) respectively illustrate the operational processes of generative and discriminative modality decoders.

1. **Generative modality decoder**: A generative modality decoder utilizes a generative model as its neural network, which can control the generation process using conditional information. It takes other modality features as conditional information and generated data as decoding output. Such a generative model can be termed as a generative modality decoder, without need of explicit construction of output projectors. For instance, DiT [50], a diffusion-based image generation model, can progressively generate images based on previous results and conditional vectors, where cross-attention mechanisms serve the function of the output projector. Similarly, the VAR model [89] also uses self-attention mechanism as the output projector function. By incorporating modality labels in the starting position, VAR knows which content needs to be controlled for generation and generate images in a autoregressive way.

   Generative modality decoders typically employ end-to-end training strategies. During training, the generation part of the model and the modality interaction part are optimized simultaneously. The training objective usually involves minimizing the difference between generated data and real data

while ensuring that generated data meets given conditions. For example, if the model's objective is to generate images based on textual descriptions, a large number of text-image pairs are used during training. And the model is optimized by comparing the similarity between generated images and real images. This similarity can be measured using pixel-level loss functions (such as mean squared error) or more advanced perceptual losses (such as VGG loss). Additionally, adversarial training can be employed to enhance generation quality.

2. **Discriminative modality decoder**: A discriminative modality decoder is responsible only for directly reconstructing feature vectors into their original data form. Thus, it requires an explicit output projector to convert feature vectors from other modalities to the target modality for use. For example, when using the decoder of VQGAN [48] as the discriminative modality decoder in a multi-modality foundation model, it is necessary to first explicitly construct an output projector to transform feature vectors from other modality domains to the image modality, and then use the decoder portion to decode the feature vectors into their original data form. This explicit output projector is typically trained using supervised way, where the model takes feature vectors from other modalities as input and learns to project these features into feature vectors of the target modality by minimizing the error between the two feature vector. Besides, modality decoders are trained together with modality encoders as autoencoders, aiming to improve reconstruction performance by minimizing reconstruction errors.

4 Applications of domain-specific foundation models

Foundation models have emerged as powerful tools with extensive application prospects across various domains. These foundation models possess the capability not only to handle massive data and complex tasks but also to bring about new breakthroughs and innovations.

• **Telecommunications**: Foundation models are expected to be widely applied in sensing, transmission and network performance optimization, etc. For example, LLM fine-tuned for the telecommunications domain can be used to process network log data, model and solve specific network problems [44]. Additionally, foundation models in telecommunications, through the utilization of spatio-temporal correlations and knowledge reasoning [45], are expected to identify and prevent experience issues caused by degraded service quality, and delayed fault response times. This lays the foundation for fine-grained network optimization in real-time. In industrial scenarios, the business understanding capability of foundation models is also expected to help optimize signal transmission and scheduling strategies, improving network efficiency. In research on network foundation models, the NetGPT architecture proposed in the article [90] is expected to become an effective approach to achieve endogenous intelligence in telecommunications networks, while the article [91] discusses the challenges and issues that may be encountered in constructing foundation models for telecommunications. The NetLLM proposed in the article [92] adapts LLM to serve several specific downstream communication tasks. [93] introduces a new memory mechanism to facilitate the embedding of foundation models into the semantic communication process. By leveraging the foundation models' capability to understand and generate multimodal data, it becomes feasible to propose new and more complex transmission approaches that enhance transmission performance.

• **Autonomous driving**: Foundation models play a core role in various key aspects such as vehicle perception, decision-making, and motion control [46]. Specifically, perception tasks in autonomous driving involve real-time monitoring of the vehicle's surroundings, including other vehicles, pedestrians, traffic signs, and road conditions. Foundation models, by analyzing data collected from sensors such as cameras, radar, and LiDAR (Light Detection and Ranging), can identify various objects and construct detailed maps of the vehicle's surroundings. This advanced perception capability is fundamental to achieving safe autonomous driving. At the decision-making level, foundation models need to make rapid and accurate judgments based on perceived information, such as avoiding obstacles, selecting appropriate driving paths, and devising optimal driving strategies in complex traffic situations. Multi-modality foundation models like DriveGPT [94] can not only process visual data but also understand and respond to language-mode instructions, such as planning routes based on voice-input destinations. Additionally, pFedLVM [95] can utilize the powerful performance of pre-trained visual foundation models for image feature extraction, serving as the basis for down-stream tasks.

