

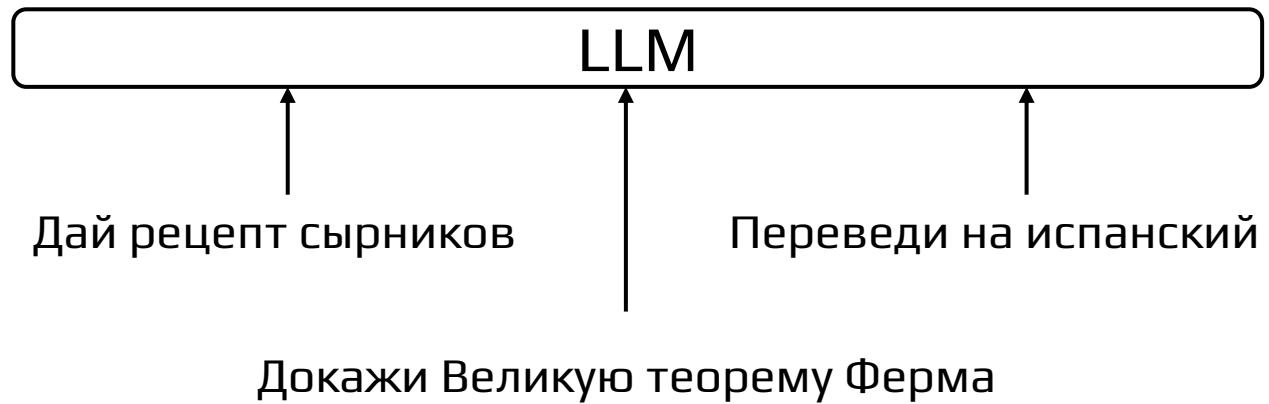


# Основы визуально-языковых моделей

Курс "Мультимодальные модели"

## Предпосылки

- LLM стали универсальными моделями для текстовых задач
- Хотелось бы иметь универсальную модель и для других модальностей
- Первый шаг — поддерживать обработку изображений и видео на вход



# Визуально-языковые модели

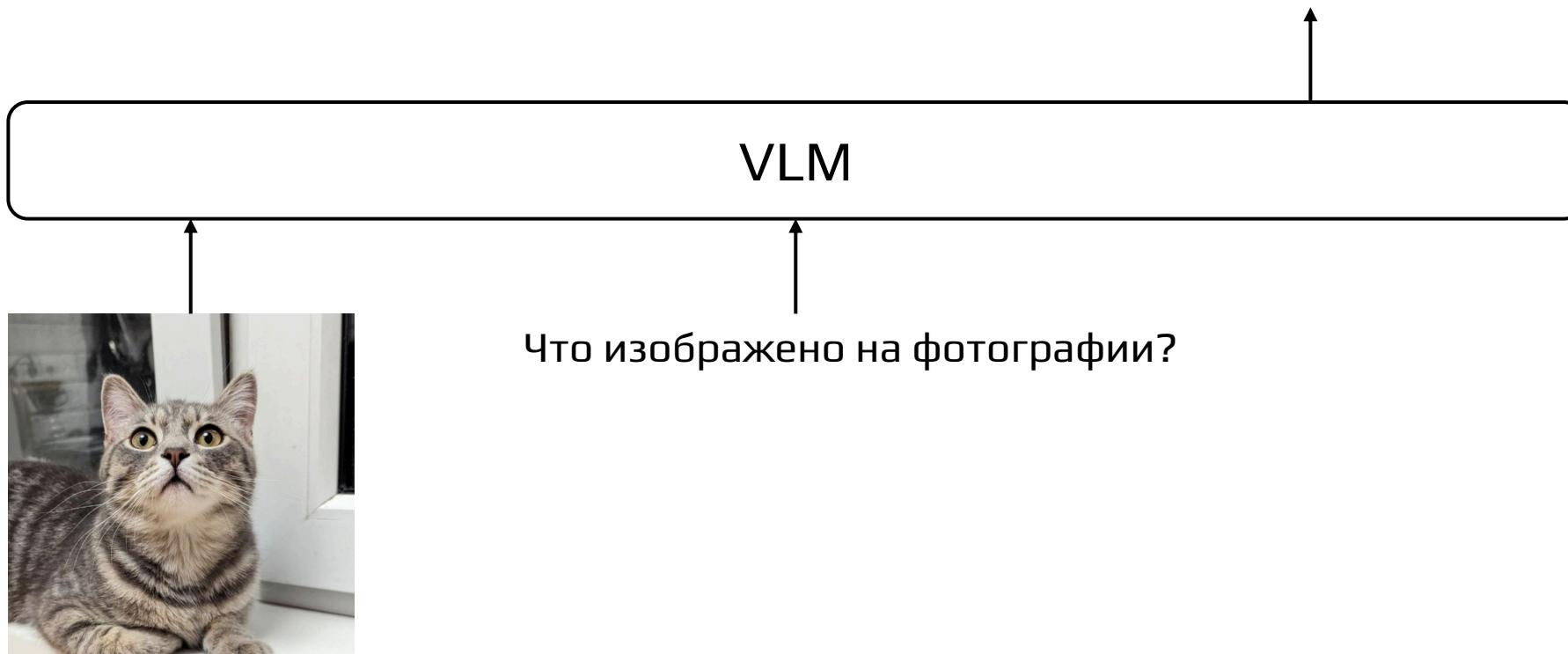
**Визуально-языковая модель – LLM, умеющая работать с визуальной модальностью.**

(a.k.a. Vision-Language Model, **VLM**)

Примеры:

- Проприетарные: GPT-4o, Claude 4 Sonnet, Gemini 2.5 Pro
- Open-source: LLaVA, Qwen-VL, InternVL

