

Chameleon 是一种典型的生成式学习方法<sup>[280]</sup>，在数据预处理阶段便将不同模态的信息整合为统一的 Token 序列，实现多模态数据的深度融合。通过将图像和文本转换为统一的 Token 表示，模型能够在处理数据时同时考虑所有模态的信息，从而提升对多模态数据的理解和生成能力。在输入处理上，Chameleon 使用两个独立的分词器：文本分词器将文本拆分为单词或子词单元，而图像分词器则将图像编码为离散 Token 序列，类似于文本中的单词。随后，这些 Token 被组合为统一的输入序列。Chameleon 基于统一的 Transformer 架构，无需为图像或文本分别设计独立的编码器或解码器。Transformer 强大的特征提取和序列建模能力，使其能够捕捉图像与文本之间的复杂关联。此外，Chameleon 在注意力机制中引入了归一化步骤，对 Query 和 Key 向量进行归一化处理，以控制输入到 Softmax 层的值范围，平衡不同模态在特征表示上的尺度。这种方法能够稳定模型训练，避免模态间竞争导致的不稳定性。Chameleon 在大量多样化的数据上进行预训练，包括文本、图像 - 文本对以及交错序列等。这种多样化的训练数据使模型能够学习丰富的多模态表示，显著提升了其泛化能力和对复杂多模态任务的适应性。

#### 4. 映射学习

VLM 的训练通常面临显著的计算开销问题，依赖庞大的计算资源和海量数据支持。为解决这一问题，映射学习范式提出了一种高效的训练方法，即在现有的大语言模型和视觉特征提取模型的基础上进行二次训练，如图 7.5 所示。该方法通过利用开源的大语言模型，重点学习文本模态与图像模态之间的映射关系。通过构建这种映射，大语言模型能够适应视觉任务，同时显著降低对计算资源的需求。



图 7.5 视觉语言模型映射学习范式<sup>[271]</sup>

Frozen<sup>[284]</sup> 是一种将预训练大语言模型与视觉信息相融合的开创性方法。该方法设计了一种简洁高效的特征转换架构，用于将图像特征映射到文本语义空间。具体来说，它采用 NF-ResNet-50 作为图像特征提取的基础模型，并训练了一个特征到语义的转换函数。语言处理部分使用了一个拥有 70 亿参数的 Transformer 模型，该模型通过 C4 数据集完成预训练。在训练阶段，Frozen 使用 Conceptual Captions 数据集对系统进行优化，专注于文本生成任务，从而实现多模态信息的高效融合。在预测过程中，系统能够同时处理图像和文本输入，展现出强大的多模态理解与生成能力。

目前，包括 MiniGPT-4<sup>[252]</sup>、LLaVA<sup>[262]</sup>、Qwen-VL<sup>[78]</sup> 等在内的绝大多数视觉语言模型都采用映射学习方法。

## 7.2.2 语音语言模型架构

语音语言模型 (Speech-Language Models, SLM) 是一种结合语音处理与自然语言理解的多模态大模型，旨在实现语音与文本模态的深度融合。与传统的语音识别后级联文本处理方法不同，SLM 通过端到端架构直接学习音频特征与语言语义的映射关系，从而增强了模型在开放世界场景中的泛化能力。语音语言模型在多模态环境中应用广泛，如语音识别、语音合成、语音翻译、语音交互等。

### 1. SLM 输入/输出模式

语音语言模型的输入/输出模式可以根据任务需求分为三种主要类型，如图7.6所示：语音到文本 (Speech-to-Text, S2T)、语音文本到文本 (Speech&Text-to-Text, ST2T) 和语音文本到语音文本 (Speech&Text-to-Speech&Text, ST2ST)。

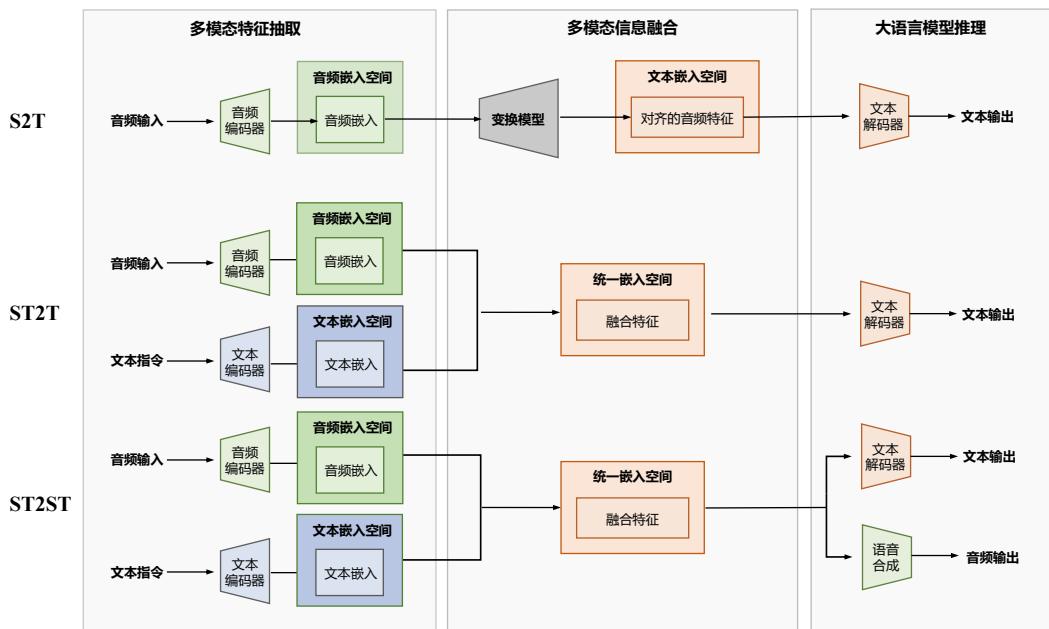


图 7.6 语音语言模型输入输出模式<sup>[285]</sup>

S2T 是最基础的模式，模型以语音作为输入，并生成对应的文本输出。这种模式通常用于自动语音识别 (Automatic Speech Recognition, ASR) 任务。模型架构中包含一个音频编码器，用于提取语音信号中的特征，而由于输入中没有文本模态，因此不需要文本编码器。这种模式通常采用解码器架构，通过一个特征转换模块将音频特征映射到文本嵌入空间，以生成精准的文本输出。

S2T 模式实现简单，适用于纯语音到文本的转换任务，但是无法处理更复杂的多模态任务。

ST2T 是目前语音语言模型中最广泛采用的模式。该模式支持同时输入语音和文本，其中文本通常作为指令或任务提示。模型通过同时处理音频与文本模态的信息，融合两者的特征后生成最终的文本输出。这种模式不仅能够支持多任务学习，还能充分发挥大语言模型的强大能力，处理更广泛的任务，可以应用于语音翻译、语音情感分析等涉及音频和文本模态的任务。

ST2ST 是一种更高级的模式，模型在输入中结合语音和文本，并在输出中同时生成语音和文本。这种模式在解码阶段需要额外的语音合成模块（Vocoder）来生成语音输出。ST2ST 模式不仅能够完成基本的语音识别任务，还支持文本语音生成（Text-to-Speech, TTS）、语音翻译及语音转换等复杂任务。

## 2. 语音嵌入表示预训练

语音嵌入表示预训练是一种通过在大规模语音数据上学习语音通用特征表示，进而提升下游语音任务性能的关键技术。近年来，基于不同模型架构的预训练方法逐渐成为研究热点，其中包括基于卷积神经网络的模型、基于 Transformer 架构的模型以及基于 Codec 的模型。