• **Mathematics**: Since many general-purpose LLMs are typically pre-trained on datasets that include open-source mathematical corpora, they inherently possess some capacity to address mathematical prob-

lems, and through domain-specific enhancements, this capacity can be significantly improved [38, 96–99]. The MAmmoTH proposed in [97] combined chain-of-thought and program-of-thought, fully leveraging the understanding capability of large language models and the computational power of programming languages, achieving good performance in mathematical reasoning. [98] indicates through experiments that the seamless integration of LLMs' programming and reasoning abilities enables them to progressively model and solve complex problems.

• **Medicine**: The applications of foundation models in medicine encompass various aspects, including disease diagnosis, patient treatment, analysis and prediction of genetic and protein structure data, and medical education [40, 41, 100–103]. The HuatuoGPT proposed in the article [102] can simulate the diagnosis and treatment process of doctors, providing preliminary medical consultation and advice for patients. This model not only reduces the workload of doctors but also enables patients to receive timely medical services in remote or resource-limited environments. [103] proposes BiomedGPT, which is capable of performing multimodal medical tasks and has achieved excellent performance in understanding and generating medical-related content.

• **Law**: Foundation models can conduct in-depth analysis of legal documents, identify key information and legal concepts in the text, thereby assisting lawyers and legal advisors in more precise case analysis and legal consultation [37, 104, 105]. For example, foundation models can identify clauses in contracts, extract important legal elements such as obligations, rights and conditions, help lawyers quickly understand document contents, and identify potential legal risks. Moreover, foundation models can be used for case logical reasoning, predicting possible outcomes of cases by analyzing historical cases and relevant legal provisions, providing data support for lawyers to formulate defense strategies. The ChatLaw [106] can provide real-time legal consultation and answers, helping non-professionals understand complex legal issues and even generate drafts of legal documents, reducing the workload of lawyers. Additionally, foundation models can assist in legal research, quickly retrieve relevant legal literature and precedents, and provide solid evidence for legal arguments.

• **Arts**: The application of foundation models is exploring and changing the ways of creative expression and the process of artistic production. By learning from a large number of artworks and creative concepts, foundation models can generate novel artworks, music, literary works, etc., providing creative inspiration and support for artists [42]. For example, generative models can be used to produce artworks [107, 108], and fine-tuning techniques can be applied to adjust the style and content of the generated works [109, 110]. Currently, in the field of video generation, Sora developed by OpenAI allows users to control generated content using text, producing lifelike video works. Furthermore, foundation models also show promising performance in art understanding [111, 112], which provides great potential in the field of art teaching and study.

• **Finance**: Foundation models can cover various tasks such as risk modeling, investment strategy management, market forecasting, etc., providing powerful tools for financial institutions and investors to optimize decision-making [39, 113]. For example, using foundation models to build credit scoring systems to assess borrowers' credit risks. These models analyze factors such as borrowers' historical credit records, financial conditions, and debt levels to predict their ability to repay loans, and based on this, decide whether to approve loan applications and the loan interest rates. Alternatively, foundation models can be used to consider the historical performance, correlations, risks, and expected returns of various asset classes, as well as historical market data, macroeconomic indicators, political events, etc., to make optimal decisions for investors. BloombergGPT, as detailed in [114], was pre-trained on a large-scale financial dataset, resulting in superior performance across many benchmarks. [115] introduces FinGPT, an open-source financial LLM, and positions it not only as a model but also as an open-source framework for Financial LLMs (FinLLMs), which has the potential to fuel innovation among researchers.