Кот, сидящий на подоконнике

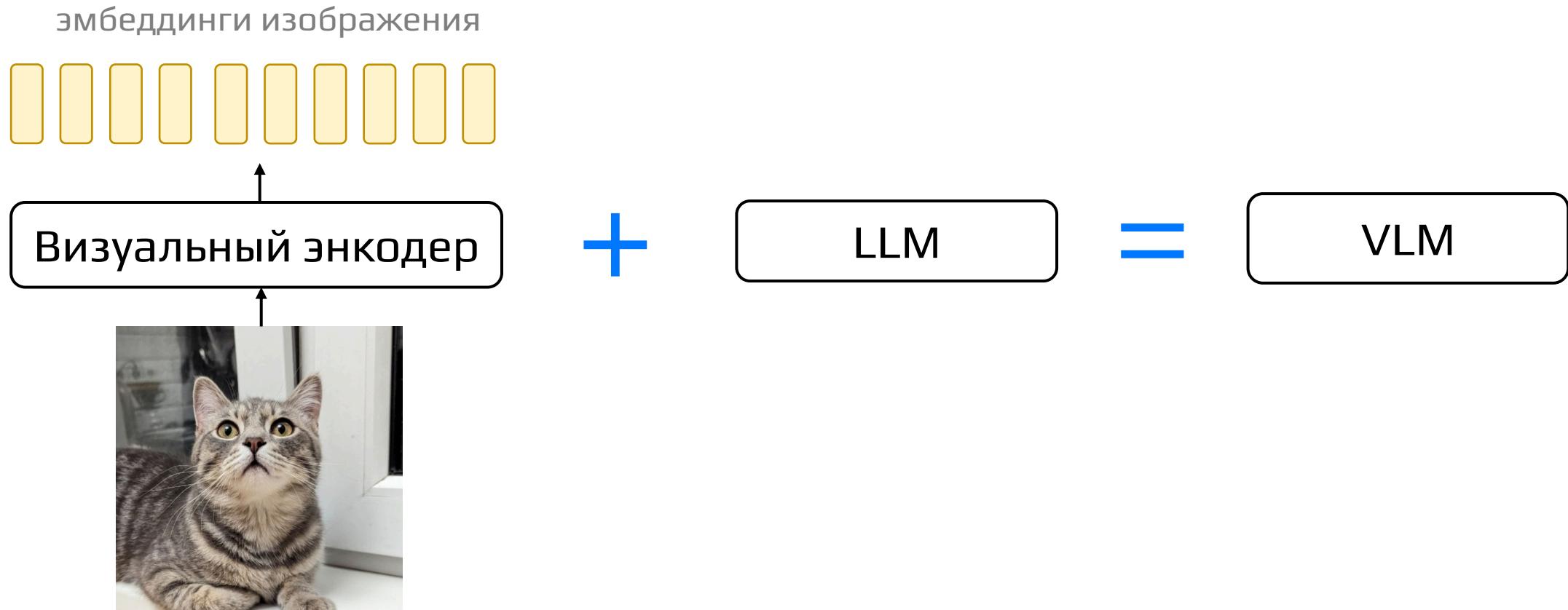


# 1. Общая архитектура современных VLM



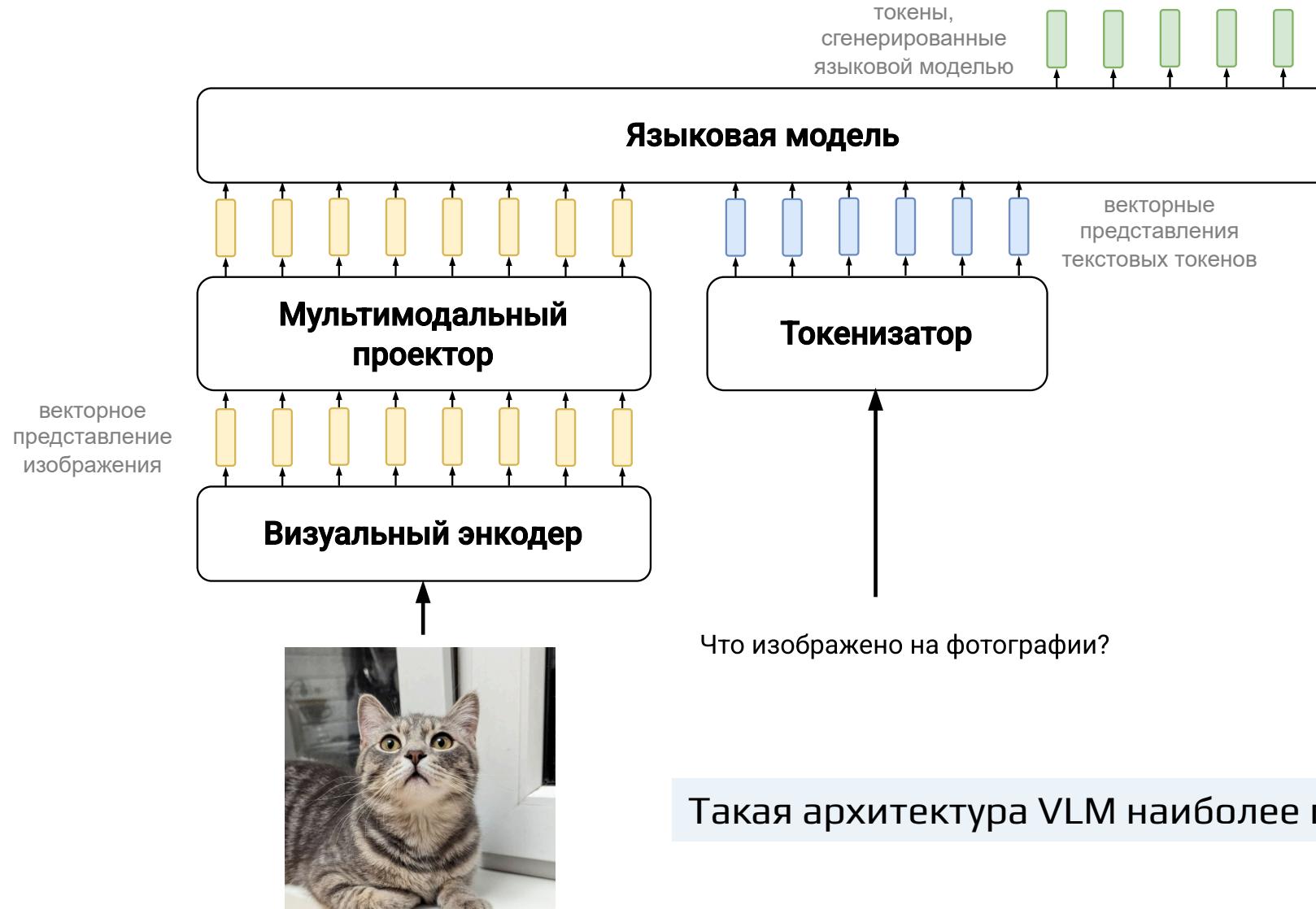
# Как строить VLM?

**Идея:** использовать предобученные визуальный энкодер и LLM  
(объединить унимодальные модели в мультимодальную)



# LLaVA (2023)

Кот, сидящий на подоконнике



# Получение визуальных эмбеддингов

**Идея:** использовать не один эмбеддинг изображения, а сразу несколько

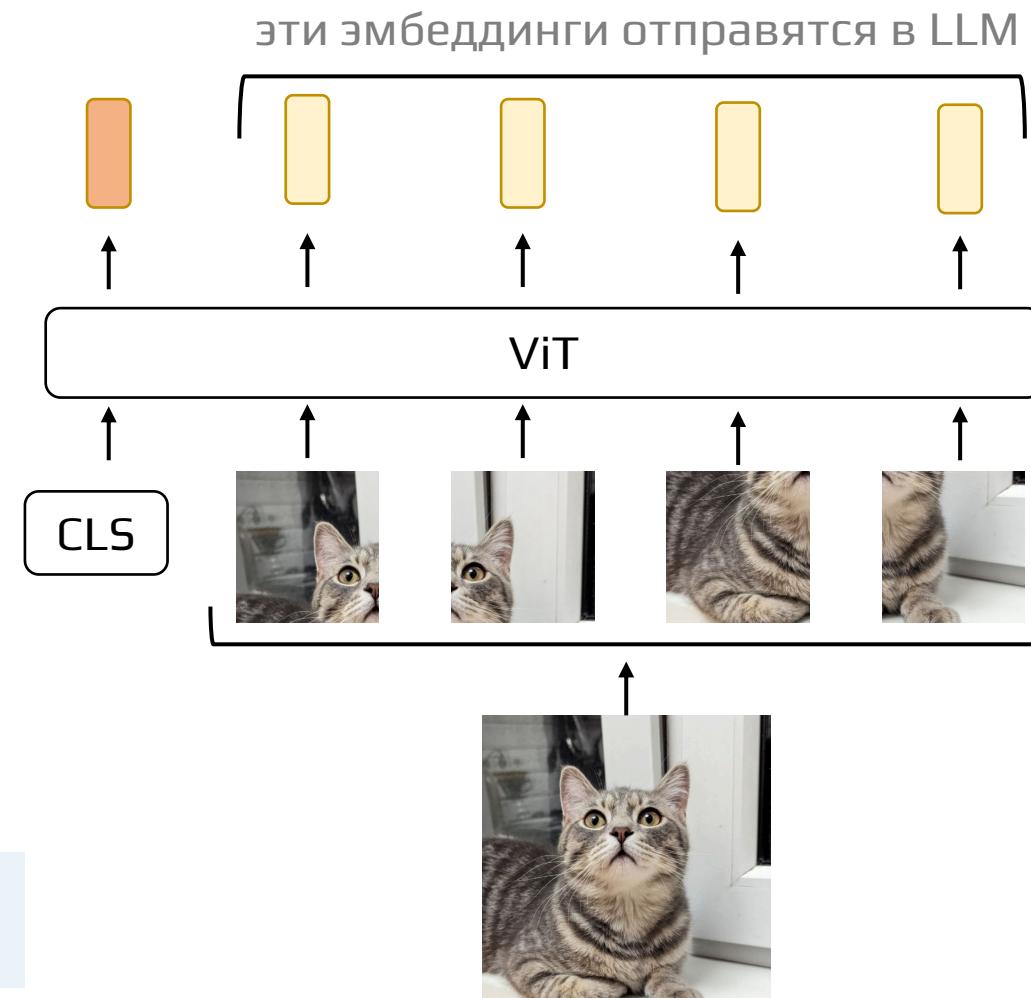
## Мотивация:

- 1) передать LLM как можно больше информации об изображении;
- 2) дать LLM пространственное понимание изображения.