卷积神经网络（Convolutional Neural Network, CNN）凭借其强大的特征提取能力、参数共享机制和稀疏连接特性，在音频处理领域具有显著优势。在语音识别系统中，基于 CNN 的模型通常通过短时傅里叶变换（Short-time Fourier Transform, STFT）将原始声波信号转换为对数梅尔频谱图（Log Mel Spectrogram）进行处理，以便更高效地提取关键特征。经典模型如 AlexNet 和 VGG 在音频分类任务中表现优异。PANNs（Pretrained Audio Neural Networks）<sup>[286]</sup> 基于 CNN14 架构，在 Audioset 标签任务中也取得了出色的结果。基于 CNN 的模型在提取局部特征方面具有显著优势，尤其擅长分析频谱图中的短期时间信息。然而，这类模型在捕捉音频信号的长时依赖关系方面存在一定的局限性。

自注意力机制的引入使得 Transformer 架构在捕捉音频序列的长程依赖方面展现出显著优势。Wav2vec 2.0<sup>[287]</sup> 结合了 CNN 和 Transformer 的优点，其首先通过卷积网络提取局部特征，然后利用自注意力机制分析时间维度的全局关联。该模型采用了一种无监督学习方法，通过将原始声学信号映射到潜在空间，并结合遮蔽和对比学习机制生成上下文表示。Wav2vec 2.0 对比学习机制通过预测被遮蔽区域的表示形式，有效增强了模型的迁移能力，使其在多种下游任务中均表现优异。Whisper<sup>[288]</sup> 则通过引入多任务训练框架进一步提升模型性能。其架构整合了 Transformer 的编码器-解码器结构与卷积单元，核心创新点在于实现了跨任务的 Next Token 预测机制，从而在多种应用场景中展现出卓越的适应性。此外，这一设计有效缓解了监督学习模型在微调阶段的过拟合问题，使模型更加稳健和通用。

AST（Audio Spectrogram Transformer）<sup>[289]</sup> 是一种完全基于注意力机制的模型，摒弃了传统的卷积架构。尽管 AST 在灵活性方面具有显著优势，但其对大规模数据的依赖较高，同时在训练过程中需要较大的 GPU 内存，并面临较长的训练时间问题。为了解决这些限制，HTSAT（Hierarchical Transformer-based Spectrogram Audio Transformer）<sup>[290]</sup> 引入了分层模型结构，通过每层 Transformer

分别捕捉时间维度和结构信息，从而更高效地处理长时间音频信号。另一方面，AudioMAE 将遮掩自编码器（Masked Autoencoder, MAE）<sup>[291]</sup> 的设计扩展至音频领域，采用基于 Transformer 的编码器-解码器架构。在预训练阶段，AudioMAE 对输入的对数梅尔谱图进行高比例遮掩，由编码器处理未遮掩的 Token，解码器则负责重建被遮掩的部分，从而实现音频表示的高效学习。

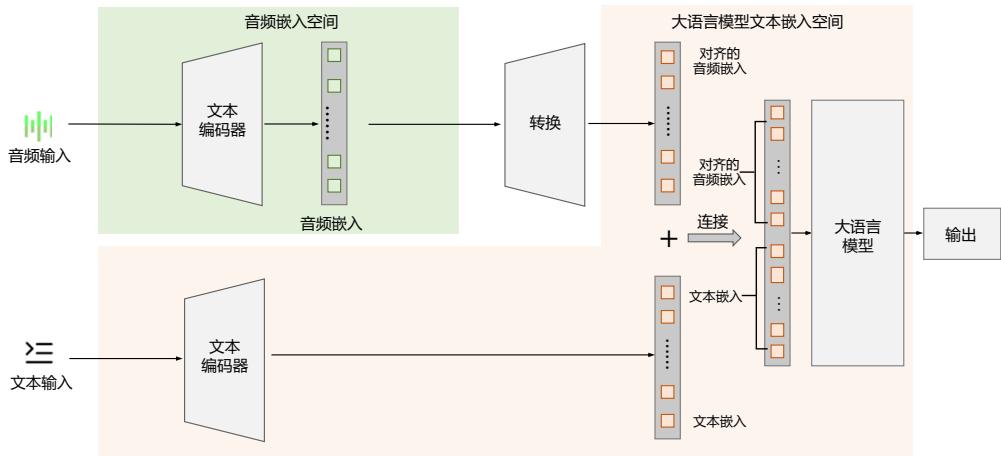
基于 Codec 的模型依托于编码器-解码器结构，能够将连续的音频信号转换为离散的 Token，为语音语言模型的开发提供了重要基础。尽管这种离散化过程难免导致一定程度的数据损失，但该类模型在声学特征提取和高质量音频重建方面表现出色。SoundStream<sup>[292]</sup> 首次提出了一种基于流式的 SEANets<sup>[293]</sup> 架构，通过残差向量量化（Residual Vector Quantization, RVQ）机制实现多流并行处理，并结合重建损失和对抗损失共同优化模型训练，从而显著提高了重建音频的质量。Encodec<sup>[294]</sup> 在此基础上引入了 LSTM 模块以增强序列分析能力，同时结合 Transformer 架构优化离散符号序列的建模能力，从而在多种语音任务中取得了显著的性能提升。

### 3. 语音和文本表示融合架构

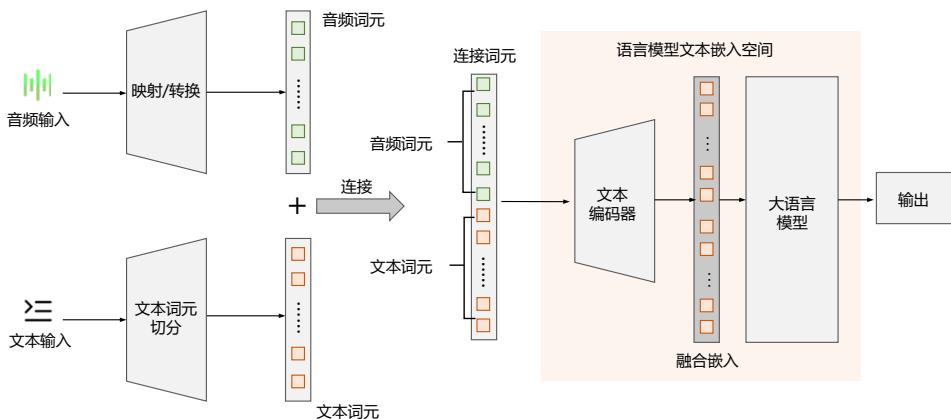
获得语音模态信息后，需要将其与文本模态信息集成，以便大语言模型进行最终推理。语音和文本表示融合主要有两个技术路线：语音模态表示转换到文本模态空间；语音和文本两个模态数据融合在同一空间联合表示。

语音到文本模态的转换是目前广泛采用的方法之一。这种方法充分考虑到大语言模型主要是为文本模态设计的特点，通过将语音模态信息投射到文本空间，实现语音与文本模态的直接对齐，从而在最大程度上保留大语言模型的能力。为了实现这一目标，通常需要引入一个“连接器”（Connector）或“投射器”（Projector）来将语音模态特征转换到文本模态特征空间。在此过程中，需尽量减少语音特征信息的损失，并保证模态转换的平滑性。目前，主要有以下两种实现方式：直接投射（Direct Projection）和 Token 映射（Token Mapping）。

直接投射方法通过连接器将语音特征映射到大语言模型的文本模态嵌入空间<sup>[295, 296]</sup>，如图7.7所示。语音特征经过编码器提取，生成包含语音信息的特征张量。该张量随后通过投射器转化为与文本模态对齐的嵌入向量。生成的语音嵌入向量与输入文本的嵌入向量拼接，形成一个融合语音和文本信息的新嵌入向量，并将其输入到大语言模型中进行处理。此外，一些研究者采用隐式投射方式，通过调整原始编码器的参数，在训练过程中直接完成语音到文本模态的映射，无需额外的连接器。

图 7.7 语音和文本表示融合架构直接投射方法<sup>[285]</sup>

Token 映射方法通过将语音特征转换为大语言模型可处理的文本 Token 实现模态转换<sup>[297]</sup>，如图7.8所示。具体而言，语音特征经过投射器或转换器生成与文本 Token 对应的表示，这些符号随后与文本的 Token 序列结合，形成一个同时包含语音和文本信息的 Token 序列，并将其输入到 LLM 中进行统一处理。该方法不仅能够较好地保留语音特征信息，还确保了 LLM 在处理数据时的连续性和一致性。