5   Future directions

The development of foundation model technology has achieved remarkable progress, but with the continuous advancement of technology, new challenges and issues arise, pointing out the future research directions.

5.1 Challenges in data

Acquiring domain-specific data and modeling data structures are the top priority and most important challenges. Foundation models typically require vast amounts of high-quality data for training [116], which may be costly and time-consuming to obtain in specific domains. Moreover, strengthening privacy regulations impose more restrictions on data collection and usage. To address this challenge, future research can focus on developing new data collection and annotation techniques, as well as utilizing synthetic data and weakly supervised learning methods to reduce reliance on extensively labeled data [117, 118]. This not only reduces costs but also effectively utilizes data resources while ensuring privacy protection. On the other hand, effective modeling of data structures in specific domains requires researchers and practitioners to deeply understand domain-specific business processes, extract key business data, and establish comprehensive data preprocessing pipelines.

Understanding multi-modality data poses another critical challenge. Although existing foundation models excel in processing textual data, their understanding of other modalities such as images and sounds still needs improvement [36], not to mention various new data modalities that may emerge in specific domains. Constructing unified models capable of comprehensively processing various modalities of data is essential for enhancing the model's performance and generalization on multi-modality tasks. This requires researchers and practitioners to not only deeply understand the characteristics of different modalities of data but also explore effective multi-modality fusion and interaction understanding mechanisms.

The availability of aligned multi-modality data presents a potential risk for the long-term development of multi-modal models. While techniques like bridging alignment can reduce the dependency on aligned data [20, 25], the performance and generalization capabilities of multi-modal foundation models still heavily rely on annotated data. This data is costly in terms of both time and financial resources. Therefore, new alignment methods are necessary to address these challenges [53].

5.2 Challenges in model architecture

In the architectural design of foundation models in specific domains, a core challenge is how to build models that can **effectively capture and express deep semantic information** specific to different domain [20, 25, 26]. This requires models to not only have a broad knowledge base but also understand and adapt to domain-specific knowledge and input modalities. Foundation models tailored for specific domains need to achieve high **modularity and customizability** in architecture, enabling adjustment and optimization according to specific application scenarios to adapt to the data characteristics and task requirements of different domains [4, 7, 16]. Furthermore, **interpretability** of models is particularly important in specific domains. When designing architectures, researchers need to consider how to construct models so that their decision-making processes and output results can be understood and trusted by domain experts and end-users. This may involve developing new model mechanisms, introducing interpretable model intermediate representations, or designing visualization tools to demonstrate the internal workings of the model.

5.3 Challenges in computational resources

Computational resources are also a significant challenge for foundation model technology. Training and running foundation models require enormous computational resources, which not only increases economic costs but also may have environmental impacts. Therefore, researching how to improve the **efficiency of model training and inference** [119] and how to **reduce energy consumption** has become an urgent issue. Future research directions may include developing more efficient **model compression and acceleration techniques**, such as knowledge distillation, model pruning, quantization, etc., as well as exploring more efficient training algorithms and specialized hardware designs.

Lightweight deployment is another important direction for foundation model technology. The size and computational requirements of foundation models often make it difficult to deploy them in mobile devices and edge computational scenarios [120]. To enable foundation models to run in resource-constrained scenarios, lightweight model architectures and deployment strategies need to be developed. This may involve simplifying, distilling, and optimizing models to reduce their size and computational requirements while maintaining or improving their performance. Alternatively, employing cloud-edge collaboration strategies can distribute the training and inference processes of foundation models across various server tiers, enabling a cooperative deployment. In cloud-edge learning, it is essential to consider

the joint optimization of sensing, computation, and communication [121–123]. Such joint optimization for the performance of neural networks and resource consumption holds vast potential for innovation in research and can generate significant value in practical application scenarios.

