A Bottom-Up Survey of Transformer-Based Multimodal Models: From Model Layer to System Layer[1]

* * *

1 Introduction

- *It is well known that in the multimodal domain, Transformer-based architectures have been widely adopted for discriminative tasks such as VQA, as well as for generative tasks such as Image Captioning.*
- *Modern generative models are typically categorized into NAR and AR paradigms. The former includes four major classes discussed in the lectures, namely VAEs, GANs, Normalizing Flows, and Diffusion Models, while the latter mainly refers to autoregressive Transformers, which constitute the primary focus of this survey.*
- *This survey adopts a bottom-up perspective inspired by the layered design of computer networks, and systematically analyzes the commonalities and design principles of Transformer-based architectures in the multimodal field across three levels: model, agent, and system.*

2 Background

2.1 Model Layer

Early work such as CLIP[1] and ViT[2] demonstrated the effectiveness of large-scale contrastive and transformer-based representation learning for visual modalities. Subsequent extensions like ViLT[3] explored fusion of vision and language encoders, reducing reliance on heavy pretraining models.

Recent vision-language fusion models have pushed this paradigm further. For instance, LLaVA-1.5[4] integrates a frozen vision encoder with an instruction-tuned LLM, enabling fine-grained visual grounding in dialogue; mPLUG-2[5] advances multimodal alignment through its specialized fusion and inverse-fusion operator layers.

While numerous architectural variations exist across fusion mechanisms, a systematic taxonomy of backbone designs will be presented in the full paper.

2.2 Agent Layer

Chain-of-Thought (CoT)[6] , and its structured variants such as Tree-of-Thought (ToT)[7] and Graph-of-Thought (GoT)[8], introduce intermediate reasoning steps that improve multi-hop and compositional tasks. Frameworks like ReAct[9] and Reflexion[10] combine reasoning traces with tool usage, allowing models to iteratively refine their outputs through external actions.

[1]Parts of the ideas are inspired by my personal blog: `https://www.cnblogs.com/Arcticus/p/18265500`.

Retrieval-Augmented Generation incorporates external knowledge into model prompts, improving factual accuracy and adaptability. Supporting infrastructure such as vector databases provides scalable semantic search, enabling multimodal models to ground outputs in large knowledge bases.

2.3 System Layer

LangChain represents a prominent orchestration framework, offering modular pipelines for integrating LLMs, multimodal encoders, retrieval components, and external APIs.

Other workflow applications, such as ComfyUI, exemplify how modular design can facilitate extensibility and community-driven innovation.

3 Main Body

3.1 Model Layer

3.1.1 Upstream (Pretraining)

Transformers have many variants. AR Transformers adopt a sequential architecture similar to RNNs, whereas NAR models such as BERT[11] follow a fully parallel architecture. Interestingly, in practical settings, generative tasks typically employ only a decoder, while discriminative tasks tend to rely on a single encoder.