На примере архитектуры энкодера ViT (Vision Transformer):

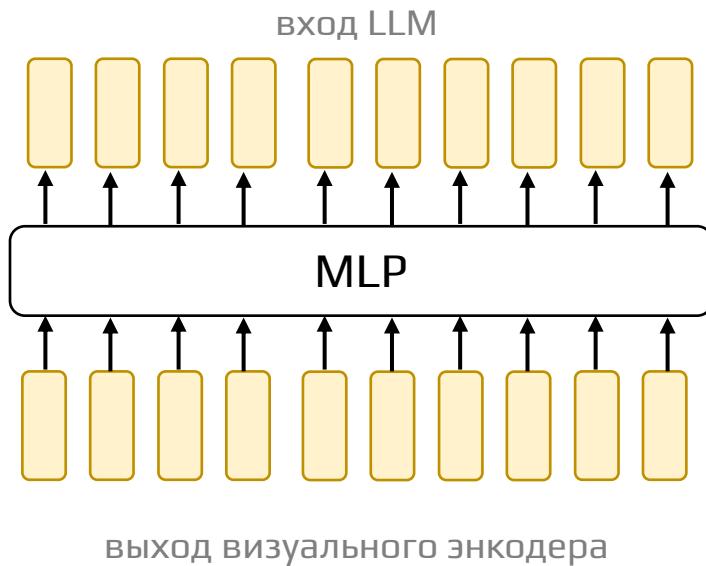
Во многих VLM используют ViT, обученный на задаче CLIP

В базовом варианте входное изображение просто resize-ится ко входному разрешению энкодера

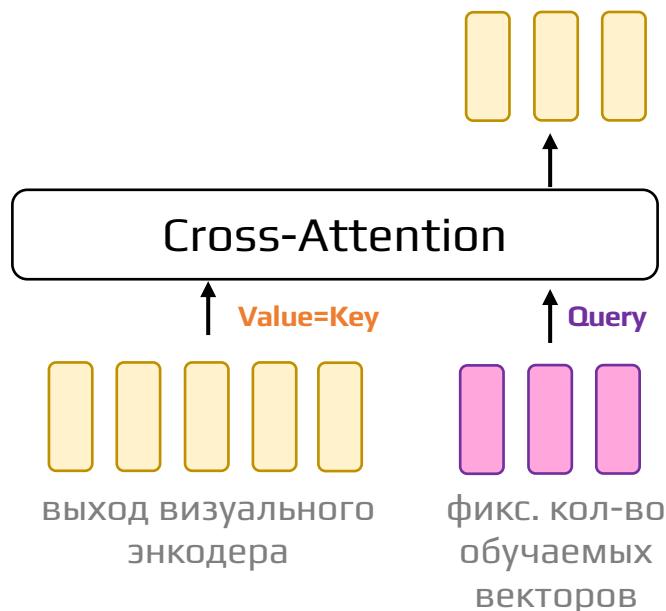


# Мультимодальный проектор

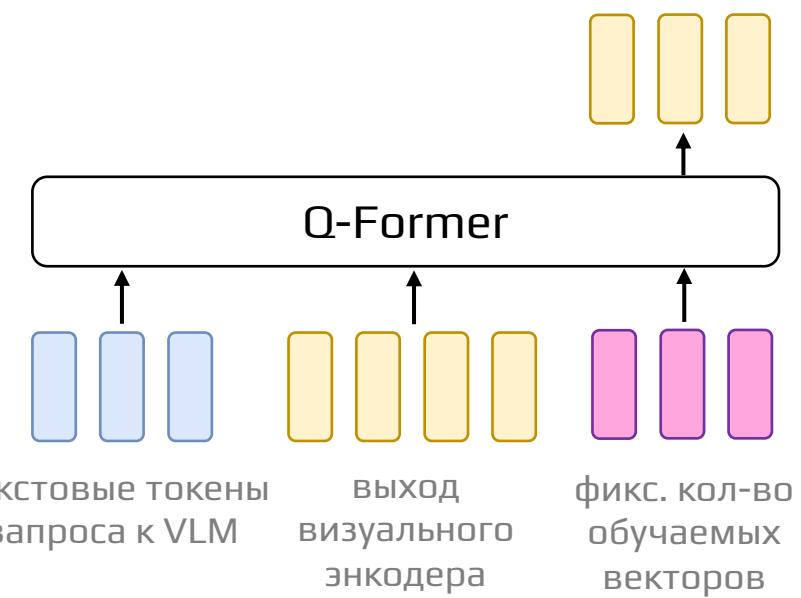
⭐ **MLP** (1-2 слоя)  
(LLaVA, Qwen2-VL, InternVL)



**Cross-Attention**  
(Flamingo, Qwen-VL)



**Q-Former**  
(InstructBLIP)



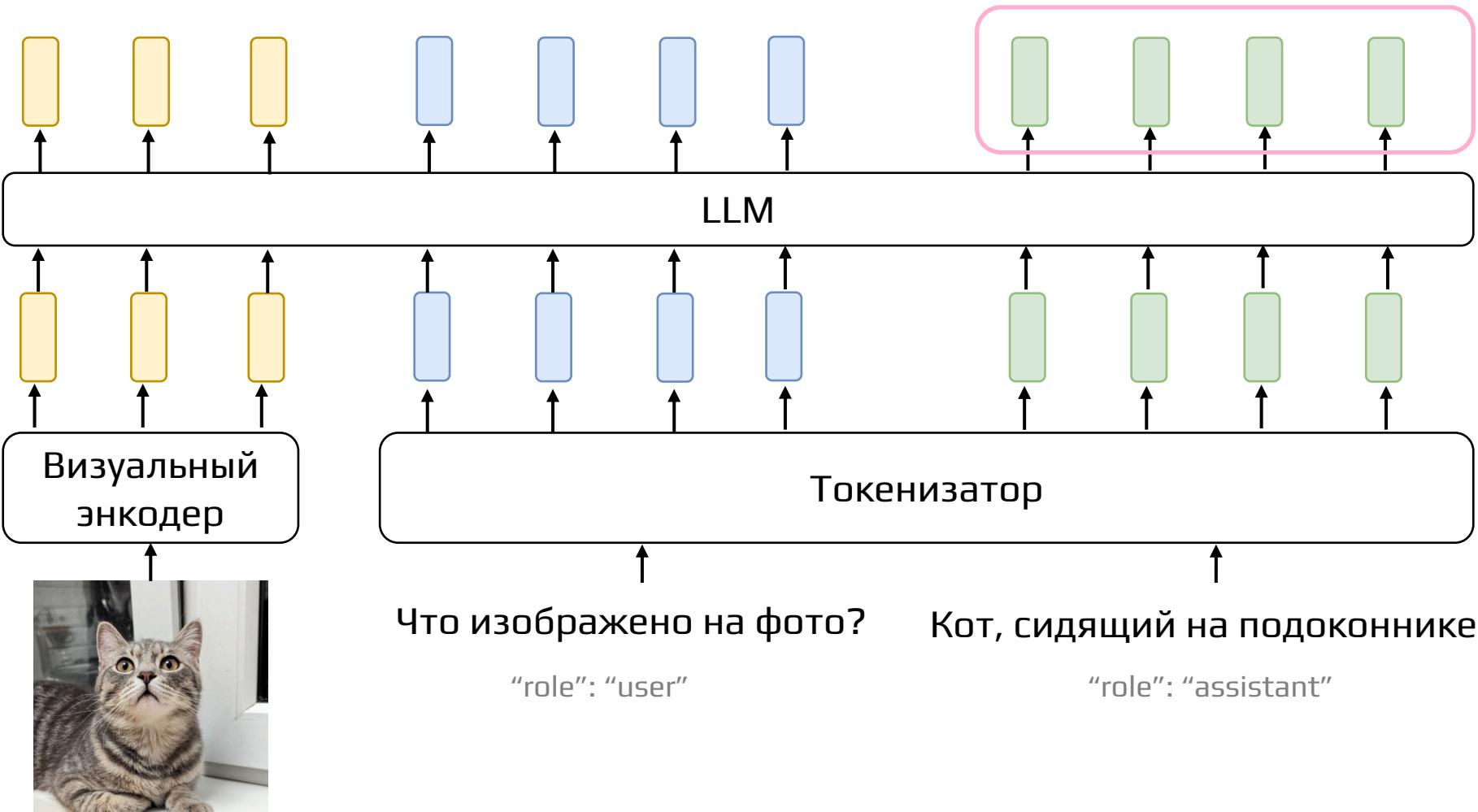
Извлечение «наиболее важных»  
визуальных признаков

