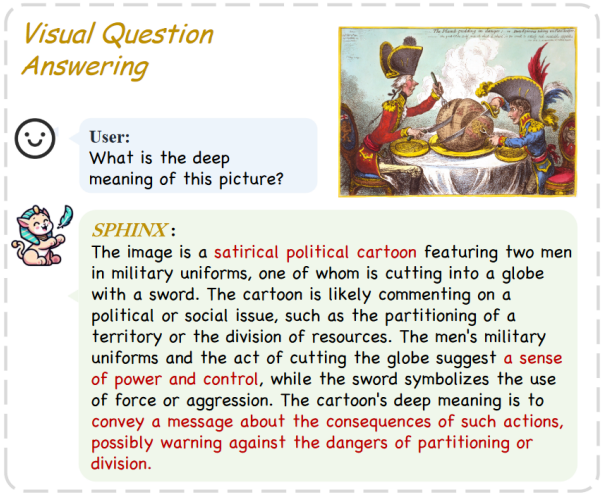
Multimodal LLMs (MLLMs)

Stanislav Fedotov, Practical Generative AI course

January 2024

1 tl;dr

1. A typical multimodal LLM (MLLM) processes images and texts and outputs texts. For example, it can solve the Visual Q&A task (like "How many apples are on this image?"). However, more advanced MLLMs can also process sound, videos etc.



2. There are high-quality proprietary models such as GPT-4V(ision) which are cool but completely obscure in terms of architecture and training, so we can only rejoice at their successes, see:

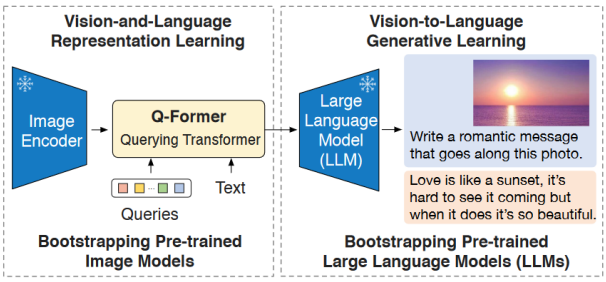
   - The Dawn of LMMs: Preliminary Explorations with GPT-4V(ision), Microsoft, Sep 2023 showcasing the capabilities of GPT-4V,
   - Two papers comparing GPT-4V with Gemini (one also considers open source SPHINX).

   However, there are some open source options which we'll check in this longread.

3. A typical MLLM is a Frankenstein creature assembled of three parts:

   - A pre-trained image encoder, making a vector or a sequence of vectors out of an image.
   - A pre-trained LLM that produces the output.
   - A connector mechanism that maps image encoder outputs and text prompt into the input space of the LLM.

   Here is an example of MLLM architecture (BLIP-2):



4. An image encoder is often a version of CLIP. For example, EVA-CLIP is quite popular.

5. An LLM is the main ingredient. The job of all other components is just to supply the LLM with high-quality, informative input, while it does all the reasoning. More capable LLMs tend to give more capable MLLMs. Vicuña is quite popular in open source MLLMs.

6. The connector mechanism is the place where researchers can apply their creativity. It can be as simple as one or two layer MLP projection

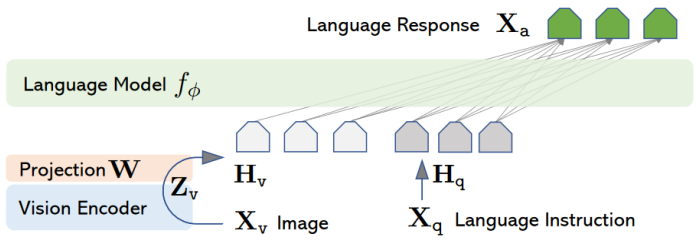
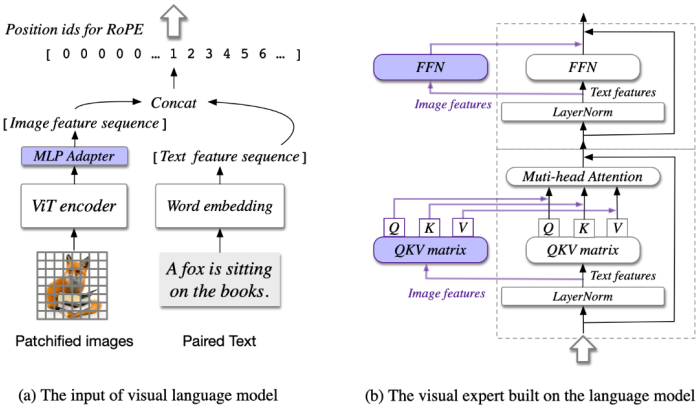


Figure 1: LLaVA network architecture.