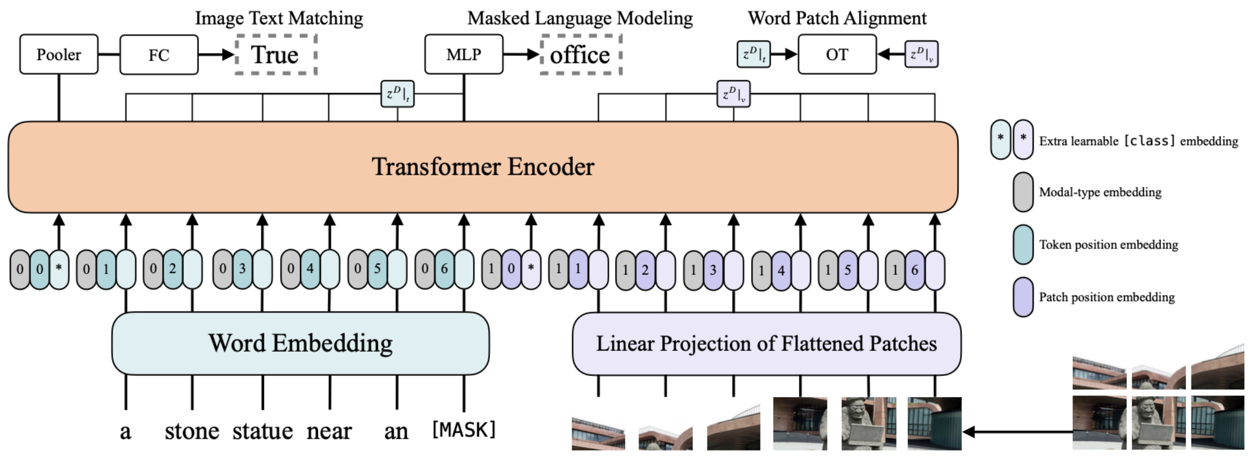


Figure 1: Overall pipeline of an encoder-only multimodal Transformer[3].

Common Pretraining Objectives

- **Image–Text Matching** — Binary classification of matched/unmatched image–text pairs

$$\mathcal{L}_{\text{ITM}} = -[y \log p_{\text{match}} + (1-y)\log(1-p_{\text{match}})]$$

- **Masked Language Modeling** — Predict masked textual tokens conditioned on visual features

$$\mathcal{L}_{\text{MLM}} = -\sum_{t\in\mathcal{M}} \log P(x_t \mid x_{\setminus\mathcal{M}}, v)$$

- **Word–Patch Alignment** — Align textual tokens with corresponding visual patches via cross-modal contrastive learning

$$\mathcal{L}_{\text{WPA}} = -\sum_{(i,j)\in\mathcal{P}} \log \frac{\exp(s(w_i, p_j)/\tau)}{\sum_k \exp(s(w_i, p_k)/\tau)}$$

Modality Fusion Module In multimodal models, a critical design component lies in how different modalities are fused before being fed into subsequent encoders or decoders.

The simplest strategy is to do nothing at all, as exemplified by CLIP[1], where alignment between modalities is achieved through a contrastive learning objective applied to separate encoders. A more direct approach is feature concatenation. More sophisticated methods, such as that adopted by ViLBERT[12], employ multi-layer neural networks to explicitly integrate multimodal representations before passing them to the encoders.

The complexities of different modality fusion strategies are illustrated in all possible combinations in the figure below.

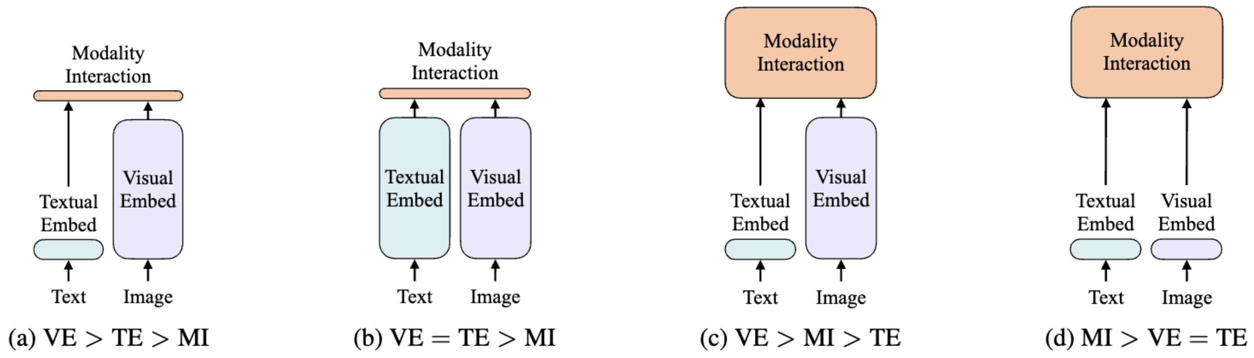


Figure 2: Modality fusion modules with different parameter allocations[3].

For example, CLIP[1] corresponds to type (b), whereas recent multimodal systems implement (c)(d).

3.1.2 Midstream (Alignment)

In practice, explicit modality fusion at the pretraining stage is relatively rare. For the majority of existing models, modality fusion is primarily carried out during the midstream alignment stage.

Fusion Mechanisms The four types of attention mechanisms in multimodal models are Co-Attention, Merged Attention, Modality-Specific Attention, and Asymmetric Attention, as illustrated in the figure below.