5.4   Challenges in security

Security issues are also challenges that foundation model technology must face. Foundation models may be used to generate false information, infringe on privacy, or be maliciously exploited, and the models themselves may also be subject to adversarial attacks [124]. To ensure the security and reliability of models, relevant work includes but is not limited to the following key points: Firstly, **strengthening the robustness of models** to resist potential adversarial attacks [125]. This may involve developing advanced adversarial training techniques and implementing more stringent data cleaning and preprocessing steps. Secondly, developing and deploying efficient **malicious input detection mechanisms**, using anomaly detection algorithms and real-time monitoring systems to identify and prevent malicious behavior. Furthermore, ensuring **privacy protection** is achieved by diminishing the model's reliance on sensitive data, thereby safeguarding the security and privacy of user information [126]. In addition to technical efforts, **enhancing security in workflow and policy aspects** is also necessary. For example, implementing security audit and certification processes to comprehensively assess the security of models and ensure that model development and deployment comply with ethical and legal standards, continuously updating security policies to address emerging security threats and challenges.

References

1 Ding Y, Fan W, Ning L, et al. A survey on RAG meets LLMs: towards retrieval-augmented large language models. 2024.
2 Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014.
3 He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. 770–778.
4 Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training. 2018.
5 Radford A, Wu J, Child R, et al. Language models are unsupervised multitask learners. 2019.
6 Brown T, Mann B, Ryder N, et al. Language models are few-shot learners. 2020.
7 Devlin J, Chang M.-W, Lee K, et al. BERT: pre-training of deep bidirectional transformers for language understanding. 2018.
8 Du Z, Qian Y, Liu X, et al. GLM: general language model pretraining with autoregressive blank infilling. 2021.
9 Zeng A, Liu X, Du Z, et al. GLM-130B: an open bilingual pre-trained model. 2022.
10 Touvron H, Lavril T, Izacard G, et al. Llama: open and efficient foundation language models. 2023.
11 Touvron H, Martin L, Stone K, et al. Llama 2: open foundation and fine-tuned chat models. 2023.
12 Chen M, Radford A, Child R, et al. Generative pretraining from pixels. In: International Conference on Machine Learning. PMLR. 2020. 1691–1703.
13 Bai Y, Geng X, Mangalam K, et al. Sequential modeling enables scalable learning for large vision models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2024. 22861–22872.
14 Kirillov A, Mintun E, Ravi N, et al. Segment anything. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023. 4015–4026.
15 Lewis M, Liu Y, Goyal N, et al. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. 2019.
16 Raffel C, Shazeer N, Roberts A, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. 2020.
17 Jin M, Wang S, Ma L, et al. Time-LLM: time series forecasting by reprogramming large language models. 2023.
18 Gao S, Koker T, Queen O, et al. UNITS: building a unified time series model. 2024.
19 Liu C, Yang S, Xu Q, et al. Spatial-temporal large language model for traffic prediction. 2024.
20 Tang Z, Yang Z, Zhu C, et al. Any-to-any generation via composable diffusion. 2024.
21 Tang Z, Yang Z, Khademi M, et al. CoDi-2: in-context interleaved and interactive any-to-any generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2024. 27425–27434.
22 Achiam J, Adler S, Agarwal S, et al. GPT-4 technical report. 2023.
23 Liu H, Li C, Wu Q, et al. Visual instruction tuning. 2024.
24 Fei N, Lu Z, Gao Y, et al. Towards artificial general intelligence via a multimodal foundation model. 2022.
25 Girdhar R, El-Nouby A, Liu Z, et al. ImageBind: one embedding space to bind them all. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2023. 15180–15190.
26 Wu S, Fei H, Qu L, et al. NExT-GPT: any-to-any multimodal LLM. 2023.
27 Zhao W. X, Zhou K, Li J, et al. A survey of large language models. 2023.
28 Liu J, Yang C, Lu Z, et al. Towards graph foundation models: a survey and beyond. 2023.
29 Mao H, Chen Z, Tang W, et al. Graph foundation models. 2024.
30 Du Y, Liu Z, Li J, et al. A survey of vision-language pre-trained models. 2022.
31 Yin S, Fu C, Zhao S, et al. A survey on multimodal large language models. 2023.
32 Yeh C.-C. M, Dai X, Chen H, et al. Toward a foundation model for time series data. In: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management. 2023. 4400–4404.
33 Jin M, Wen Q, Liang Y, et al. Large models for time series and spatio-temporal data: a survey and outlook. 2023.
34 Cao Y, Li S, Liu Y, et al. A comprehensive survey of AI-generated content (AIGC): a history of generative AI from GAN to ChatGPT. 2023.
35 Zhou C, Li Q, Li C, et al. A comprehensive survey on pretrained foundation models: a history from BERT to ChatGPT. 2023.