(this can resemble prompt tuning) or invasive, resembling LoRA



(a) The input of visual language model

(b) The visual expert built on the language model

Figure 3: The architecture of CogVLM. (a) The illustration about the input, where an image is processed by a pretrained ViT and mapped into the same space as the text features. (b) The Transformer block in the language model. The image features have a different QKV matrix and FFN. Only the purple parts are trainable.

7. On high level, the training process has two potential stages:

   - Pre-training on image captioning data (such as LAION or COCO). Here, the model trains to align text and image data. However, thanks to the LLM capabilities, pre-trained MLLM often exhibits few-shot visual Q&A and other abilities. Improving quality of captions may lead to better models, see ShareGPT4V paper, for example.

     Image encoder usually stays frozen, LLM often too, though sometimes it is also fine-tuned or paramertic-efficiently fine-tuned (with smth like LoRA). The connector is alwas trainable.
   - (optional) Instructional fine tuning. This stage directly teaches an MLLM to follow instructions such as "Tell me what is the eye color of the person standing to the left of a golden retriever". However,

even creating instructions even for LLMs is prohibitively expensive for open source developers, and even more so for MLLMs.

So, people create synthetic data using datasets for supervised learning (Visual Q&A, for example) or asking GPT-4 to create instructions and answers based on captions and object bounding boxes.

Context type 1: Captions
A group of people standing outside of a black vehicle with various luggage.
Luggage surrounds a vehicle in an underground parking area
People try to fit all of their luggage in an SUV.
The sport utility vehicle is parked in the public garage, being packed for a trip
Some people with luggage near a van that is transporting it.

Context type 2: Boxes
person: [0.681, 0.242, 0.774, 0.694], backpack: [0.384, 0.696, 0.485, 0.914], suitcase: ...<omitted>

Response type 1: conversation
Question: What type of vehicle is featured in the image?
Answer: The image features a black sport utility vehicle (SUV) ...<omitted>

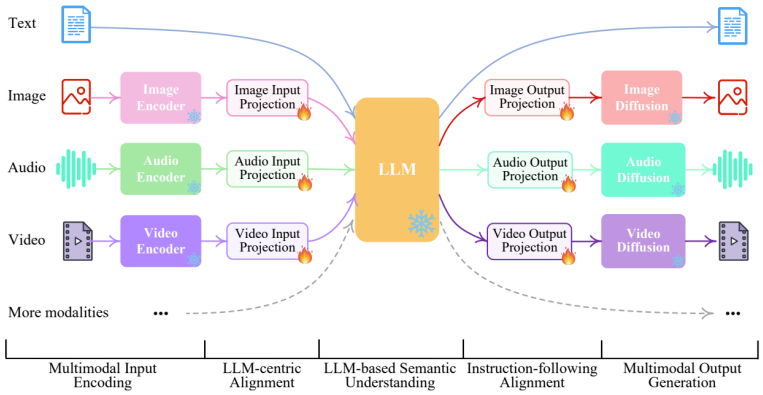
Response type 2: detailed description
The image is an underground parking area with a black sport utility vehicle (SUV) parked. There are three people in the scene, with one person standing closer to the left side of the vehicle, another person in the middle, and the third person on the right side. They are all working together to pack their luggage into the SUV for a trip. ...<omitted>

Response type 3: complex reasoning
Question: What challenges do these people face?
Answer: In the image, a group of people is standing outside a black SUV in a parking area, surrounded by various pieces of luggage, including suitcases and backpacks. They are facing the challenge of fitting all their luggage into the black SUV. There are multiple suitcases and backpacks to be packed, which suggests that the group has a significant amount of belongings ...<omitted>

Table 1: One example to illustrate the instruction-following data. The top block shows the contexts such as captions and boxes used to prompt GPT, and the bottom block shows the three types of responses. Note that the visual image is not used to prompt GPT, we only show it here as a reference.

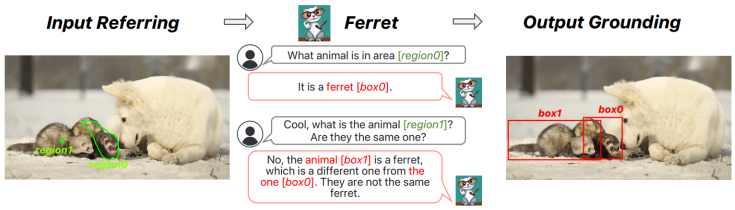
The quality of these instructions sometimes leaves much to be desired, and researchers are able to tune quality a bit by just fixing mistakes.

8. Sometimes authors are able to make an MLLM not only to process things from different modalities, but also output both images, texts, videos and other things. See NExT-GPT: Any-to-Any Multimodal LLM, for example. In any case, the LLM stays the main think tank and all others components provide anything→text and text→anything transformations.