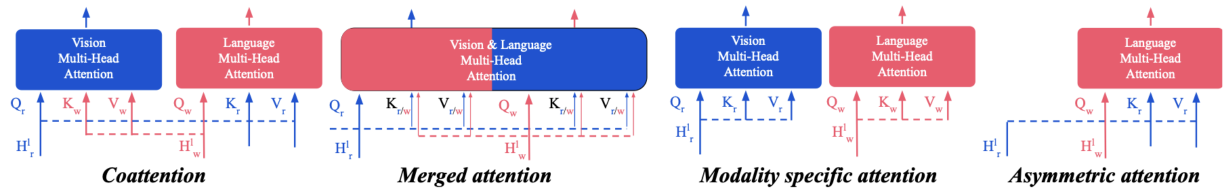

Vision Multi-Head Attention
Language Multi-Head Attention
Vision & Language Multi-Head Attention
Coattention
Merged attention
Modality specific attention
Asymmetric attention

Figure 3: Four fundamental fusion mechanisms[13].

Although different modalities have been preliminarily aligned during the upstream stage, in the midstream or downstream phase, models typically introduce small task-specific modules or apply task-oriented modifications to the attention layers in order to better adapt to individual tasks.

Examples of Modern Mechanisms

- **Projection-based Interaction**
  → *e.g., LLaVA*[14]: maps visual embeddings into the LLM token space.

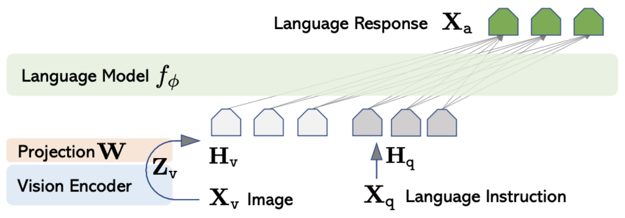


Figure 4: Modality interaction in LLaVA[14] .

  It was the first to introduce GPT-4[15] as a teacher model, generating image–text pair datasets for instruction tuning training. Compared with the original version of LLaVA[14] , LLaVA-1.5[4] replaced the linear projection in the alignment process with an MLP and scaled up with various new datasets.

- **Gating Mechanisms**
  → *e.g., MM-CoT*[16]: selectively regulate cross-modal information flow.

  A gated fusion mechanism is adopted to integrate the language representation H_{language} and the visual representation H_{vision}. The resulting fused representation $H_{\text{fuse}} \in \mathbb{R}^{n\times d}$ is computed as follows:

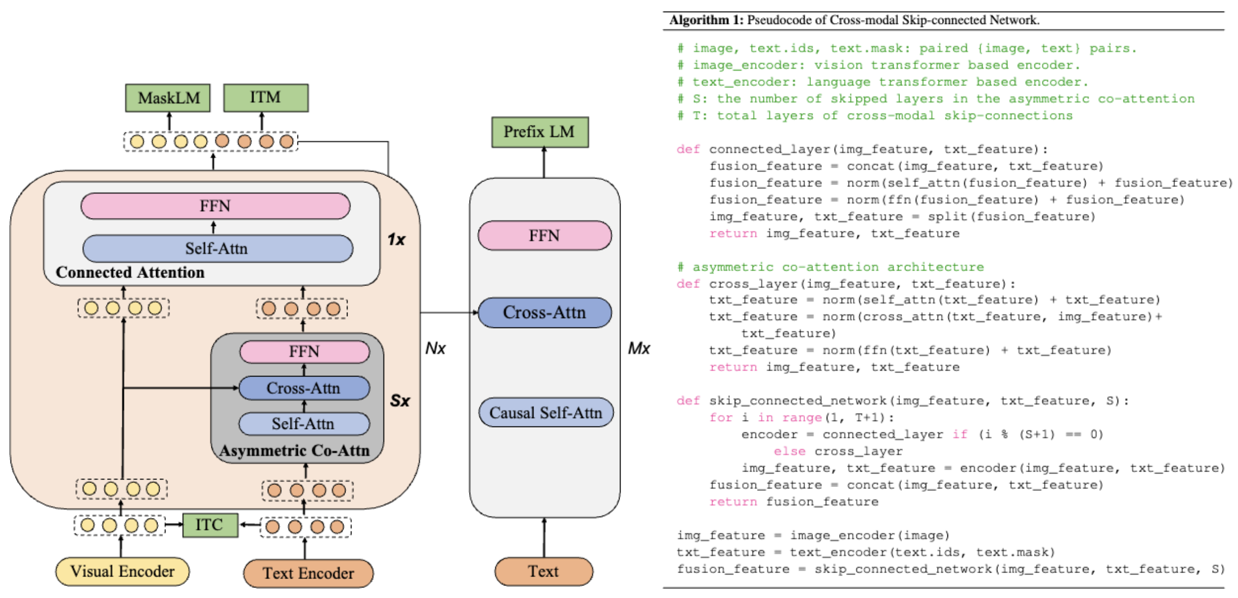
$$\lambda = \text{Sigmoid}(W_l H_{\text{language}} + W_v H_{\text{vision}}^{\text{attn}}),$$