Извлечение визуальных  
признаков, «наиболее важных»  
для конкретного запроса

## Обучение VLM. Функция потерь

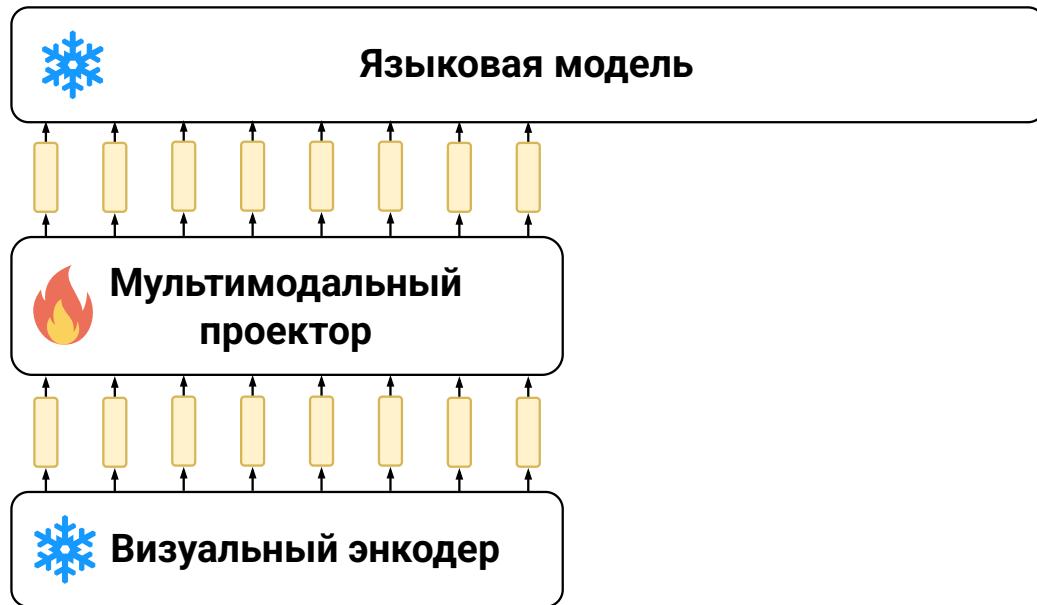
$$CE(y_{\text{target}} \parallel y_{\text{pred}})$$

По этим токенам считается  
функция потерь



# Обучение VLM. LLaVA-v1

## Предобучение



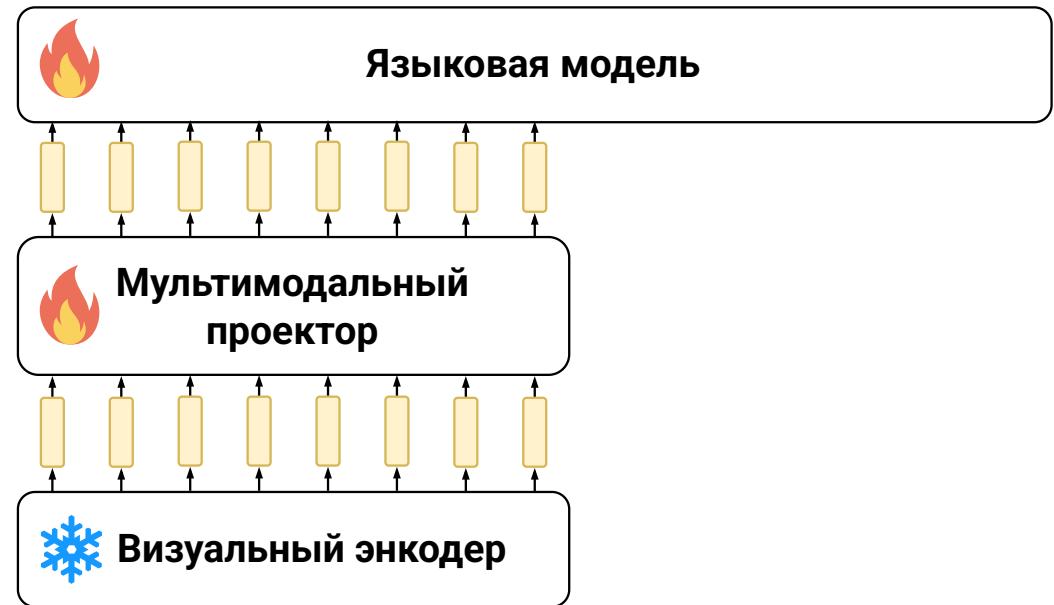
простые данные:

вход:



выход: Кошка сидит на задних лапах.

## Дообучение на инструкциях



сложные instruction-following данные:

вход:



Какой породы кошка?

выход: Кошка породы Девон-рекс.

## Обучение VLM. Модификации

- Разморозка визуального энкодера
- RLHF

1. Общая архитектура современных VLM
2. Вариации архитектуры VLM:
  - а) обучение без разморозки LLM