However, I know at least one paper where non-text modalities (images, text, audio, depth, thermal, and IMU data) are mapped into one representation space by aligning with image data, see IMAGEBIND: One Embedding Space To Bind Them All.

9. If you want to see, how MLLMs are able to work with requests concerning arbitrary subregions of an image, such as



please check Ferret paper.

Now, if you're still curious, let's see how these principles show up in several important models.

2 BLIP-2

Introduced in BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models, Jan' 23

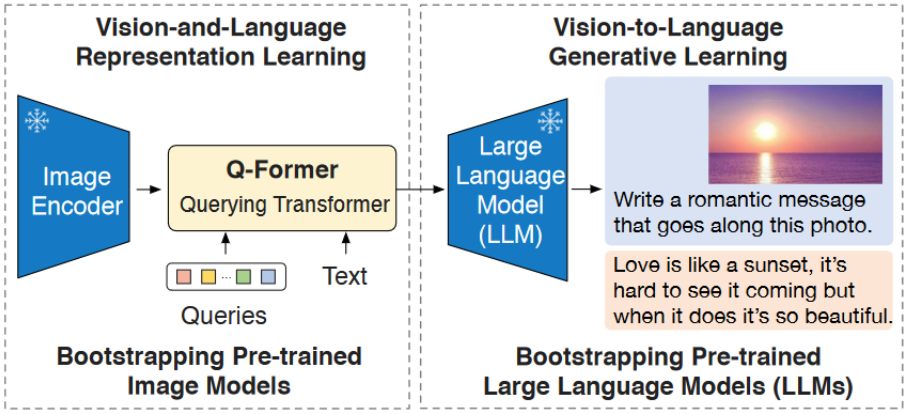
2.1 Architecture

The BLIP-2 input is (image, text_prompt), for example, an image of a cat and a question "What is the color of the cat's eyes?".

The architecture is quite typical. We have:

- A frozen and pre-trained image encoder model (image encoder from a CLIP model) and
- A frozen and pre-trained LLM (an OPT model).

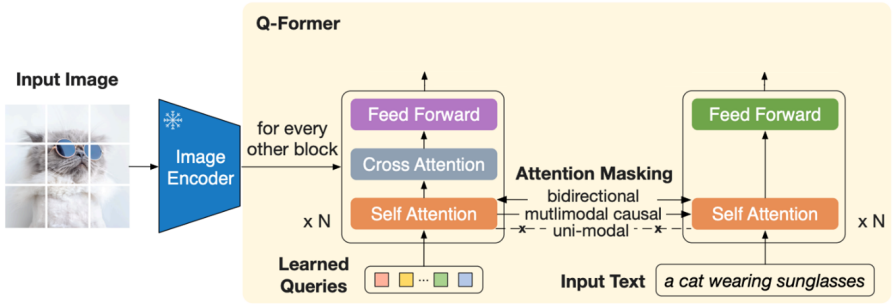
They are both shown in blue with snowflakes:



Vision encoder extracts information from the image, LLM does all the hard work producing the output. But we still need to:

- Process the text of a prompt,
- Map image representation to the world of text tokens.

BLIP-2 does it with a lightweight Querying Transformer ("Q-former"):



It consists of two transformer components sharing the same self-attention weights:

1. The visual component (to the left) receives information from the image using cross attention. Its input is a sequence of trainable "query embeddings". You may roughly consider them as embeddings of "Tell me what the property X of the given image is" tokens.

2. The text component (to the right) that receives text prompt as an input.

Together, these two components produce sequence of embeddings, and a fully connected layer ("projection") turns it into LLM inputs.

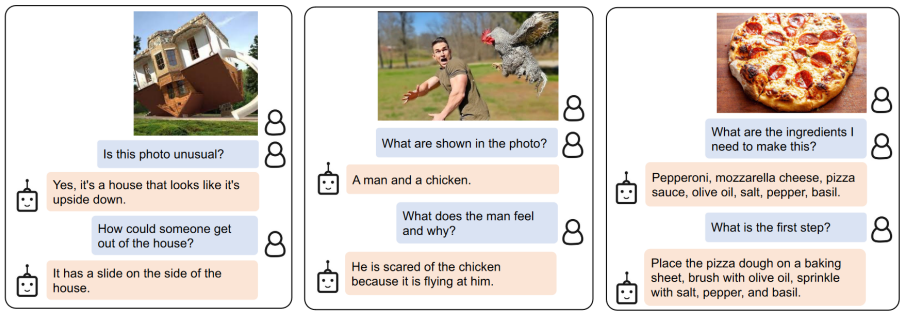
2.2 Training

In this subsection we discuss the **pre-training stage** which corresponds to the pre-training stage of LLMs.