$$H_{\text{fuse}} = (1-\lambda)\cdot H_{\text{language}} + \lambda \cdot H_{\text{vision}}^{\text{attn}},$$

  where W_l and W_v denote learnable parameters.

  It serves as the SoTA model for ScienceQA in 2024.

- **Indicator-based Fusion**
  → *e.g., mPLUG*[17]: employs learned modality indicators for adaptive feature mixing.



Algorithm 1: Pseudocode of Cross-modal Skip-connected Network.

```
# image, text.ids, text.mask: paired {image, text} pairs.
# image_encoder: vision transformer based encoder.
# text_encoder: language transformer based encoder.
# S: the number of skipped layers in the asymmetric co-attention
# T: total layers of cross-modal skip-connections

def connected_layer(img_feature, txt_feature):
    fusion_feature = concat(img_feature, txt_feature)
    fusion_feature = norm(self_attn(fusion_feature) + fusion_feature)
    fusion_feature = norm(ffn(fusion_feature) + fusion_feature)
    img_feature, txt_feature = split(fusion_feature)
    return img_feature, txt_feature

# asymmetric co-attention architecture
def cross_layer(img_feature, txt_feature):
    txt_feature = norm(self_attn(txt_feature) + txt_feature)
    txt_feature = norm(cross_attn(txt_feature, img_feature)+
        txt_feature)
    txt_feature = norm(ffn(txt_feature) + txt_feature)
    return img_feature, txt_feature

def skip_connected_network(img_feature, txt_feature, S):
    for i in range(1, T+1):
        encoder = connected_layer if (i % (S+1) == 0)
            else cross_layer
        img_feature, txt_feature = encoder(img_feature, txt_feature)
    fusion_feature = concat(img_feature, txt_feature)
    return fusion_feature

img_feature = image_encoder(image)
txt_feature = text_encoder(text.ids, text.mask)
fusion_feature = skip_connected_network(img_feature, txt_feature, S)
```

Figure 5: Modality interaction in mPLUG[17].

The translation adopts a hybrid design that combines Asymmetric Attention and Merged Attention. In the Asymmetric Attention, a fixed stride is used to skip part of the input tokens. This design is trained during both the alignment stage and the fine-tuning stage.

3.1.3 Downstream (Fine-tuning)

After completing pretraining and alignment, the entire model can be fine-tuned using task-specific datasets.

Fine-tuning Categories

- **Prompt Tuning**
  Prefix and suffix tuning are also counted as prompt tuning. Introduces a small set of learnable prompt tokens to the input sequence.

  Note that this is not prompt engineering. Instead, the input tokens undergo an explicit prompt optimization process as part of the model before being fed into the subsequent encoder.

- **Instruction Tuning**
  Fine-tunes the model on instruction–response pairs to align with human-style commands.

  This is arguably one of the most common paradigms among all fine-tuning approaches, and it is also frequently associated with methods such as RL and RLHF.

- **Adapter Tuning**
  Inserts lightweight adapter modules into each Transformer block.

This is a commonly adopted parameter-efficient training strategy, where lightweight modules are inserted between layers while the remaining parameters are kept frozen. It serves as a practical and effective solution under low-compute-resource settings.