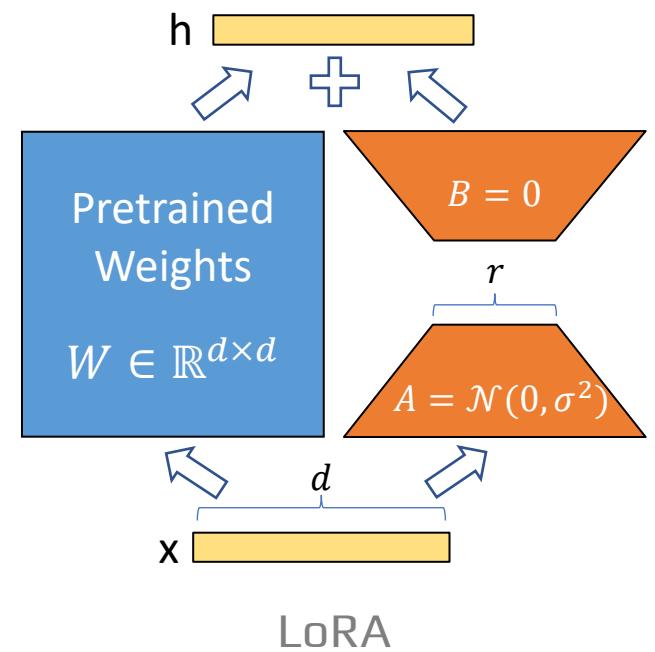
# Обучение VLM без разморозки LLM

## Мотивация:

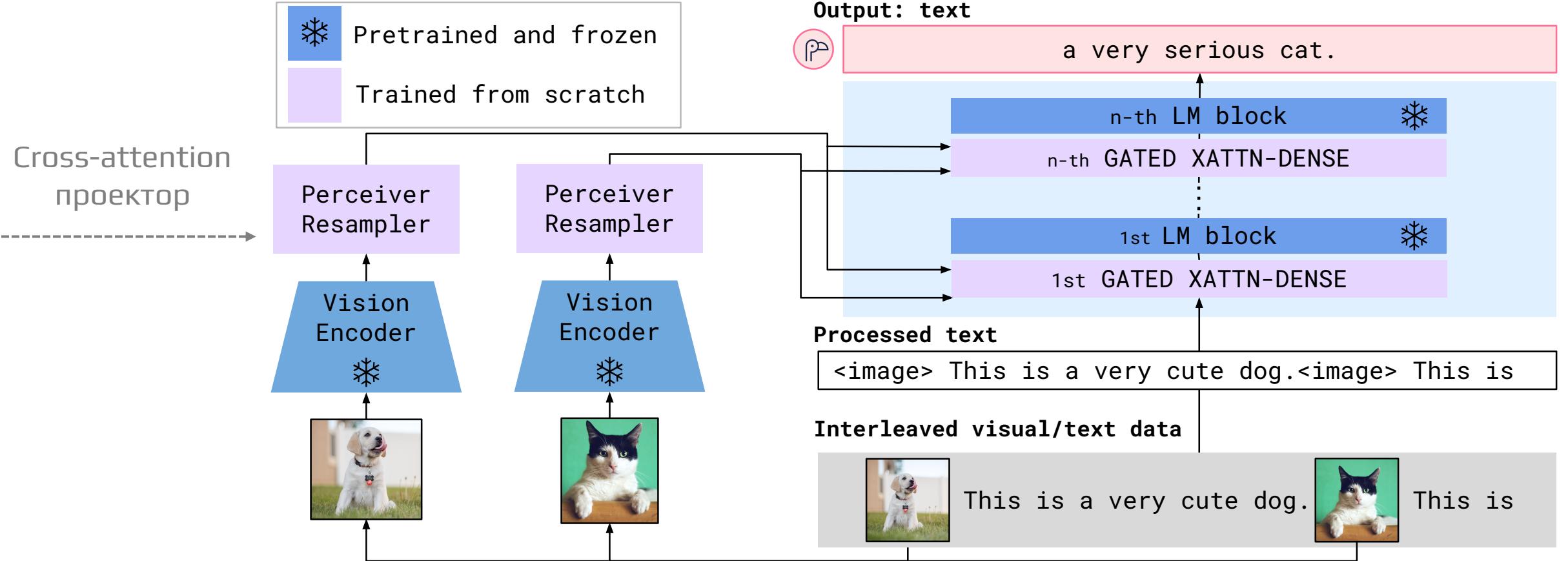
- Дешевле/быстрее обучение
- LLM не переобучается под наши данные:
  - a) VLM будет работать на text-only задачах так же хорошо
  - b) выше обобщающая способность в целом

## Варианты:

- Обучение адаптеров для LLM (например, LoRA: Low-Rank Adaptation)
- Использование LLM в качестве blackbox

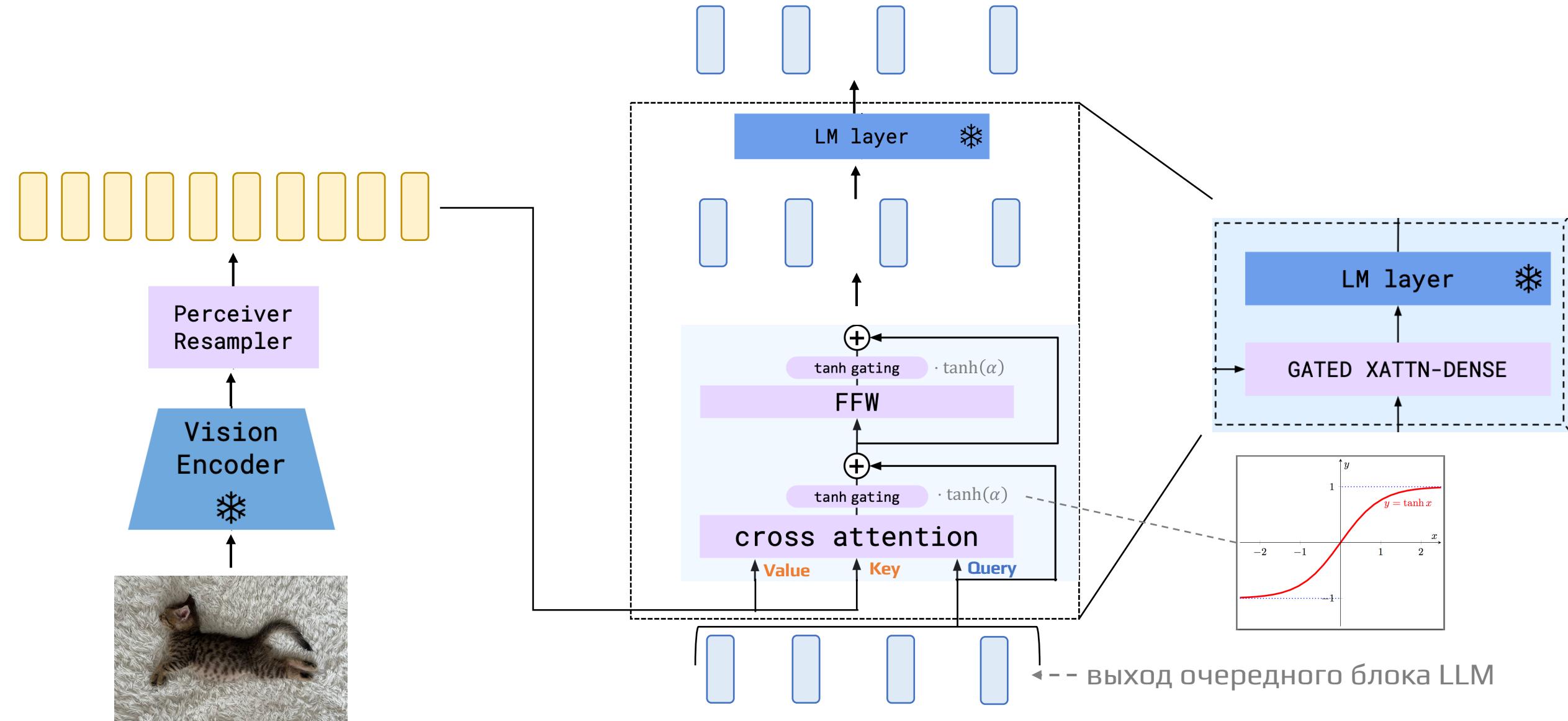


# Flamingo (2022). Общая архитектура



Deep Fusion визуальных признаков, в отличие от Early Fusion в LLaVA

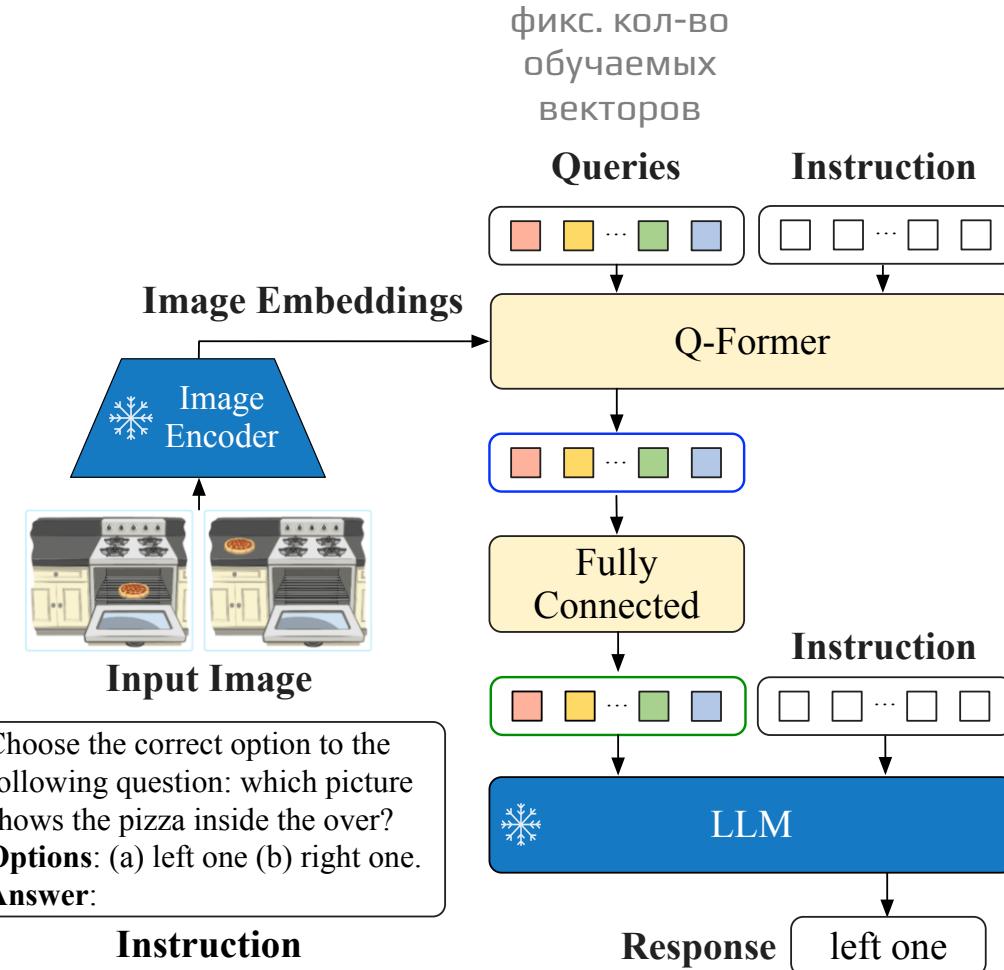
# Flamingo. Подача визуальных эмбеддингов в LLM