An important remark. Pre-training of BLIP-2 is done solely on image caption data, no instructions. So, during pre-training models never sees actual (image, prompt) $\longrightarrow$ output correspondence. The authors have to perform two-stage training:

- Stage 1: image + prompt = alignment,
- Stage 2: (image, "") $\longrightarrow$ output.

Nevertheless, BLIP-2 is capable of Instructed Zero-shot Image-to-Text Generation like:

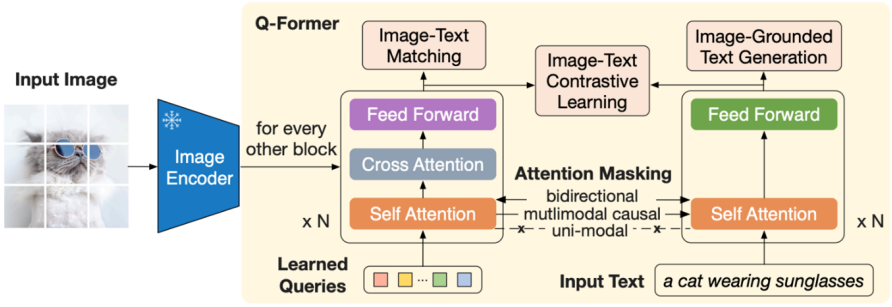


But for this we mostly need to thank the LLM whose capabilities we manage not to harm during training.

Now, let's look at the two training stages.

Vision-language representation learning stage with a frozen image encoder

Please note that on this stage the LLM is not used at all and therefore it stays frozen. We only work with frozen image encoder and Q-former and we train them for three tasks (i.e. with three loss functions) that represent different forms of joint image+text understanding:



- Image-Text Matching (ITM) is a binary classification task where the model is asked to predict whether an image-text pair is positive (matched) or negative (unmatched).
- Image-grounded Text Generation (ITG) loss trains the Q-Former to generate texts, given input images as the condition.
- Image-Text Contrastive Learning (ITC) learns to align image representation and text representation. It achieves so by contrasting the image-text similarity of a positive pair against those of negative pairs.

After this stage Q-former is able to extract information from images and align it with information from the text prompt.

Bootstrap Vision-to-Language Generative Learning from a Frozen LLM

Please note that on this stage the image encoder stays frozen as well as the LLM.

After the previous stage, the Q-former is able to output joint representations of the input image (prompts are not used). This stage will align these representations with LLM.

Basically, we train Q-former to supply inputs to the LLM such that the LLM produces right answers.

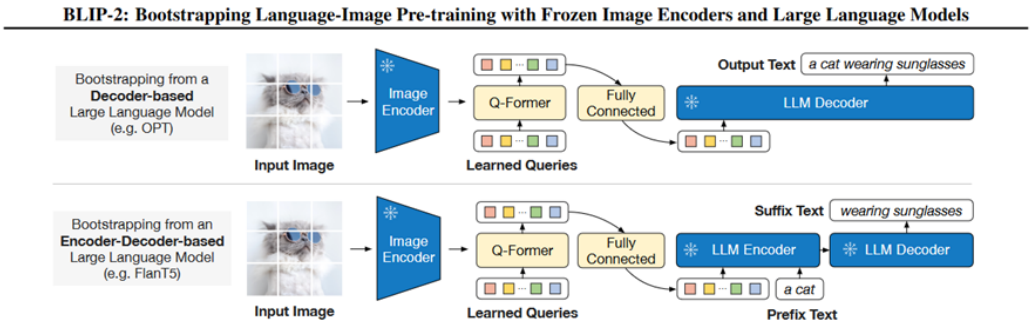


Figure 3. BLIP-2's second-stage vision-to-language generative pre-training, which bootstraps from frozen large language models (LLMs). **(Top)** Bootstrapping a decoder-based LLM (e.g. OPT). **(Bottom)** Bootstrapping an encoder-decoder-based LLM (e.g. FlanT5). The fully-connected layer adapts from the output dimension of the Q-Former to the input dimension of the chosen LLM.

2.3 Instruction tuning

Introduced in InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning, June' 23.