36   Zhang D, Yu Y, Li C, et al. MM-LLMs: recent advances in multimodal large language models. 2024.

37   Lai J, Gan W, Wu J, et al. Large language models in law: a survey. 2023.

38   Ahn J, Verma R, Lou R, et al. Large language models for mathematical reasoning: progresses and challenges. 2024.

39   Li Y, Wang S, Ding H, et al. Large language models in finance: a survey. In: Proceedings of the Fourth ACM International Conference on AI in Finance. 2023. 374–382.

40   Thirunavukarasu A. J, Ting D. S. J, Elangovan K, et al. Large language models in medicine. 2023.

41   Kazerouni A, Aghdam E. K, Heidari M, et al. Diffusion models in medical imaging: a comprehensive survey. 2023.

42   Shahriar S. GAN computers generate arts? A survey on visual arts, music, and literary text generation using generative adversarial network. 2022.

43   Hou X, Zhao Y, Liu Y, et al. Large language models for software engineering: a systematic literature review. 2023.

44   Bariah L, Zhao Q, Zou H, et al. Large language models for telecom: the next big thing?. 2023.

45   Zhou H, Hu C, Yuan Y, et al. Large language model (LLM) for telecommunications: a comprehensive survey on principles, key techniques, and opportunities. 2024.

46   Cui C, Ma Y, Cao X, et al. A survey on multimodal large language models for autonomous driving. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2024. 958–979.

47   Yang J, Jin H, Tang R, et al. Harnessing the power of LLMs in practice: a survey on ChatGPT and beyond. 2024.

48   Esser P, Rombach R, Ommer B. Taming transformers for high-resolution image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021. 12873–12883.

49   Rombach R, Blattmann A, Lorenz D, et al. High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2022. 10684–10695.

50   Peebles W, Xie S. Scalable diffusion models with transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023. 4195–4205.

51   Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. 2017.

52   Su W, Zhu X, Cao Y, et al. VL-BERT: pre-training of generic visual-linguistic representations. 2019.

53   Du C, Wang Y, Song S, et al. Probabilistic contrastive learning for long-tailed visual recognition. 2024.

54   Tay Y, Dehghani M, Abnar S, et al. Scaling laws vs model architectures: how does inductive bias influence scaling?. 2022.

55   Zhai X, Kolesnikov A, Houlsby N, et al. Scaling vision transformers. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR). 2022. 12104–12113.

56   Kaplan J, McCandlish S, Henighan T, et al. Scaling laws for neural language models. 2020.

57   Chung H. W, Hou L, Longpre S, et al. Scaling instruction-finetuned language models. 2024.

58   Iyer S, Lin X. V, Pasunuru R, et al. OPT-IML: scaling language model instruction meta learning through the lens of generalization. 2022.

59   Henighan T, Kaplan J, Katz M, et al. Scaling laws for autoregressive generative modeling. 2020.

60   Wei J, Tay Y, Bommasani R, et al. Emergent abilities of large language models. 2022.

61   Wei J, Wang X, Schuurmans D, et al. Chain-of-thought prompting elicits reasoning in large language models. 2022.

62   Schick T, Schütze H. Exploiting cloze questions for few shot text classification and natural language inference. 2020.