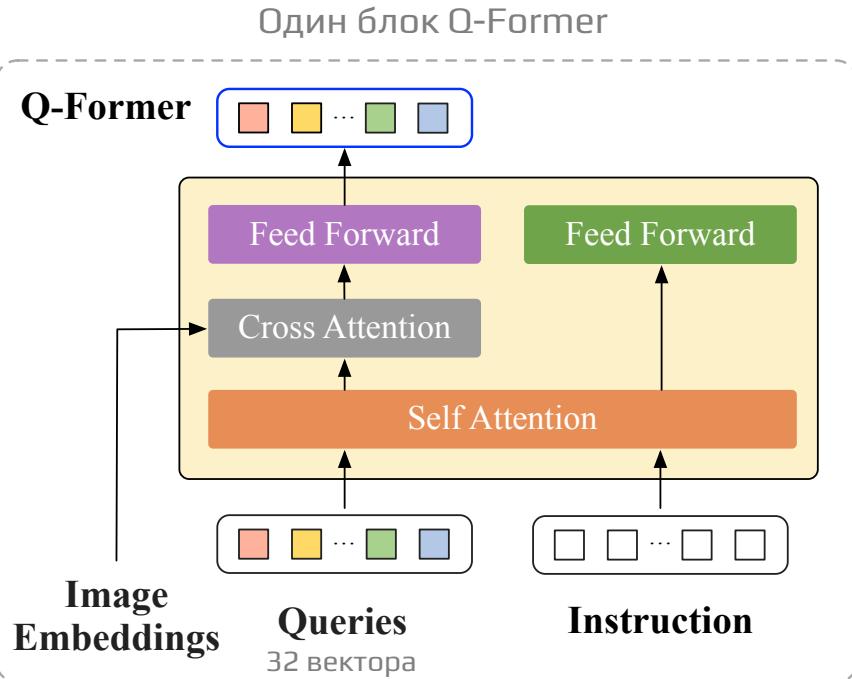
## Flamingo. Сравнение с полной разморозкой LLM

- Обучаемых параметров примерно столько же — экономии по памяти/времени для обучения почти нет
- Прямой проход (forward) всей VLM дольше — дольше обучение/инференс
- Веса исходной LLM не меняются — знания сохраняются

# InstructBLIP (2023). Общая архитектура

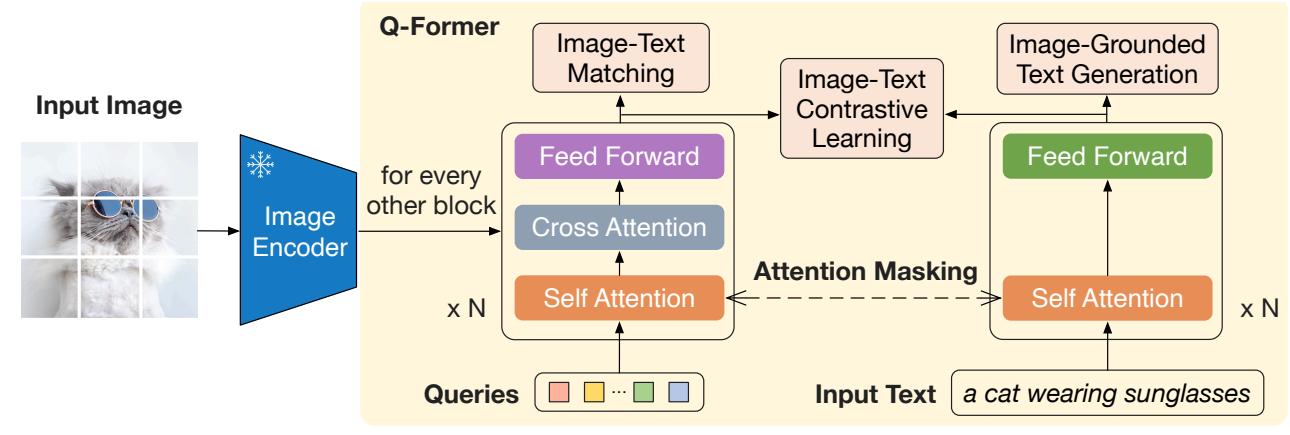


# InstructBLIP. Q-Former



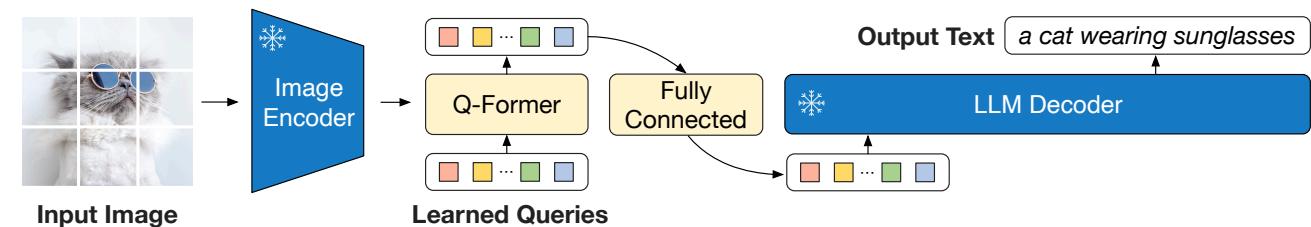
Self Attention и Feed Forward  
инициализируются весами BERT

## 1-й этап обучения:

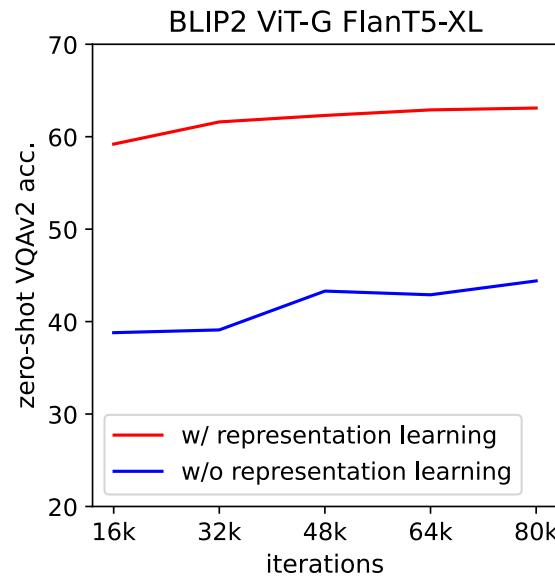
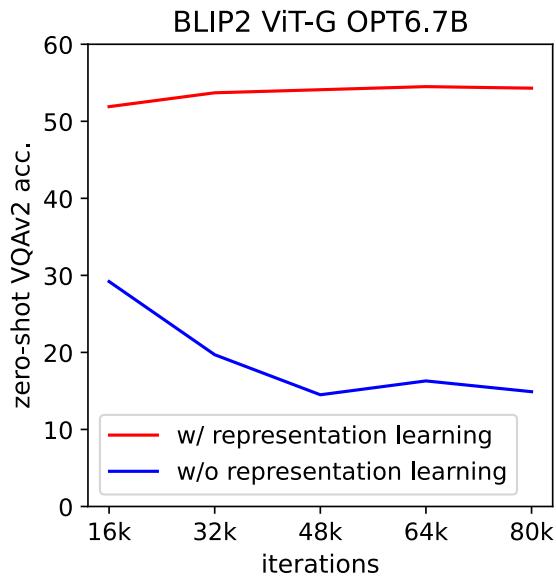