The authors take almost the same architecture as in BLIP-2, but use Vicuña instead of OPT and they plug in instructions ("prompt" in the subsections above) two times: into the text processor of the Q-former and into the final LLM itself:

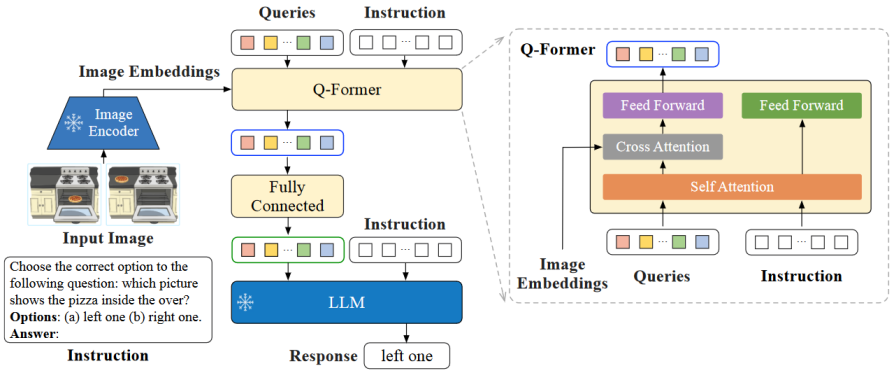


Figure 3: Model architecture of InstructBLIP. The Q-Former extracts instruction-aware visual features from the output embeddings of the frozen image encoder, and feeds the visual features as soft prompt input to the frozen LLM. We instruction-tune the model with the language modeling loss to generate the response.

The largest problem is data. There was no instruction data except for what was generated for LLaVA-Instruct-150K (see next section). So, the authors took 26 datasets covering different particular tasks such as visual reasoning, visual Q&A, image classification etc, and transformed them to instruction format using specific task-dependent prompt templates such as:

- ⟨Image⟩Given the image, answer the following question with no more than three words. {Question} [for Visual Q&A]
- ⟨Image⟩Could you use a few words to describe what you perceive in the photo? [for image captioning]

Visual instruction tuning

3 LLaVA

Introduced in Visual Instruction Tuning, Apr' 23

Chronologically, it appeared before InstructBLIP.

3.1 Architecture

It is a way simpler than BLIP-2. Here, we just use a single fully connected layer (projection) to connect outputs of the image encoder and inputs of the LLM:

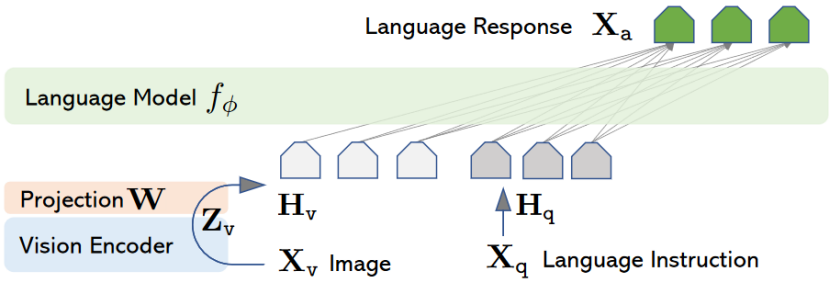


Figure 1: LLaVA network architecture.

The authors later showed that taking two-layer MLP as a projection component significantly improves quality.

3.2 Training

It is done in two stages:

Pre-training. As in BLIP-2, the model is trained on image captioning data: (image, -) $\longrightarrow$ output.

Note that we don't use the text modality in input. We can afford this, because we'll account for this on the next training stage.

The vision encoder and LLM are frozen during pre-training. Only the projection layer is trained.

Instruction tuning. On this stage both projection layer and LLM are fine tuned.

Where do we get the data? At this point, almost no image-based instruction data is available, and gathering it with humans is way too expensive. So, the authors create synthetic data with help of GPT-4!

To do this, they transform pictures into the following two types of text representation:

- Comprehensive captions,
- Bounding boxes for various objects on the pictures.

Now, GPT-4 is used to create the following types of instruction data from captions and bounding boxes:

- Conversation between the assistant and a person asking questions about the image. The answers are in a tone as if the assistant is seeing the image and answering the question.
- Detailed description.
- Complex reasoning.

This way, the authors are able to collect 158k unique language-image instructions. Examples:

Context type 1: Captions
A group of people standing outside of a black vehicle with various luggage.
Luggage surrounds a vehicle in an underground parking area
People try to fit all of their luggage in an SUV.
The sport utility vehicle is parked in the public garage, being packed for a trip
Some people with luggage near a van that is transporting it.

Context type 2: Boxes
person: [0.681, 0.242, 0.774, 0.694], backpack: [0.384, 0.696, 0.485, 0.914], suitcase: ...<omitted>

Response type 1: conversation
Question: What type of vehicle is featured in the image?
Answer: The image features a black sport utility vehicle (SUV) ...<omitted>
Response type 2: detailed description
The image is an underground parking area with a black sport utility vehicle (SUV) parked. There are three people in the scene, with one person standing closer to the left side of the vehicle, another person in the middle, and the third person on the right side. They are all working together to pack their luggage into the SUV for a trip. ...<omitted>
Response type 3: complex reasoning
Question: What challenges do these people face?
Answer: In the image, a group of people is standing outside a black SUV in a parking area, surrounded by various pieces of luggage, including suitcases and backpacks. They are facing the challenge of fitting all their luggage into the black SUV. There are multiple suitcases and backpacks to be packed, which suggests that the group has a significant amount of belongings ...<omitted>

4 SPHINX

Introduced in SPHINX: the joint mixing of weights, tasks, and visual embeddings for multi-modal large language models, Nov' 23

On top-level, it's much alike LLaVA:

- Several frozen image embedding models are connected to a final LLM through projection layers.
- Training has two stages: pre-training on image captions and instruction tuning, which is done on mostly synthetic data (including the data generated by creators of previous instruct-tuned MLLMs).