- **LoRA Tuning**
  Decomposes the weight update matrices into low-rank components that are trained while keeping the original model weights frozen.

  The detailed underlying principles are related to optimization theory and are typically covered in specialized optimization courses. In practice, this category includes a wide range of algorithms, and different methods are adopted depending on the specific conditions and constraints.

Parameter Optimization Strategies A critical design decision involves determining which modules to freeze during fine-tuning.

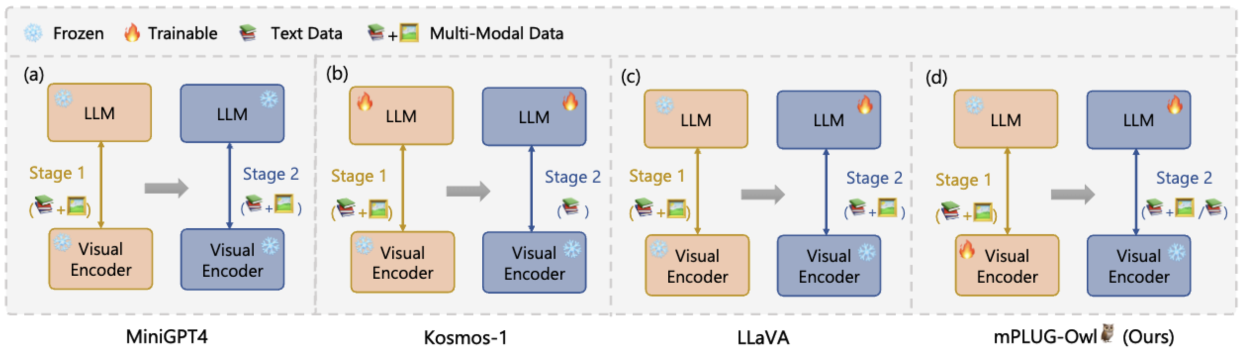


Figure 6: An overview of four common parameter optimization strategies for fine-tuning[18].

3.2 Agent Layer

3.2.1 Agents

The above discussion focuses on the model-structure level. In contrast, an agent is defined as an integrated object composed of one or multiple models and external tools. These external tools include reasoning chains, memory, databases, and related components. It should be noted that this definition of an agent is different from that in RL.

3.2.2 External Tools

- **ReAct[9]**
  ReAct[9] is an agent interaction paradigm that interleaves explicit reasoning with external action execution.

  The ReAct[9] model operates in a continuous loop of Thought → Action → Observation.

- **CoT[6]**
  Chain-of-Thought is a prompting strategy that elicits explicit intermediate reasoning steps from large language models.

Although numerous variants have been proposed, such as Tree-of-Thoughts[7], Graph-of-Thoughts[8], and even cyclic decision-making schemes, these methods fundamentally remain within the scope of pure prompt engineering.

- **Vector Database**
  A vector database is a specialized database system designed for storing, indexing, and efficiently retrieving high-dimensional vector embeddings based on similarity search.

  It serves as a core infrastructure for semantic retrieval, long-term memory, and retrieval-augmented generation in modern LLM-based systems.

- **RAG**
  Retrieval-Augmented Generation is a framework that integrates external knowledge retrieval with generative modeling, where relevant information is first retrieved from a knowledge source such as a vector database and then fused into the input of a language model to condition the generation process.

  Common forms of RAG include conventional vector-based retrieval RAG, as well as more modern approaches such as GraphRAG[19][2] based on graph algorithms.

3.3 System Layer

As expected (analogous to the relationship between models and agents), the system layer integrates one or multiple agents into a unified large-scale system. Each agent, as a constituent component of the system, plays its own functional role, and can be organized in either a sequential or a parallel manner to accomplish tasks collaboratively. From the designer's perspective, this layer is more closely related to software engineering, particularly the paradigm of Software as a Service (SaaS).

3.3.1 Notion of Workflow

At the system level, the entire architecture is often described in terms of a *workflow*. Moreover, the VQA systems that users actually interact with in practice require comprehensive service-level support, rather than being limited to the invocation of a single agent or model API.