63   Li X. L, Liang P. Prefix-tuning: optimizing continuous prompts for generation. 2021.

64   Liu X, Zheng Y, Du Z, et al. GPT understands, too. 2023.

65   Ding Y, Zhang L. L, Zhang C, et al. LongRoPE: extending LLM context window beyond 2 million tokens. 2024.

66   Dai Z, Yang Z, Yang Y, et al. Transformer-XL: attentive language models beyond a fixed-length context. 2019.

67   Lewis P, Perez E, Piktus A, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. 2020.

68   Edge D, Trinh H, Cheng N, et al. From local to global: a graph RAG approach to query-focused summarization. 2024.

69   Yu Y, Ping W, Liu Z, et al. RankRAG: unifying context ranking with retrieval-augmented generation in LLMs. 2024.

70   Houlsby N, Giurgiu A, Jastrzebski S, et al. Parameter-efficient transfer learning for NLP. In: International Conference on Machine Learning. PMLR. 2019. 2790–2799.

71   Pfeiffer J, Kamath A, Rücklé A, et al. AdapterFusion: non-destructive task composition for transfer learning. 2020.

72   Liu H, Tam D, Muqeeth M, et al. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. 2022.

73   Chen P.-Y. Model reprogramming: resource-efficient cross-domain machine learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2024. vol. 38. 22584–22591.

74   Hu E. J, Shen Y, Wallis P, et al. LoRA: low-rank adaptation of large language models. 2021.

75   Kim J.-H, On K.-W, Lim W, et al. Hadamard product for low-rank bilinear pooling. 2016.

76   Hackbusch W, Khoromskij B. N. Low-rank Kronecker-product approximation to multi-dimensional nonlocal operators. Part I. Separable approximation of multi-variate functions. 2006.

77   Ding N, Qin Y, Yang G, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. 2023.

78   Beltagy I, Peters M. E, Cohan A. Longformer: the long-document transformer. 2020.

79   Zaheer M, Guruganesh G, Dubey K. A, et al. Big bird: transformers for longer sequences. 2020.

80   Yang Z, Dai Z, Yang Y, et al. XLNet: generalized autoregressive pretraining for language understanding. 2019.

81   Gao Y, Xiong Y, Gao X, et al. Retrieval-augmented generation for large language models: a survey. 2023.

82   Chen D, Fisch A, Weston J, et al. Reading wikipedia to answer open-domain questions. 2017.

83   Karpukhin V, Oğuz B, Min S, et al. Dense passage retrieval for open-domain question answering. 2020.

84   Cuconasu F, Trappolini G, Siciliano F, et al. The power of noise: redefining retrieval for RAG systems. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2024. 719–729.

85   Han Z, Gao C, Liu J, et al. Parameter-efficient fine-tuning for large models: a comprehensive survey. 2024.

86   Zhang Q, Chen M, Bukharin A, et al. Adaptive budget allocation for parameter-efficient fine-tuning. In: The Eleventh International Conference on Learning Representations. 2022.

87   Pan R, Liu X, Diao S, et al. LISA: layerwise importance sampling for memory-efficient large language model fine-tuning. 2024.

88   Ma J, Zhao Z, Yi X, et al. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018. 1930–1939.

89   Tian K, Jiang Y, Yuan Z, et al. Visual autoregressive modeling: scalable image generation via next-scale prediction. 2024.

90   Chen Y, Li R, Zhao Z, et al. NetGPT: an ai-native network architecture for provisioning beyond personalized generative services. 2024.

91   Tong W, Peng C, Yang T, et al. Ten issues of NetGPT. 2023.

92   Wu D, Wang X, Qiao Y, et al. Large language model adaptation for networking. 2024.

93   Xie H, Qin Z, Tao X, et al. Towards intelligent communications: large model empowered semantic communications. 2024.

94   Xu Z, Zhang Y, Xie E, et al. DriveGPT4: interpretable end-to-end autonomous driving via large language model. 2023.

95   Kou W.-B, Lin Q, Tang M, et al. pFedLVM: a large vision model (LVM)-driven and latent feature-based personalized federated learning framework in autonomous driving. 2024.