However, there are four interesting features, so let's go through each of them.

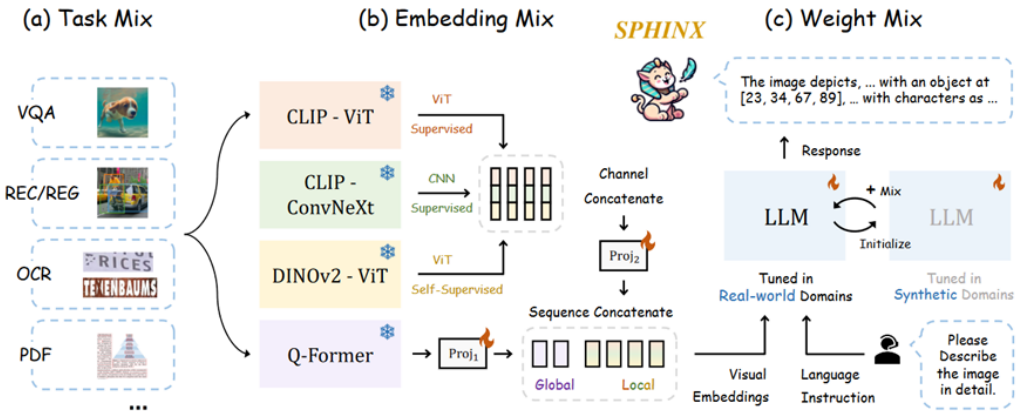


Figure 3: **The joint mixing paradigm of *SPHINX*.** with mixed tuning tasks (a), mixed visual embeddings (b), and mixed model weights (c).

4.1 Unfreezing LLM for stage-1 pre-training

This allows to better align the LLM with vision-language data. To prevent deterioration of pure text-text alignment, LLM is in the same time tuned on RefinedWeb dataset (the one from the Falcon paper).

4.2 Mixed model weights

In pursuit of getting more data, the authors have to use not only real-world image descriptions (such as LAION-400M), but also synthetic ones (such as LAION-COCO). Training on data from two different distributions, especially when it's done consequently, can lead to problems. So, what the authors do is:

- First, pre-train on the most common domain data (LAION-400M) to endow the MLLM with fundamental visual understanding capabilities. Save the resulting weights θ_{real}.
- Then, fine-tune on synthetic data (LAION-COCO) to get θ_{syn}.
- Then, take weighted average of the weights: $\theta_{mix} = \beta \cdot \theta_{real} + (1-\beta)\theta_{syn}$.

4.3 Mixed tuning tasks for stage-2 fine-tuning

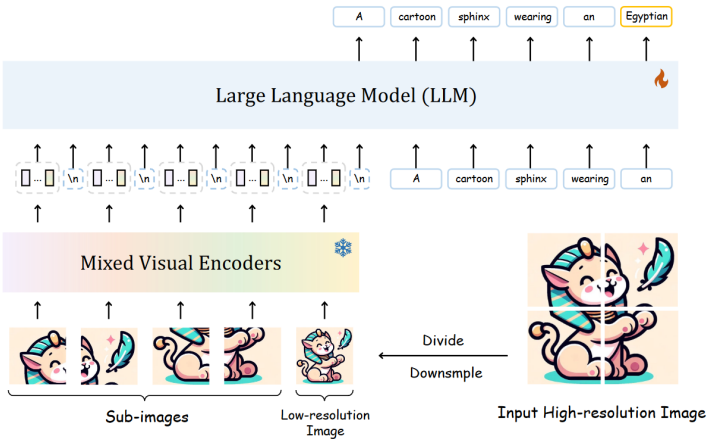
The authors claim that previous open-source MLLMs can only perform simple visual question answering (VQA) and single large object referring. To avoid this, they collect instruction data from a wider range of multi-modal tasks including multi-object detection and relation reasoning, text-oriented chart/document VQA, and human pose estimation.