It is natural to construct task-specific workflows that can be reused by different users under varying initial conditions or contexts. Users neither need nor are expected to understand the underlying code below the agent level; instead, they only need to be aware of the expected inputs and outputs of an agent or model, as well as the functionality of each module as described through natural language.

3.3.2 Common System Frameworks

- *LangChain* is one of the most widely adopted and lightweight frameworks for organizing LLMs. It is typically designed for a single-agent setting and follows a canonical interaction pattern: User $\rightarrow$ LLM Agent $\leftrightarrow$ Tools $\rightarrow$ Output.

[2]Due to space limitations, a detailed blog post on GraphRAG has been written by me: `https://www.cnblogs.com/Arcticus/p/18344606`.

- *AutoGen*[20] follows a multi-agent interaction pipeline, typically structured as: User → Planner Agent → Coder Agent → Critic Agent → Tool Agent, where each agent internally consists of one or multiple core models augmented with its own role-specific set of tools.

4 Conclusion

From a systematic perspective, this survey organizes the framework of modern popular multimodal models into three hierarchical layers: the model layer, the agent layer, and the system layer.

At the model layer, the discussion is structured around the upstream–midstream–downstream pipeline, corresponding to the stages of pretraining, alignment, and fine-tuning. Each stage is illustrated with representative models, explaining how multiple modalities are jointly integrated, and how the resulting hidden representations are processed through diverse and sophisticated architectures to support different multimodal tasks.

At the agent layer, I first define an agent as an integration of a model and external tools. I then introduce representative techniques such as ReAct[9], CoT[6] , Vector Databases, and Retrieval-Augmented Generation, to demonstrate how these tools can substantially enhance the capability of a model in solving specific tasks.

At the system layer, I explain the concept of workflow and provide practical examples to illustrate how one or multiple agents can be composed into a complete agent system. Typical cases include the single-agent paradigm adopted by LangChain, as well as the multi-agent collaboration paradigm exemplified by AutoGen[20].

* * *

This survey is compiled based on a review of around 20 research papers. It serves not only as the final project for the course CS 680, but also as a structured summary of my understanding of transformer-based multimodal models, ranging from low-level model architectures to high-level application systems. Through the process of studying the literature, comparing different approaches, and identifying both the similarities and differences among various works, I have gained substantial insights into this research field.

References

[1] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021.

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

[3] Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*. PMLR, 2021.

[4] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023.

[5] Shengze Ye, Yu Qiao, Yuxin Zhao, Yang Deng, Peng Cao, Jun Chen, Yi Liu, Wei Ke, Xiaofeng Yang, et al. mplug-2: A modularized multi-modal foundation model across text, image and video. *arXiv preprint arXiv:2302.00402*, 2023.

[6] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[7] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuanfang Xu, Bo Dai, Yelong Shen, and Danqi Chen. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

[8] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michał Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 38, pages 17682–17690. AAAI Press, 2024.

[9] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[10] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS) 2023*, 2023.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[12] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[13] Lisa Anne Hendricks, John Mellor, Rosalia Schneider, Jean-Baptiste Alayrac, and Aida Nematzadeh. Decoupling the role of data, attention, and losses in multimodal transformers. *Transactions of the Association for Computational Linguistics (TACL)*, 9:570–585, 2021.

[14] Haotian Liu, Chunyuan Lin, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.

[15] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[16] Yi Zhang, Hao Li, Xinyang Liu, Peng Zhang, Yixin Cao, and Qun Liu. Multimodal chain-of-thought reasoning in language models. *arXiv preprint arXiv:2302.00923*, 2023.

[17] Shengze Ye, Peng Cao, Jun Chen, Yi Liu, and Yu Qiao. mplug: Effective and efficient vision-language learning by cross-modal skip-connections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[18] Shengze Ye, Jun Chen, Yuxin Zhao, Yang Deng, Peng Cao, Yi Liu, and Yu Qiao. mplug-owl: Modularization empowers large language models with multimodal reasoning. *arXiv preprint arXiv:2304.14178*, 2023.

[19] D. Edge et al. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

[20] Qingyun Wu, Gagan Bansal, Jeffrey Zhang, Yiran Wu, Bei Li, Suresh Santhanam, Zixuan Zhang, Junjie Wang, Oscar Gonzalez, Yanjun Peng, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.