Под каждую из задач используют разные  
маски во время Self Attention

## 2-й этап обучения:



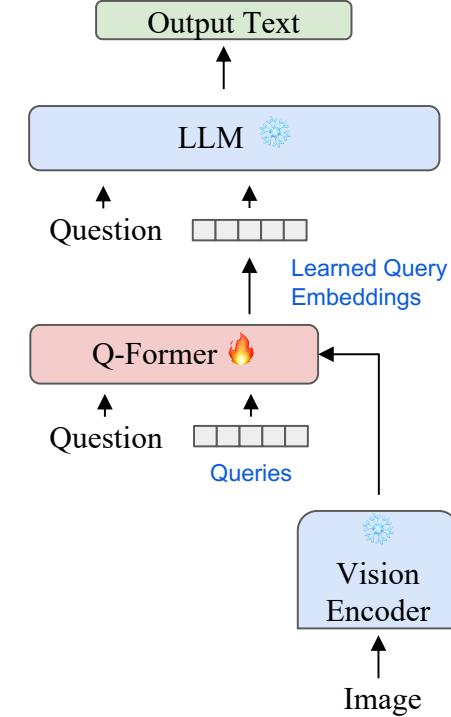
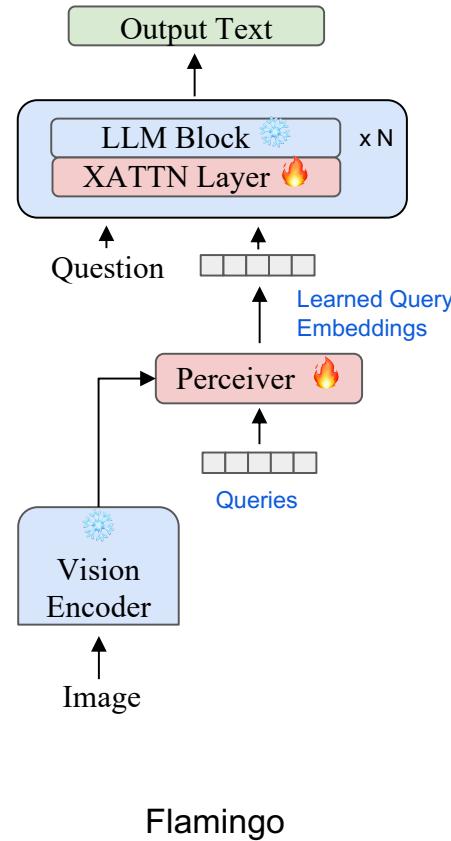
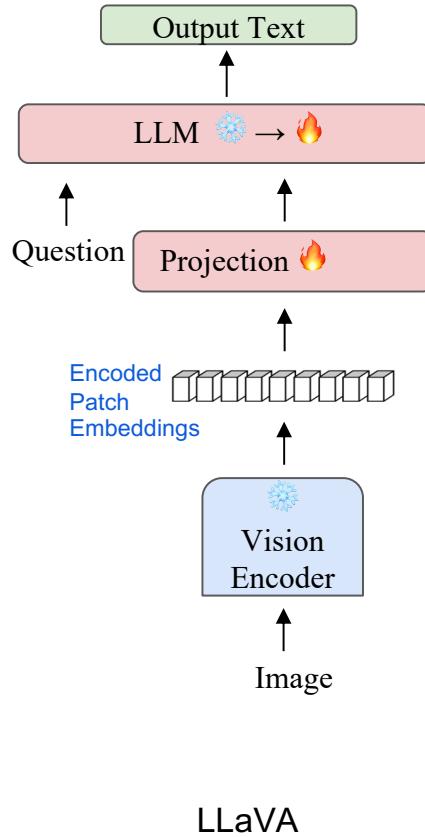
## InstructBLIP. А нужен ли 1-й этап?



**Note:** Графики для BLIP2, который отличался от InstructBLIP

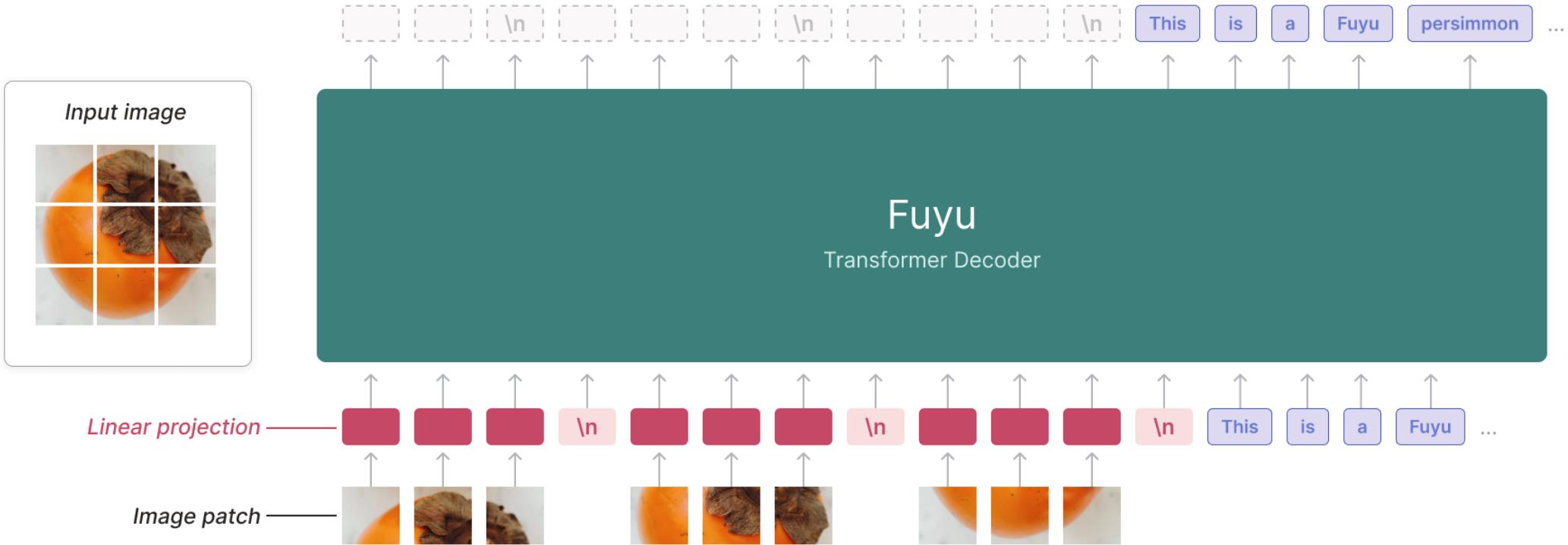
Предварительное отдельное обучение Q-Former улучшает общее качество VLM

# Recap



1. Общая архитектура современных VLM
2. Вариации архитектуры VLM:
  - а) обучение без разморозки LLM
  - б) VLM без визуального энкодера