图 7.8 语音和文本表示融合架构 Token 映射方法<sup>[285]</sup>

尽管将语音模态投射到文本模态空间的方法简便高效，但在模态转换过程中难以避免信息损失和模态冲突的问题。为了解决这些不足，研究者提出了一种通过修改大语言模型的输入空间，在

Token 空间中直接融入语音模态信息，实现语音和文本的深度融合。该方法通过增加 Token 空间，在原有文本 Token 的基础上新增语音 Token，形成扩展的 Token 空间<sup>[298-300]</sup>，如图7.9所示。具体而言，首先从语音特征中提取信息并生成语音 Token；然后将这些语音 Token 与文本 Token 结合，形成一个新的输入 Token 序列；最后将该序列作为 LLM 的输入，直接进行语音和文本模态的联合建模。这种方法通过在大语言模型的 Token 空间中引入语音信息，最大程度地保留了语音的原始特征，同时有效避免了模态转换过程中可能出现的信息损失问题。

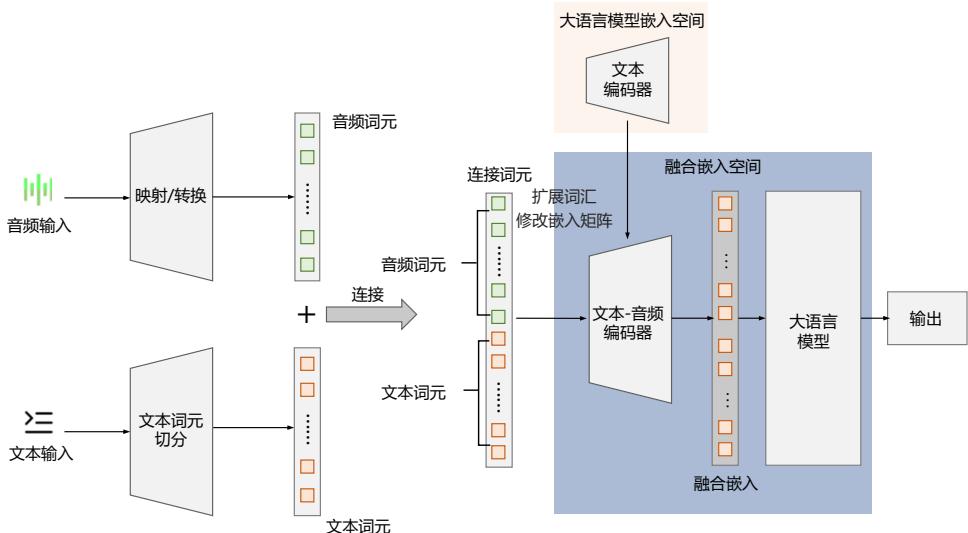


图 7.9 语音和文本表示融合架构语音文本 Token 空间融合方法<sup>[285]</sup>

### 7.2.3 多模态大语言模型架构

多模态大语言模型的架构种类繁多，其设计方式根据任务需求和输入输出的模态复杂性而有所不同。本节将重点介绍两种具有代表性的多模态模型：一是能够处理任意模态输入与输出的多模态大语言模型 AnyGPT<sup>[300]</sup>，二是具有多视觉编码器融合架构的眸思（MouSi）<sup>[301]</sup>。AnyGPT 通过统一的框架实现了跨模态的无缝交互，具备高度灵活的适应性，而眸思则通过集成多个视觉编码器，大幅增强了对复杂视觉信息的理解与生成能力。两者在多模态领域均展现出强大的性能和应用潜力。

#### 1. AnyGPT

AnyGPT 将所有模态的数据转换为统一的离散化表示，并基于大语言模型采用的 Next Token Prediction 任务进行统一训练。基于 GPT 的原始架构以及多模态的离散化表示，AnyGPT 统一了文本、语音、图像和音乐四种模态，并实现了任意模态组合的相互转换，为多模态交互提供了一个

统一的框架，如图 7.10 所示。

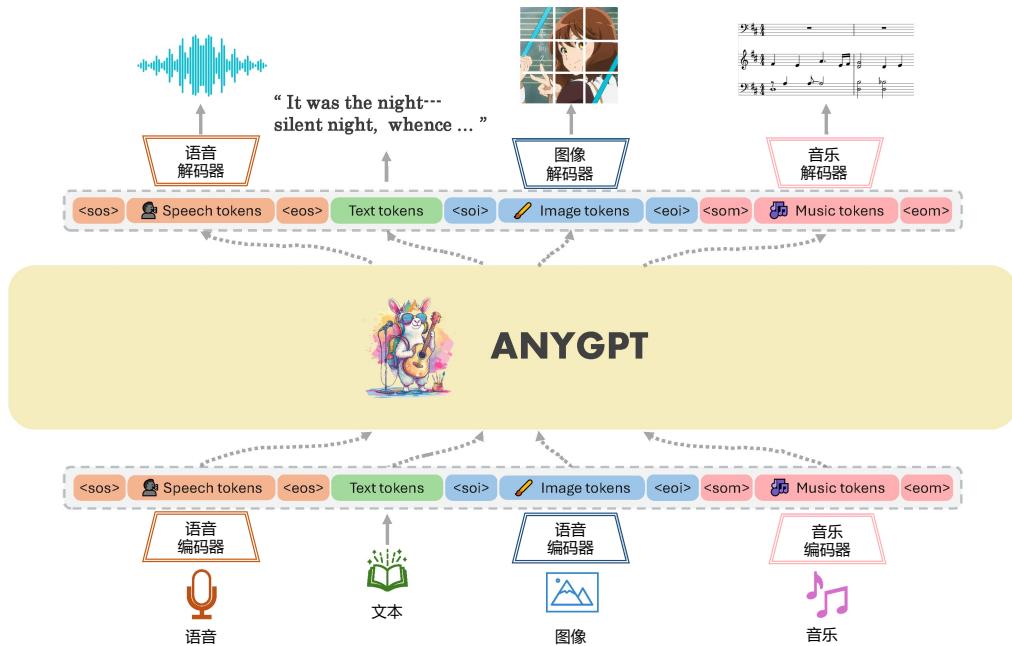


图 7.10 AnyGPT 模型框架<sup>[300]</sup>

AnyGPT 所提出的统一的多模态生成框架由三个核心组成部分构成：多模态分词器、多模态大语言模型以及多模态生成器。具体来说，多模态分词器的作用是将连续的非文本模态数据转换为离散的 Token，并将这些 Token 组织成多模态交错序列。随后，大语言模型以 Next Token 预测损失为目标，在这些多模态序列上进行统一训练。在推理阶段，生成的多模态 Token 会通过对应的生成器解码回原始的模态表示。为了进一步提升生成结果的质量，还可以借助多模态增强模块对输出进行后处理，例如声音克隆或图像超分辨率等技术。

AnyGPT 使用 SEED<sup>[302]</sup> 作为图像分词器。SEED 由 ViT 编码器、因果 Q-Former、VQ (Vector Quantization) 码本、多层次感知机以及 UNet 解码器组成，其内部码本 (Codebook) 包含 8192 个码元 (Entry)。在具体实现上，SEED 将尺寸为  $224 \times 224$  的 RGB 图像分解为  $16 \times 16$  的小块 (Patch)，经过编码后将这些小块转换为量化的码元序列。这些码元与预训练的 unCLIP Stable Diffusion 模型的编码空间对齐，最终通过 UNet 解码器将码元序列恢复为原始图像。

SpeechTokenizer<sup>[303]</sup> 则应用于 AnyGPT 作为语音分词器。SpeechTokenizer 的内部结构包含 8 个码本，每个码本包含 1024 个词元表示。其架构基于编码器-解码器，并结合残差向量量化 (RVQ)，能够将单通道音频序列压缩为离散的矩阵表示。下采样后的帧率为 50Hz，语音分词器通过结合语义损失和重建损失，将语音信息解耦为语义信息和副语言学信息。具体来说，10 秒的音频会被转