96   Yang Z, Ding M, Lv Q, et al. GPT can solve mathematical problems without a calculator. 2023.

97   Yue X, Qu X, Zhang G, et al. MAmmoTH: building math generalist models through hybrid instruction tuning. 2023.

98   Wang K, Ren H, Zhou A, et al. MathCoder: seamless code integration in LLMs for enhanced mathematical reasoning. 2023.

99   Wu Y, Jia F, Zhang S, et al. MathChat: converse to tackle challenging math problems with LLM agents. In: ICLR 2024 Workshop on Large Language Model (LLM) Agents. 2024.

100   Garg R. K, Urs V. L, Agarwal A. A, et al. Exploring the role of ChatGPT in patient care (diagnosis and treatment) and medical research: a systematic review. 2023.

101   Xi Z, Chen W, Guo X, et al. The rise and potential of large language model based agents: a survey. 2023.

102   Zhang H, Chen J, Jiang F, et al. HuatuoGPT, towards taming language model to be a doctor. 2023.

103   Zhang K, Yu J, Yan Z, et al. BiomedGPT: a unified and generalist biomedical generative pre-trained transformer for vision, language, and multimodal tasks. 2023.

104   Zhou Z, Shi J.-X, Song P.-X, et al. LawGPT: a chinese legal knowledge-enhanced large language model. 2024.

105   Fei Z, Zhang S, Shen X, et al. InternLM-Law: an open source chinese legal large language model. 2024.

106   Cui J, Li Z, Yan Y, et al. Chatlaw: open-source legal large language model with integrated external knowledge bases. 2023.

107   Schumacher D, LaBounty Jr F, Schellekens M. Enhancing Suno's bark text-to-speech model: addressing limitations through Meta's EnCodec and pre-trained Hubert. 2023.

108   Liang X, Zhao Z, Zeng W, et al. PianoBART: symbolic piano music generation and understanding with large-scale pre-training. 2024.

109   Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. 2020.

110   Zhang L, Rao A, Agrawala M. Adding conditional control to text-to-image diffusion models. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2023. 3836–3847.

111   Zhao Z. Adversarial-MidiBERT: symbolic music understanding model based on unbias pre-training and mask fine-tuning. 2024.

112   Garcia N, Vogiatzis G. How to read paintings: semantic art understanding with multi-modal retrieval. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops. 2018. 0–0.

113   Lopez-Lira A, Tang Y. Can ChatGPT forecast stock price movements? Return predictability and large language models. 2023.

114   Wu S, Irsoy O, Lu S, et al. BloombergGPT: a large language model for finance. 2023.

115   Yang H, Liu X.-Y, Wang C. D. FinGPT: open-source financial large language models. 2023.

116   Du F, Ma X.-J, Yang J.-R, et al. A survey of LLM datasets: from autoregressive model to AI chatbot. 2024.

117   Zhou Z.-H. A brief introduction to weakly supervised learning. 2018.

118   Van Engelen J. E, Hoos H. H. A survey on semi-supervised learning. 2020.

119   Wang Z, Zhou Y, Shi Y, et al. Federated fine-tuning for pre-trained foundation models over wireless networks. 2024.

120   Yin W, Xu M, Li Y, et al. LLM as a system service on mobile devices. 2024.

121   Xu W, Yang Z, Ng D. W. K, et al. Edge learning for B5G networks with distributed signal processing: semantic communication, edge computing, and wireless sensing. 2023.

122   Zhu G, Lyu Z, Jiao X, et al. Pushing AI to wireless network edge: an overview on integrated sensing, communication, and computation towards 6G. 2023.

123   Wen D, Liu P, Zhu G, et al. Task-oriented sensing, computation, and communication integration for multi-device edge AI. 2023.

124   Yao Y, Duan J, Xu K, et al. A survey on large language model (LLM) security and privacy: the good, the bad, and the ugly. 2024.

125   Zou A, Wang Z, Kolter J. Z, et al. Universal and transferable adversarial attacks on aligned language models. 2023.

126   Feng Q, Kasa S. R, Yun H, et al. Exposing privacy gaps: membership inference attack on preference data for LLM alignment. 2024.