4.4 Mixed embeddings for visual encoding

Instead of just one image encoder, SPHINX uses a variety of vision backbones with

- Different network architectures: CNN and ViT,
- Different pre-training paradigms: supervised (like text-supervised CLIP) and self-supervised (like DINOv2),
- Different information granularity. Both CNNs and ViT work on patch level, so the authors adopt Q-Former to summarize visual embeddings via querying from the global context

4.5 Working in higher resolutions

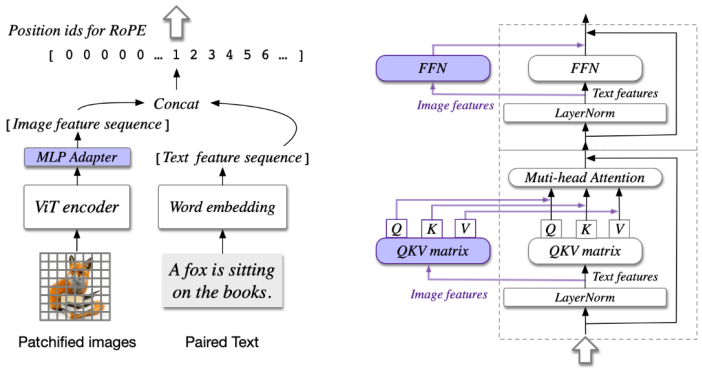


Frozen image encoders of most previous MLLMs could only work with 224×224 pictures. To work with larger pictures, the authors suggest feeding into SPHINX a series of patches + downsampled version of a picture. For each of them, there will be its own input to the final LLM, so larger picture means that we have a longer string supplied to the LLM.

5 CogVLM

Introduced in CogVLM: visual expert for large language models

The previous architectural approaches can be characterized as shallow alignment: they map image features into the input space of an LLM. CogVLM introduces this information deeper into the LLM. They compare the previous approaches to prompt tuning and their own to LoRA (with LoRA being a more efficient fine tuning mechanism).



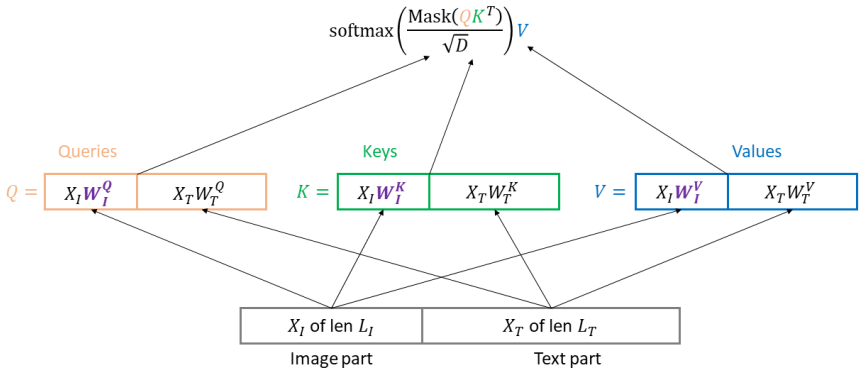
(a) The input of visual language model (b) The visual expert built on the language model

Figure 3: The architecture of CogVLM. (a) The illustration about the input, where an image is processed by a pretrained ViT and mapped into the same space as the text features. (b) The Transformer block in the language model. The image features have a different QKV matrix and FFN. Only the purple parts are trainable.

There are two trainable entities here:

- MLP adapter that projects ViT outputs to the space of text embeddings. It is a two-layer MLP.
- Visual expert module inside every transformer block. It works as follows. Imagine that the LLM receives a sequence with L_I tokens coming from an image and L_T tokens coming from the text prompt.

  In self-attention we use trainable matrices W_I^Q, W_I^K, W_I^V to produce queries, keys and values for the image part and frozen matrices W_T^Q, W_T^K, W_T^V for the text part.



Text and visual queries, keys and values are further concatenated and used as in usual attention mechanism. The Mask thing stands for the triangular mask not allowing the model to "look into the past".

The FFN layer is done separately for image and text tokens:

$$\text{FFN}(X) = \text{concat}\left(\text{FFN}(X_I), \text{FFN}(X_T)\right)$$

With this, the authors achieve SoTA quality on many tasks:

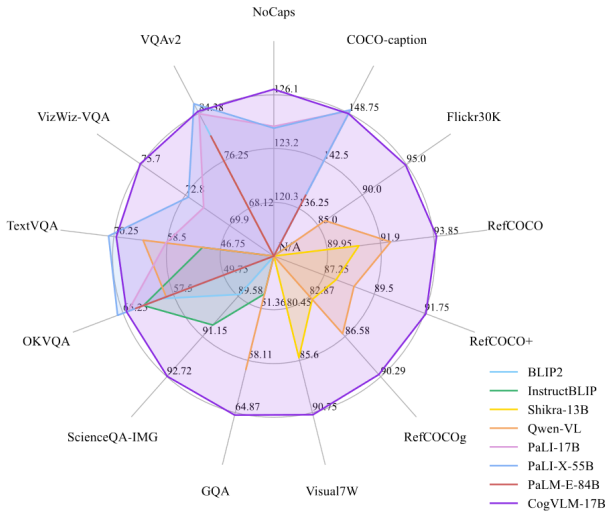


Figure 1: The performance of CogVLM on a broad range of multi-modal tasks compared with existing models.