换为一个大小为  $500 \times 8$  的矩阵，其中包含  $500 \times 1$  的语义 Token 和  $500 \times 8$  的声学 Token。

AnyGPT 使用 `Encodec`<sup>[294]</sup> 作为音乐分词器。`Encodec` 内部包含 4 个码本，每个码本包含 2048 个词元表示。具体实现中，使用一个在音乐数据上预训练的模型，输入为 32kHz 的单声道音频。编码器将输入音频转换为嵌入向量，随后通过残差向量量化 (RVQ) 进行量化，使用 4 个量化器，每个量化器包含 2048 个码元，从而生成一个总数为 8192 的音乐 Token 表示。对于 5 秒长度的音频，`Encodec` 会将其量化为一个大小为  $250 \times 4$  的码元矩阵。为了适配语言模型的输入格式，将这些码元按逐帧方式展平成一维序列，便于语言模型预测完整的音乐信息。

为了将多模态的离散表示纳入预训练的大语言模型，AnyGPT 对模型进行了扩展，具体包括将每种模态的 Token 加入到词汇表中，并相应地扩展嵌入层和预测层。新加入的参数均采用随机初始化。最终，所有模态的 Token 组合形成了一个新的词汇表，其大小等于所有模态的 token 数之和。借助特定模态的分词器，能够将多模态数据压缩为离散的 Token 序列。语言模型在这些序列上执行 Next Token Prediction 任务进行训练，从而使核心的 LLM 能以自回归的方式自然地统一多模态感知、理解、推理和生成等任务。AnyGPT 使用 LLaMA-2 7B 的参数对大语言模型进行初始化，除了扩展嵌入矩阵和预测头外，语言模型的其余部分保持不变。

使用大语言模型生成高质量的多模态数据是一项具有挑战性的任务，因为图像和音频的精确表示需要大量存储，导致序列长度显著增加，从而提高了语言模型的计算复杂度。为了解决这一问题，AnyGPT 提出了一种两阶段框架，用于高质量多模态数据生成，包括语义信息建模和感知信息建模。在语义层面，自回归语言模型生成融合且对齐的多模态 Token 序列；随后，非自回归模型将这些多模态语义 Token 转换为高保真的多模态内容，从而在性能和效率之间取得平衡。

具体来说，在视觉语言建模中使用 SEED 标记，并通过扩散模型将其解码为高质量图像。在语音生成任务中，采用 SoundStorm 模型生成声学 Token，随后将其解码为原始音频数据。对于音乐生成，使用 `Encodec` 标记以捕提高频细节，并通过 `Encodec` 解码器将其重构为高保真的音频数据。通过这种设计，AnyGPT 在显著减少语音序列长度的同时，能够生成高质量的多模态数据，从而在生成效果和计算效率之间实现了良好的平衡。

## 2. 眇思 (MouSi)

当前的视觉语言模型经常遭遇单视觉编码器组件能力不足和视觉 Token 过长等挑战。这些挑战会限制模型准确理解繁复的视觉信息和过长的上下文信息。解决这些难题对于提高 VLM 的性能和可用性至关重要。

为解决上述问题，多模态大模型眸思 (MouSi)<sup>[301]</sup> 提出了使用多专家技术以协同各视觉编码器的能力，这些能力包括图像文本匹配，光学字符识别，图像分割等。该技术引入一个融合网络使得来自不同视觉专家的输出得到统一，同时弥合了视觉编码器和预训练 LLM 之间的差异。此外，还提出了二维可训练图像位置编码方法，减轻了由于图像特征序列过长而造成的位置编码浪费，有效解决了位置溢出和长度限制的问题。多视觉专家融合多模态大模型 MouSi 框架如图7.11所示。

图 7.11 眇思 (MouSi) 模型框架<sup>[301]</sup>

基于 MouSi 模型，当用户上传一张描绘风媒花授粉过程的图片并询问“哪些球果产生花粉？”时，该图片依次经过 CLIP 专家、SAM 专家、Layout Mv3 专家及其他专家的编码处理，产生多组不同的视觉标记。随后，一个多视觉融合网络压缩融合多通道视觉信息，并将其与视觉输入标记对齐。用户的问题通过大语言模型的嵌入层被处理成文本标记。最终，MouSi 通过对视觉语言标记进行处理，完成 VQA（视觉问答）和 OCR（光学字符识别）任务，从图片中识别答案文本，生成正确答案“雄性球果产生花粉。”

由于不同视觉专家的输出序列在维度和数量上往往存在差异，因此需要设计融合网络来统一处理这些输出。为了更好地整合多专家信息，MouSi 模型对两种方法进行了改进，提出了 MLP 投影融合网络和 Q-Former 融合网络。然而，在实际应用中，多个视觉专家输出的大量视觉标记不仅增加了视觉语言模型的计算成本和内存使用率，还可能超过推理过程中最大序列长度的限制。为了解决这一问题，MouSi 模型提出了多补丁-单标记投影方法，以按比例减少每个专家的输出标记数量。具体而言，由于图像信号具有局部性和稀疏性属性，用一个标记表示相邻的多个补丁是合理的。这种方法通过对局部视觉信息进行压缩，将多个补丁映射为单个标记，从而实现了多通道视觉信号的高效传输。通过多补丁-单标记投影，不仅有效降低了视觉信号传输的冗余，还减少了视觉大语言模型后续处理的计算成本，显著提高了推理效率，为多视觉专家的高效整合提供了切实可行的解决方案。

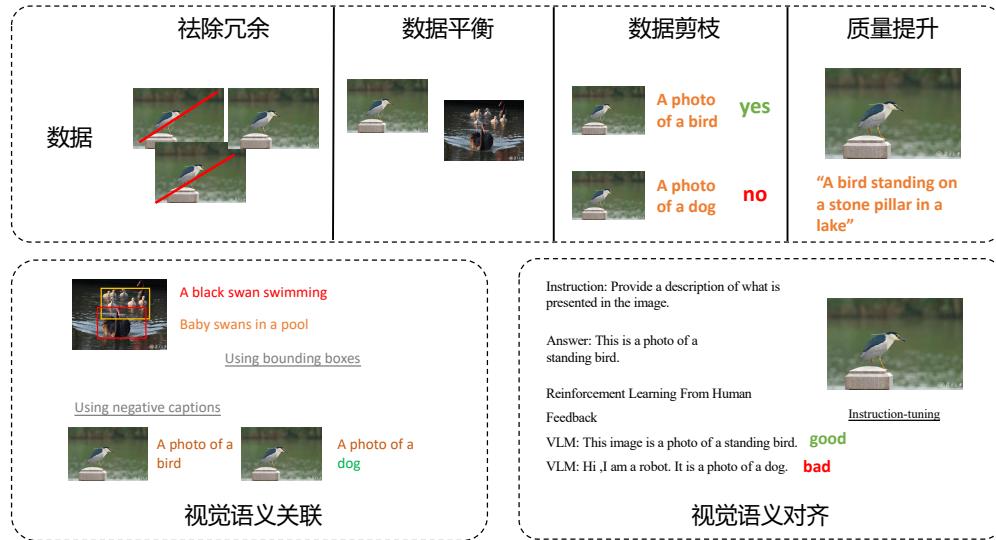
尽管通过多补丁-单标记操作或在 Q-Former 中定义少量查询可以显著减少视觉标记的数量，但

在推理过程中，视觉标记对位置编码的占用仍然是一个不可忽视的问题。事实上，视觉标记的长度通常比文本标记高出 500 倍以上，在具有位置感知的视觉语言模型中，这会消耗大量的位置嵌入资源。鉴于视觉专家本身已经包含位置编码信息，为每个视觉标记再次分配视觉大语言模型的位置嵌入显得冗余且低效。为了解决这一问题，MouSi 模型提出了一种二维可训练图像位置编码方法，通过直接在视觉标记中引入可训练的二维位置编码，避免了对视觉大语言模型位置嵌入的额外占用。这种方法不仅有效解决了多视觉专家导致的超长序列问题，还减少了位置编码的冗余分配，从而优化了视觉标记的处理效率，为多模态模型的可扩展性提供了重要支持。

### 7.3 多模态大语言模型训练策略

深度神经网络缩放法则（Scaling Law）为多模态大语言模型的训练策略提供了重要参考。以往业界普遍采用增加计算资源和模型规模的方式来提升性能，然而，根据文献 [304] 的研究成果，优化数据处理环节亦可带来突破性进展。以 CLIP 为例，其采用 4 亿张图像进行训练，开源版本 OpenCLIP<sup>[305]</sup> 则需数百卡 GPU 集群运行数天至数周。文献 [304] 提出通过构建高效的数据处理管道，可实现性能提升，同时避免成本大幅上升。

如图 7.12 所示，数据是训练多模态大语言模型的核心要素之一。构建一个多样且平衡的数据集对于模型学习覆盖足够多概念的良好世界模型至关重要。清除大型数据集中常见的重复数据同样重要，这不仅能够节省大量计算资源，还能降低模型过度记忆的风险。与此同时，数据剪枝也是数据处理的重要环节，确保文本描述与图像内容高度相关，有助于模型更好地理解和对齐多模态信息。可以通过改进模型对视觉语义关联（Grounding）能力来增强对图文关系的理解，并通过引入人类偏好优化对齐效果。在 OCR 任务中，使用专门的增强技术可以进一步提升文本读取和翻译能力。通过结合高效数据处理、合理的模型架构选择和针对性优化策略，可以显著提升多模态大语言模型的训练效果和应用能力。

图 7.12 多模态大语言模型训练策略<sup>[271]</sup>

本节主要从数据处理、视觉语义关联、文本对齐等方面进行介绍。

### 7.3.1 数据处理

在多模态大语言模型的训练中，数据质量对模型性能起着至关重要的作用。高效的数据处理与筛选策略能够显著提升模型的学习效果及其在下游任务中的泛化能力。为评估基础数据集的质量，研究团队提出了 DataComp 框架<sup>[305]</sup>。该框架基于标准化的 CLIP 架构与预训练参数，旨在构建能够在 38 项下游任务中表现卓越的图像-文本数据组合。DataComp 构建了一个包含 128 万至 128 亿对图像-文本样本的噪声网络数据库，并系统性地探索了多种数据筛选策略。研究表明，剪枝优化是提升跨模态大语言模型效果的关键技术手段，为高能模型的训练提供了重要支持。

数据剪枝的方法可以分为三类：(1) 使用启发式方法去除低质量样本；(2) 基于预训练 VLM 的打分方法对图文对进行排序，丢弃对齐较差的样本；(3) 创建多样化且平衡的数据集。

启发式方法可以分为单模态和多模态过滤两种类型。在单模态过滤中，常见策略包括去除文本复杂度较低的描述（如文本中涉及的对象、属性和动作数量较少）<sup>[306]</sup>，使用 fastText<sup>[307]</sup> 去除非英文的图片文本描述，以及基于图像分辨率和宽高比过滤低质量图像。相比之下，多模态过滤策略更加复杂，通常通过图像分类器检测图像中的对象，并过滤掉那些图像与文本描述中对象无法匹配的样本<sup>[308]</sup>。此外，由于网络数据集中图像往往包含部分文本信息，多模态过滤还可以采用文本检测工具（如 text-spotters<sup>[309]</sup>）来去除图像与文本描述高度重叠的样本。这种方法有助于模型

更专注于学习高级视觉语义，而非过度依赖 OCR 任务，从而提升模型在对象和场景相关零样本任务中的性能。

基于预训练 VLM 的剪枝方法是目前提升数据质量与模型训练效率的最有效策略之一。这些方法通过计算图文对的嵌入相似性来评估对齐程度。其中，CLIPScore<sup>[310]</sup> 依托预训练的 CLIP 模型，计算图像和文本嵌入之间的余弦相似度，并据此对图文对进行排序；LAION 的筛选策略基于由 4 亿对图文对训练的 OpenAI CLIP 模型，对大规模网络数据集进行对齐评估，并过滤得分最低的样本；T-MARS 方法<sup>[311]</sup> 在计算 CLIPScore 前，通过检测并遮盖图像中的文本区域，从而提升对齐分数的准确性；而 Sieve<sup>[312]</sup> 则利用在小而精的数据集上预训练的生成式图像描述模型，有效减少了 CLIPScore 排序中的误判（如高分或低分错误）。通过这些优化策略，图文对的筛选变得更加精准，显著提高了数据质量和模型性能。

多样化且平衡的数据集是提升多模态大语言模型泛化能力的核心因素<sup>[313]</sup>。为构建这样的数据集，DataComp 提出了从多样化设计的数据集中进行采样的策略。具体而言，采样方法主要分为基于文本和基于图像两种：基于文本的采样方法保留与 ImageNet 类别相关联的图文对描述；而基于图像的采样则通过利用 OpenAI CLIP 的 ViT-L/14 模型对图像进行编码，并借助 FAISS 工具将大规模噪声图像聚类为 100,000 个组，然后根据 ImageNet 训练样本的嵌入，选择与这些样本最相近的聚类，从而生成具有多样性的图像数据集。尽管这些方法能有效提升数据的多样性，但它们对 ImageNet 等语义数据集的依赖可能会引入类别偏倚，从而限制模型在新下游任务中的泛化能力。此外，MetaCLIP<sup>[314]</sup> 提出了另一种方法，利用来自 Wikipedia 和 WordNet 的 500,000 个查询作为元数据，构建覆盖广泛概念的预训练数据分布。通过“平衡采样”算法，MetaCLIP 限制每个查询的样本数量（最多 20,000 个），在概念的多样性和代表性之间寻求平衡，从而进一步提升模型的泛化能力。

### 7.3.2 视觉语义关联

视觉语义关联是多模态大语言模型和生成模型研究中的一项核心挑战。其主要目标是解决模型对文本提示理解不充分的问题，这种不足可能导致模型忽略提示中的某些关键信息，或错误生成不存在的内容。模型在处理视觉与文本的关联时，需要克服诸多复杂性，例如物体的空间位置关系（如左右位置）、否定表达、计数能力，以及属性理解（如颜色和纹理）。虽然目前尚无单一的方法能够完全解决这些问题，但研究者提出了一些行之有效的策略来提升模型的视觉语义关联能力。本节将重点介绍两种常用的改进方法：基于边界框标注和负样本生成方法。

#### 1. 基于边界框标注

基于边界框标注是一种直接且高效的方式，用于增强视觉语义关联能力。例如，X-VLM<sup>[315]</sup> 模型通过结合边界框回归与交并比 (IoU) 损失，成功实现了视觉概念的精确定位，并将这些概念与对应的文本描述对齐。通过明确标注图像中物体的位置及其相关描述，该模型能够更精准地将文本提示与正确的视觉线索关联，从而显著提升语义理解能力。这种方法的核心在于细粒度的视觉

标注，它有效地帮助模型理解复杂的视觉与文本关系。X-VLM 的训练依赖于多个大规模标注数据集，包括 COCO<sup>[316]</sup>、Visual Genome<sup>[317]</sup>、SBU 和 Conceptual Captions<sup>[318]</sup>，总计包含约 1600 万张图像。这些数据集丰富的标注信息为模型提供了大量高质量的视觉语义关联训练样本，使其在图文检索、视觉推理、视觉语义对齐以及图像描述等任务中均表现优异，超越了其他现有方法。这表明，边界框标注不仅能够提升模型的性能，还为复杂任务提供了更强的泛化能力。