6 Appendix. Multimodality through orchestration

Though LLMs only produce texts, these texts can contain plans for execution of complex pipelines. And a natural idea for working with different modalities is to make from LLM an orchestrator of modality-specific tools.

This approach drew considerable interest in early 2023. I won't delve deep into it, just share two interesting examples:

1. Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models

   The heart of this system is a prompt manager that transforms visual signals into text and loads some general principles into ChatGPT which creates queries to particular vision foundational models.

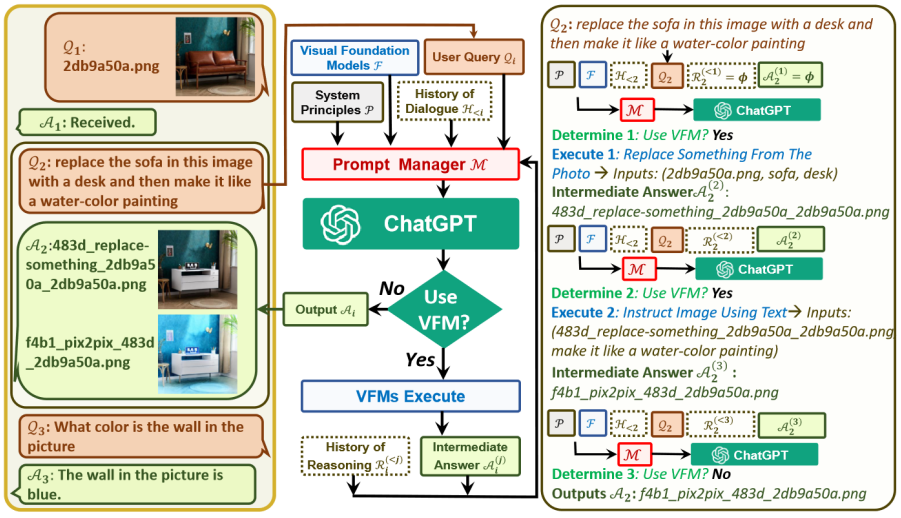
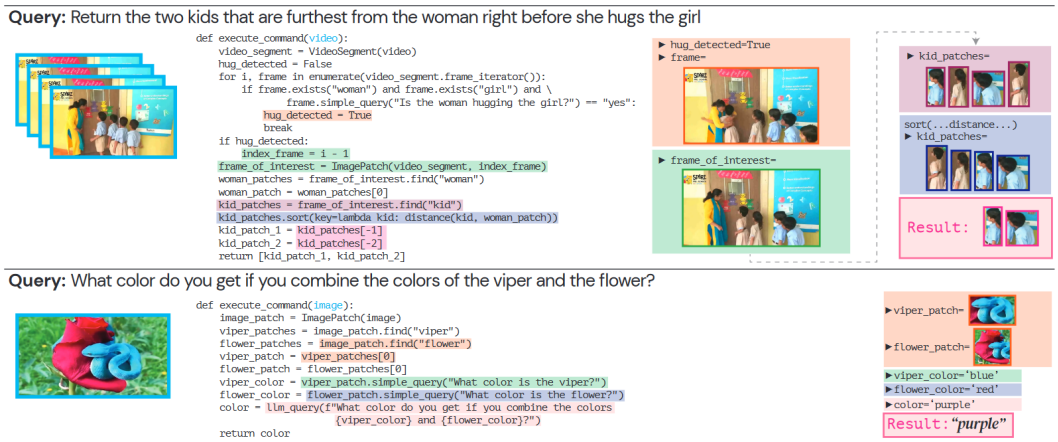


Figure 2. Overview of Visual ChatGPT. The left side shows a three-round dialogue, The middle side shows the flowchart of how Visual ChatGPT iteratively invokes Visual Foundation Models and provide answers. The right side shows the detailed process of the second QA.

2. ViperGPT: Visual Inference via Python Execution for Reasoning

   Given a query, this model outputs Python code that would process the visual data. It leverages Codex code-writing LLM and uses a specfic system prompt to describe the task and API details for Codex.



3. It's curious to note that the idea of orchestration by LLM returns in more up-to-date MLLMs. For example, in NExT-GPT: Any-to-Any Multimodal LLM LLM decides what should be the modality of the answer: text, image, video, or audio.