除了直接利用现成的标注数据集，一些研究者选择通过公开模型生成新的图文对数据集。例如，Kosmos-2<sup>[259]</sup> 使用网络爬取的数据构建了大规模图文对。其方法首先借助 spaCy 从文本中提取名词，然后通过基础模型 GLIP<sup>[319]</sup> 检测与这些名词相关的边界框。随后，使用 spaCy 从文本中进一步提取与名词对应的描述，生成能够与检测到的边界框匹配的图文对。这种方法显著扩展了标注数据的规模，为提升模型在视觉语义关联任务中的表现提供了支持。然而，这种生成式方法的效果在很大程度上依赖于基础模型的性能。如果基础模型（如 GLIP）在某些稀有名词或复杂实例的检测上表现不佳，生成的边界框及其对应的描述可能存在误差。这种误差可能会导致后续的下游任务表现受限。因此，如何进一步提升基础模型的准确性，或设计更鲁棒的生成方法，是未来研究的重要方向。

## 2. 负样本生成方法

负样本生成在对比学习目标中扮演着关键角色，被广泛应用于缓解模型训练中的崩塌问题、提升泛化能力以及学习更具辨别力的特征<sup>[276, 320–323]</sup>。通过将正样本（相似或相关样本）与负样本（不相似或无关样本）进行对比，模型被引导去学习更深层次的特征表示，不再仅依赖于表面特征的匹配，而是掌握类别区分的潜在模式。引入负样本能够帮助模型在训练中识别错误的关联信号，避免因数据中的噪声或偏差而导致过度拟合。这种方式不仅提高了模型对不同类别间微小差异的感知能力，还增强了其在处理多样化数据时的鲁棒性。因此，负样本生成成为对比学习中不可或缺的关键机制，推动了模型在复杂场景下的表现优化。

在多模态大语言模型的研究中，负样本生成同样被证明是一项关键技术，用于解决模型在训练和推理中的问题<sup>[324–327]</sup>。这类研究通过负样本评估模型在图像与文本描述之间建立正确关联的能力。比如，ARO<sup>[324]</sup> 通过提供错误或无意义的图文配对，测试模型的区分能力，观察其是否能够识别负样本并避免错误关联。研究表明，让模型接触负样本可以显著提升其在多模态任务中的表现，使其在语义关联任务中更为精准，并具备更强的上下文理解能力。这种方法不仅优化了模型的整体性能，还增强了其在复杂和多样化场景下的稳健推理能力，从而进一步推动了多模态模型的发展。

### 7.3.3 多模态文本对齐

多模态文本对齐是多模态大语言模型中的核心任务，其目标是将视觉和语言信息精准关联，从而在多模态任务中实现更高级的语义理解能力。受指令微调在语言领域成功应用的启发，多模态大语言模型也开始引入指令微调和人类反馈强化学习，以提升多模态对话能力，并使模型输出更

加贴合人类需求。此外，多模态大模型处理文本丰富的图像理解面临特定挑战，相关领域也涌现出大量研究，推动了技术的持续发展。本节针对上述内容进行介绍。

### 1. 多模态指令微调与 RLHF

多模态指令微调通过在包含指令、输入和期望响应的监督数据上对多模态文本对齐进行优化，从而提升模型理解和执行复杂指令的能力。与大规模的预训练数据集相比，指令微调数据集的规模通常较小，其样本数量从几千到一百万不等<sup>[328]</sup>。代表性的视觉语言模型如 LLaVA、InstructBLIP 和 OpenFlamingo<sup>[329]</sup> 均引入了指令微调技术，显著提升了多模态任务的表现。

RLHF 则专注于通过人类反馈使模型输出更符合人类偏好。具体来说，RLHF 首先通过训练一个奖励模型来评估模型响应的质量，捕捉人类偏好的特征。借助这一奖励模型，RLHF 能有效模拟人类偏好，从而减少对人工标注的依赖。随后，通过奖励模型对多模态大语言模型进行微调，使其生成的响应更加贴合人类期望。

LLaVA<sup>[320]</sup> 通过指令微调提升了多模态对话能力，采用了 15 万条合成视觉指令样本进行训练。通过将预训练的 Vicuna 语言模型编码器与 CLIP ViT-L/14 视觉编码器的输出融合到相同的维度空间，LLaVA 在合成指令跟随任务和 Science QA 基准测试中表现出显著的改进。LLaVA 1.5<sup>[320]</sup> 在 LLaVA 的基础上进一步优化了多模态文本对齐能力。其改进包括引入跨模态全连接多层次感知机（MLP）层，并结合视觉问答（VQA）指令数据进行训练。LLaVA 1.5 仅使用 60 万条图文对数据，在 8 张 A100 GPU 上约一天即可完成训练。LLaVA-NeXT (v1.6)<sup>[330]</sup> 在 LLaVA 1.5 的基础上进行了多方面的改进，进一步推动了多模态文本对齐的性能。通过将全图和小图块的视觉特征分别输入视觉编码器，并将其拼接后处理，提高了图像分辨率的利用效率。优化了视觉指令调优数据集，新增了更好的视觉推理、OCR、世界知识和逻辑推理样本。

由于高质量视觉指令调优数据的稀缺，LLaVA 等模型可能在视觉和文本模态对齐上存在偏差，甚至生成幻觉性输出。为了解决这一问题，LLaVA-RLHF<sup>[262]</sup> 提出了基于人类反馈强化学习的创新方法——事实增强 RLHF (Factually Augmented RLHF)。该方法将 RLHF 从文本领域适配到视觉语言任务，通过在奖励模型中加入图像标题和真实多选题的额外事实信息，减少奖励滥用问题。LLaVA-RLHF 还利用 GPT-4 生成的训练数据及人工编写的图文对进一步提升其通用能力。在 LLaVA-Bench 中，其性能达到了 GPT-4 的 94%，在专注于减少幻觉的 MMHAL-BENCH 中，相较基线模型提升了 60%。

### 2. 富含文本信息的图像理解

富含文本信息的图像（Text-rich Image，如电影海报、书籍封面、文档扫描等）不仅需要模型理解视觉内容，还需要解析其中包含的细粒度文本信息，并与视觉语义进行有效关联。传统的多模态大语言模型在处理这类任务时往往面临文本识别能力不足、分辨率限制以及上下文信息捕获不充分等问题。为应对这些挑战，近年来涌现出一系列创新方法和模型，包括 LLaVAR、Monkey、Lumos 等，它们专注于提升文本丰富图像的理解能力。

LLaVAR<sup>[331]</sup> 针对多模态模型在理解图像中文本细节方面的不足，改进了视觉指令微调流程。通过引入包含大量文本的图像（如电影海报和书籍封面），该模型显著提升了文本细节处理能力。研究者使用 OCR 工具从 LAION 数据集中提取了 42.2 万张文本丰富的图像，并结合 GPT-4 生成了 1.6 万条基于这些图像的对话数据，每条数据包含多个问答对。将这些新生成的数据与现有的多模态指令跟随数据结合后，LLaVAR 显著改进了 LLaVA 模型的能力，在文本相关的视觉问答数据集上准确率提高了 20%，并在自然图像任务中取得了轻微提升。这表明，针对文本丰富图像的指令微调能够显著增强模型的文本理解和语义对齐能力。

当前大多数多模态模型的输入图像分辨率限制在  $224 \times 224$  像素，这是其视觉编码器架构的默认输入大小。这种限制导致模型在处理需要高分辨率和细节分析的文本任务时表现不佳。例如场景文本中心的 VQA（Scene Text-Centric VQA）、面向文档的 VQA（Document-Oriented VQA）以及关键信息提取（KIE）。Monkey<sup>[332]</sup> 针对这一问题，提出了一种高分辨率图像处理方法。使用滑窗方法将输入图像分割为多个与视觉编码器适配的图像块，每个图像块由静态视觉编码器独立处理，并通过 LoRA 调整和可训练的视觉重采样器增强。Monkey 支持处理分辨率可以达到  $1344 \times 896$  像素的图像，能够捕捉复杂视觉场景中的细节信息。Monkey 采用多级描述生成技术，丰富场景与对象之间的上下文关联。

Lumos<sup>[333]</sup> 提出了一种端云协同计算的多模态助手，专注于场景文本的识别与理解。引入了一个解耦的场景文本识别（Scene Text Recognition, STR）模块，作为多模态大语言模型的输入预处理层，包含四个子组件：感兴趣区域（Region Of Interest, ROI）检测、文本检测、文本识别和阅读顺序重建。ROI 检测识别图像中的显著区域并裁剪出包含关键信息的部分；文本检测从裁剪的图像中检测单词并输出边界框坐标；文本识别提取单词内容；阅读顺序重建根据图像布局将识别的单词组织成段落并排列阅读顺序。识别到的文本和坐标随后被传递到云端的多模态大语言模型进行处理。该解耦设计使 STR 模块能够在设备端运行，从而降低传输高分辨率图像到云端的计算成本和延迟。同时，STR 模块支持处理高达  $3000 \times 4000$  分辨率的图像，使其在复杂文本理解任务中表现优异，并与 Monkey 的高分辨率处理能力形成互补。

## 7.4 MiniGPT-4 实践

OpenAI 在 GPT-4 的发布会上展示了其多模态能力。例如，使用 GPT-4 可以生成非常详细与准确的图像描述、解释输入图像中不寻常的视觉现象、发现图像中蕴含的幽默元素，甚至可以根据一幅手绘的草图构建真实的前端网站。但是 GPT-4 的技术细节从未被正式公布，如何实现这些能力亟待研究。来自阿卜杜拉国王科技大学（King Abdullah University of Science and Technology (KAUST)）的研究人员认为，这些视觉感知能力可能来源于更先进的大语言模型的辅助。为了证实该假设，研究人员设计了 MiniGPT-4 模型，期望构造出类似于 GPT-4 的多模态能力。本章以 MiniGPT-4 为例，介绍多模态大语言模型实践。

### 7.4.1 MiniGPT-4 模型架构

MiniGPT-4 期望将来自预训练视觉编码器的图像信息与大语言模型的文本信息对齐，它的模型架构如图7.13所示，具体来说主要由三个部分构成：预训练的大语言模型 Vicuna<sup>[41]</sup>、预训练的视觉编码器，以及一个单一的线性投影层。

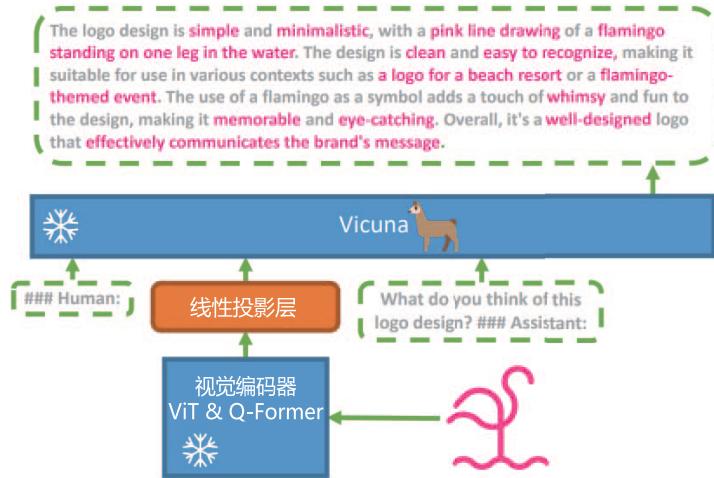


图 7.13 MiniGPT-4 的模型架构<sup>[252]</sup>

#### 1. Vicuna 模型

Vicuna 是一个基于解码器的大语言模型，它建立在 LLaMA<sup>[34]</sup> 的基础上，可以执行多种复杂语言任务。在 MiniGPT-4 中，它的主要任务是同时理解输入的文本与图像数据，对多个模态的信息具有感知理解能力，生成符合指令的文本描述。在具体的构建过程中，MiniGPT-4 并不从头开始训练大语言模型，而是直接利用现有的 Vicuna-13B 或 Vicuna-7B 版本，冻结所有的参数权重，降低计算开销。相关的预训练代码可以参考第 4 章和第 5 章的相关内容。

#### 2. 视觉编码器

为了让大语言模型具备良好的视觉感知能力，MiniGPT-4 使用了与 BLIP-2<sup>[263]</sup> 相同的预训练视觉语言模型。该模型由两个部分组成：视觉编码器 ViT<sup>[261]</sup> 和图文对齐模块 Q-Former。输入图像在传入视觉编码器后，首先会通过 ViT 做初步的编码，提取图像中的基本视觉特征，然后通过预训练的 Q-Former 模块，进一步将视觉编码与文本编码对齐，得到语言模型可以理解的向量编码。

对于视觉编码器 ViT，MiniGPT-4 使用了 EVA-CLIP<sup>[268]</sup> 中的 ViT-G/14 进行实现，初始化该模块的代码如下：

```

def init_vision_encoder(
    cls, model_name, img_size, drop_path_rate, use_grad_checkpoint, precision
):
    # 断言确保使用的ViT与当前版本的MiniGPT-4适配
    assert model_name == "eva_clip_g",
        "vit model must be eva_clip_g for current version of MiniGPT-4"

    # 创建Eva-ViT-G模型，这是一种特定的视觉基础模型
    visual_encoder = create_eva_vit_g(
        img_size, drop_path_rate, use_grad_checkpoint, precision
    )

    # 创建LayerNorm用于视觉编码器的标准化
    ln_vision = LayerNorm(visual_encoder.num_features)

    # 返回初始化的视觉编码器和标准化层
    return visual_encoder, ln_vision

```

在上段代码中，`img_size` 表示输入图像的尺寸；`drop_path_rate` 表示使用 `drop_path` 的比例，这是一种正则化技术；`use_grad_checkpoint` 表示是否使用梯度检查点技术来减少内存使用；`precision` 表示训练过程中的精度设置。该函数通过创建 ViT 视觉编码器模型，将输入图像转换为特征表示，以供进一步的处理。

对于图文对齐模块 Q-Former，在具体实现中通常使用预训练的 BERT 模型。它通过计算图像编码和查询（一组可学习的参数）之间的交叉注意力，更好地将图像表示与文本表示对齐。初始化该模块的代码如下：

```
def init_Qformer(cls, num_query_token, vision_width, cross_attention_freq=2):
    # 使用预训练的BERT模型配置Q-Former
    encoder_config = BertConfig.from_pretrained("bert-base-uncased")
    # 分别设置编码器的宽度与查询长度
    encoder_config.encoder_width = vision_width
    encoder_config.query_length = num_query_token
    # 在BERT模型的每两个块之间插入交叉注意力层
    encoder_config.add_cross_attention = True
    encoder_config.cross_attention_freq = cross_attention_freq

    # 创建一个带有语言模型头部的BERT模型作为Q-Former模块
    Qformer = BertLMHeadModel(config=encoder_config)
    # 创建查询标记并初始化，这是一组可训练的参数，用于查询图像和文本之间的关系
    query_tokens = nn.Parameter(
        torch.zeros(1, num_query_token, encoder_config.hidden_size)
    )
    query_tokens.data.normal_(mean=0.0, std=encoder_config.initializer_range)

    # 返回初始化的Q-Former模块和查询标记
    return Qformer, query_tokens
```

### 3. 线性投影层

视觉编码器虽然已经在广泛的图像-文本任务中做了预训练，但它本质上没有针对 LLaMA、Vicuna 等大语言模型做过微调。为了减小视觉编码器和大语言模型之间的差距，MiniGPT-4 中增加了一个可供训练的线性投影层，期望通过训练将编码的视觉特征与 Vicuna 语言模型对齐。通过定义一个可训练的线性投影层，将 Q-Former 输出的图像特征映射到大语言模型的表示空间，可便于结合后续的文本输入做进一步的处理和计算。创建该模块并处理图像输入的代码如下：

```

# 创建线性投影层，将经过Q-Former转换的图像特征映射到大语言模型的表示空间
# img_f_dim是图像特征的维度
# llama_model.config.hidden_size是大语言模型隐藏状态的维度
self.llama_proj = nn.Linear(
    img_f_dim, self.llama_model.config.hidden_size
)

# 输入图像后，MiniGPT-4完整的处理流程
def encode_img(self, image):
    device = image.device

    with self.maybe_autocast():
        # 使用视觉编码器对图像进行编码，再使用LayerNorm进行标准化处理
        image_embeds = self.ln_vision(self.visual_encoder(image)).to(device)

        # 默认使用冻结的Q-Former
        if self.has_qformer:
            # 创建图像的注意力掩码
            image_atts = torch.ones(image_embeds.size()[:-1], dtype=torch.long).to(device)

            # 扩展查询标记以匹配图像特征的维度
            query_tokens = self.query_tokens.expand(image_embeds.shape[0], -1, -1)

            # 使用Q-Former模块计算查询标记和图像特征的交叉注意力，更好地对齐图像和文本
            query_output = self.Qformer.bert(
                query_embeds=query_tokens,
                encoder_hidden_states=image_embeds,
                encoder_attention_mask=image_atts,
                return_dict=True,
            )
            # 通过线性投影层将Q-Former的输出映射到大语言模型的输入
            inputs_llama = self.llama_proj(query_output.last_hidden_state)
        # 创建大语言模型的注意力掩码
        atts_llama = torch.ones(inputs_llama.size()[:-1], dtype=torch.long).to(image.device)

        # 返回最终输入大语言模型的图像编码和注意力掩码
    return inputs_llama, atts_llama

```

为了减少训练开销、避免全参数微调带来的潜在威胁，MiniGPT-4 将预训练的大语言模型和视觉编码器同时冻结，只需要单独训练线性投影层，使视觉特征和语言模型对齐。如图7.13 所示，输入的粉色 logo 在经过一个冻结的视觉编码器模块后，通过可训练的线性投影层被转换为 Vicuna 可理解的图像编码。同时，输入基础的文本指令，例如：“你觉得这个 logo 怎么样？”大语言模型成功理解多个模态的数据输入后，就能产生类似“logo 的设计简约，用粉红色……”的全面图像描述。

### 7.4.2 MiniGPT-4 训练策略

为了获得真正具备多模态能力的大语言模型，MiniGPT-4 提出了一种分为两阶段的训练方法。第一阶段，MiniGPT-4 在大量的图像-文本对数据上进行预训练，以获得基础的视觉语言知识。第二阶段，MiniGPT-4 使用数量更少但质量更高的图像-文本数据集进行微调，以进一步提高预训练模型的生成质量与综合表现。

#### 1. MiniGPT-4 预训练

在预训练阶段，MiniGPT-4 希望从大量的图像-文本对中学习视觉语言知识，所以使用了来自 Conceptual Caption<sup>[318, 334]</sup>、SBU<sup>[335]</sup> 和 LAION<sup>[336]</sup> 的组合数据集进行模型预训练。以 Conceptual Caption 数据集为例，数据格式如图7.14 所示，包含基本的图像信息与对应的文本描述。



by Joi Ito

the trail climbs steadily  
uphill most of the way.



by Danail Nachev

the stars in the night sky.



by Justin Higuchi

musical artist performs on  
stage during festival.



by Viaggio Routard

popular food market showing  
the traditional foods from the  
country.

图 7.14 Conceptual Caption 数据集的格式

在第一阶段的训练过程中，预训练的视觉编码器和大语言模型都被设置为冻结状态，只对单个线性投影层进行训练。预训练共进行了约 2 万步，批量大小为 256，覆盖了 500 万个图像-文本对，在 4 块 NVIDIA A100 80GB GPU 上训练了 10 小时。以下代码示例有助于读者更好地理解 MiniGPT-4 的训练过程：

```

def forward(self, samples):
    image = samples["image"]

    # 对输入图像进行编码
    img_embeds, atts_img = self.encode_img(image)

    # 生成文本指令
    instruction = samples["instruction_input"] if "instruction_input" in samples else None

    # 将指令包装到提示中
    img_embeds, atts_img = self.prompt_wrap(img_embeds, atts_img, instruction)

    # 配置词元分析器以正确处理文本输入
    self.llama_tokenizer.padding_side = "right"
    text = [t + self.end_sym for t in samples["answer"]]

    # 使用词元分析器对文本进行编码
    to_regress_tokens = self.llama_tokenizer(
        text,
        return_tensors="pt",
        padding="longest",
        truncation=True,
        max_length=self.max_txt_len,
        add_special_tokens=False
    ).to(image.device)

    # 获取batch_size
    batch_size = img_embeds.shape[0]

    # 创建开始符号的嵌入向量和注意力掩码
    bos = torch.ones([batch_size, 1],
                    dtype=to_regress_tokens.input_ids.dtype,
                    device=to_regress_tokens.input_ids.device) *
          self.llama_tokenizer.bos_token_id
    bos_embeds = self.embed_tokens(bos)
    atts_bos = atts_img[:, :1]

    # 连接图像编码、图像注意力、文本编码和文本注意力
    to_regress_embeds = self.embed_tokens(to_regress_tokens.input_ids)
    inputs_embeds, attention_mask, input_lens = \
        self.concat_emb_input_output(img_embeds, atts_img,
                                     to_regress_embeds, to_regress_tokens.attention_mask)

    # 获得整体的输入编码和注意力掩码
    inputs_embeds = torch.cat([bos_embeds, inputs_embeds], dim=1)
    attention_mask = torch.cat([atts_bos, attention_mask], dim=1)

```

这段代码实现了整个 MiniGPT-4 模型的前向传播过程，包括图像和文本的编码、提示处理、多模态数据编码的连接，以及最终损失的计算。通过在 Conceptual Caption、SBU 等组合数据集上进行计算，即可获得预训练的 MiniGPT-4 模型。

在第一轮训练完成后，MiniGPT-4 获得了关于图像的丰富知识，并且可以根据人类查询提供合理的描述。但是它在生成连贯的语句输出方面遇到了困难，例如，可能会产生重复的单词或句子、碎片化的句子或者完全不相关的内容。这样的问题降低了 MiniGPT-4 与人类进行真实交流时流畅的视觉对话能力。

## 2. 高质量数据集构建

研究人员注意到，预训练的 GPT-3 曾面临类似的问题。虽然在大量的语言数据集上做了预训练，但模型并不能直接生成符合用户意图的文本输出。GPT-3 通过从人类反馈中进行指令微调和强化学习，产生了更加人性化的输出。借鉴这一点，研究人员期望预训练的 MiniGPT-4 也可以做到与用户意图对齐，增强模型的可用性。

为此，研究人员精心构建了一个高质量的、视觉语言领域的图像-文本数据集。该数据集的构建主要通过以下两个基本操作实现。

(1) 提供更全面的描述：为了使预训练的 MiniGPT-4 生成更加全面、更加综合的文本描述，避免生成不完整的句子，研究人员使用构建提示的策略，鼓励基于 Vicuna 的多模态模型生成给定图像的全面描述。具体的提示模板如下：

```
###Human: <Img><ImageFeature></Img> Describe this image in detail.  
Give as many details as possible. Say everything you see. ###Assistant:
```

其中，###Human 和 ###Assistant 分别代表用户输入和大语言模型的输出。<Img></Img> 作为提示符，标记了一张图像输入的起止点。<ImageFeature> 代表输入图像在经过视觉编码器和线性投影层后的视觉特征。在这步操作中，一共从 Conceptual Caption 数据集中随机选择了 5000 张图像，生成对应的、内容更加丰富的文本描述。

(2) 提供更高质量的描述：预训练的 MiniGPT-4 并不能生成高质量的文本描述，仍然存在较多的错误和噪声，例如不连贯的陈述、重复的单词或句子。因此，研究人员利用 ChatGPT 强大的语言理解和生成能力，让其作为一个自动化的文本质量评估者，对生成的 5000 个图像-文本对进行检查。期望通过这步操作修正文本描述中的语义、语法错误或结构问题。该步操作使用 ChatGPT 自动改进描述。具体的提示模板如下：