# Fuyu (2023).



Подаём патчи изображений не в ViT, а сразу в LLM

## Fuji (2023). Подача изображения напрямую в LLM

### **Плюсы:**

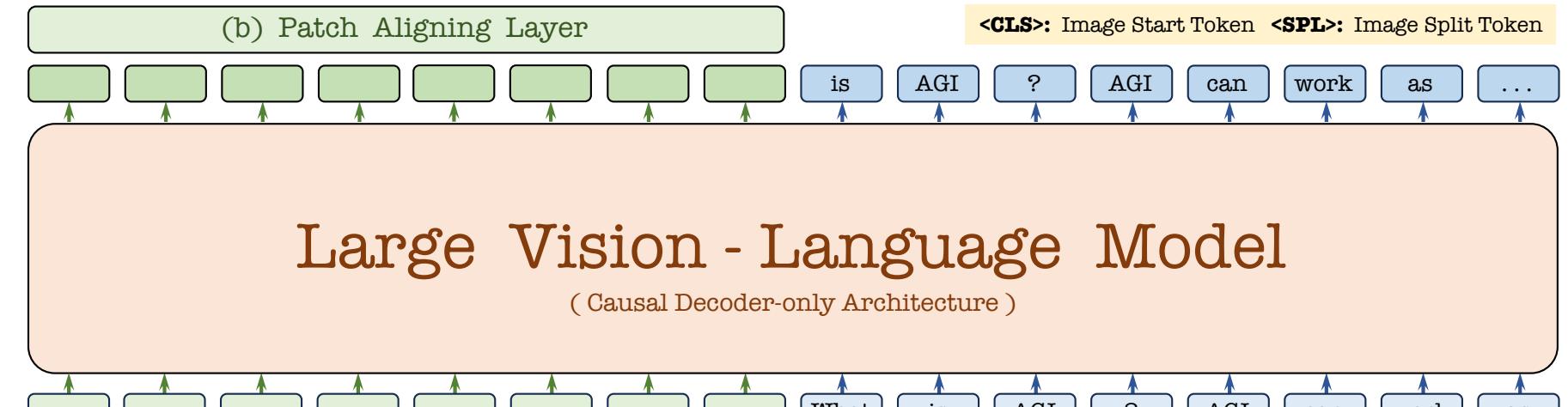
- «Нативно» поддерживается более детальная обработка high-res изображений
- Не нужно выбирать энкодер
- Не нужно выбирать размораживаемые части

### **Минусы:**

- Качество хуже, чем у других VLM
- LLM приходится стать визуальным энкодером → забывание
- Энкодер легче, его можно учить на большем объёме данных → он будет лучше кодировать изображения, чем LLM

# EVE (2024).

На 1-м этапе обучения дистиллируют предобученный энкодер



свёртка +  
attention +  
линейный слой

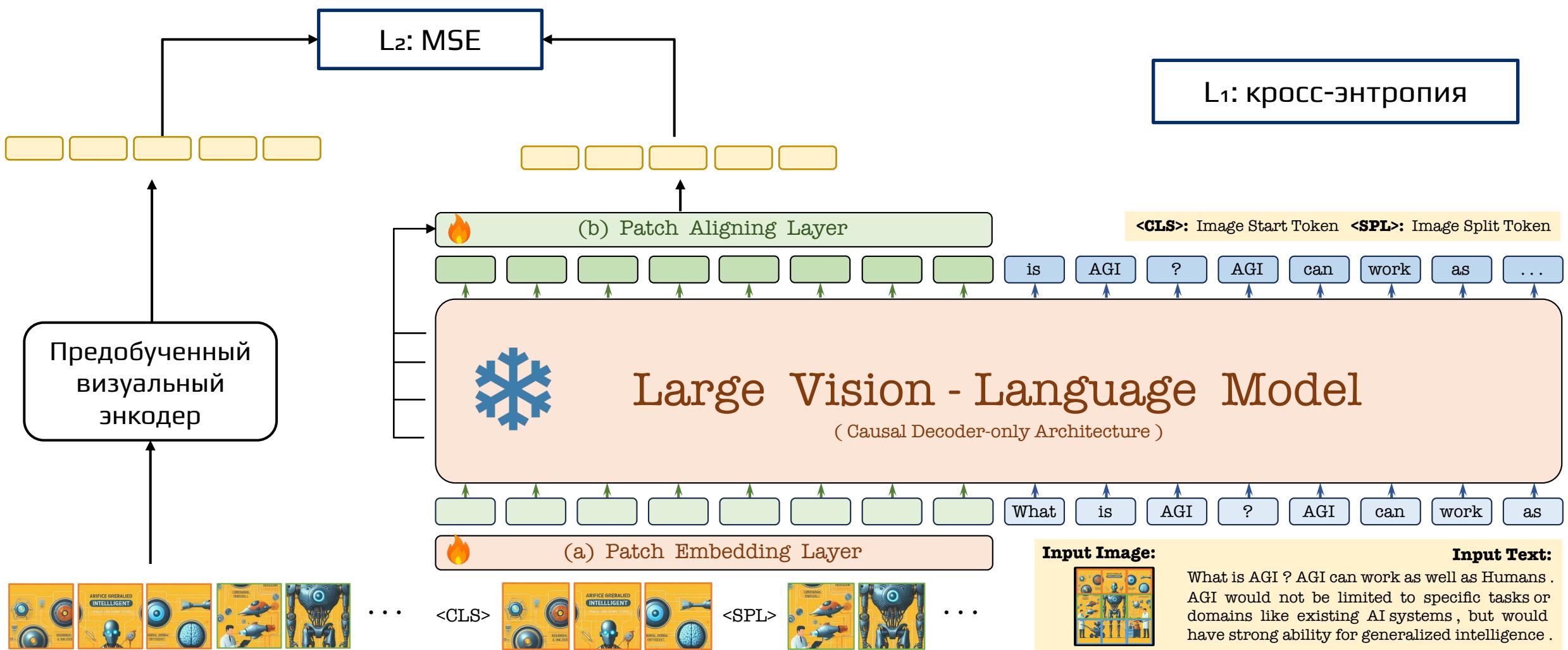
Обучают небольшой кодирующий слой

**Input Image:**  
What is AGI ? AGI can work as well as Humans .  
AGI would not be limited to specific tasks or  
domains like existing AI systems , but would  
have strong ability for generalized intelligence .



**Input Text:**

# EVE. 1-й этап обучения

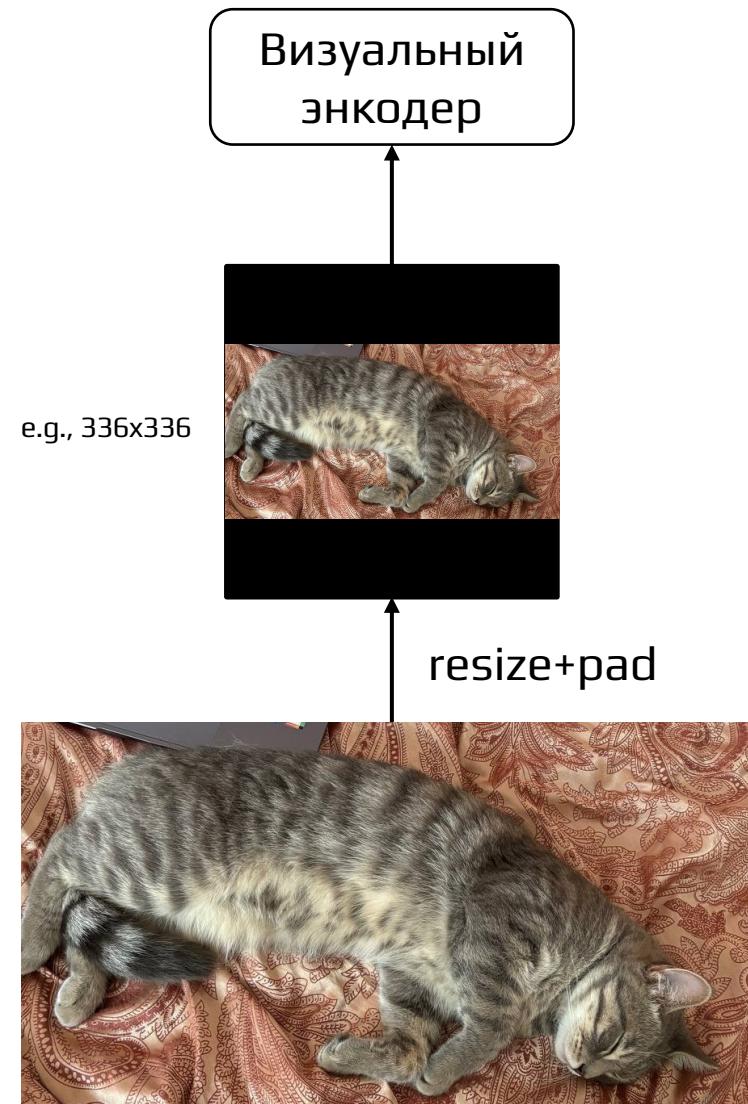


1. Общая архитектура современных VLM
2. Вариации архитектуры VLM:
  - а) обучение без разморозки LLM
  - б) VLM без визуального энкодера
3. Обработка изображений
  - а) high-res

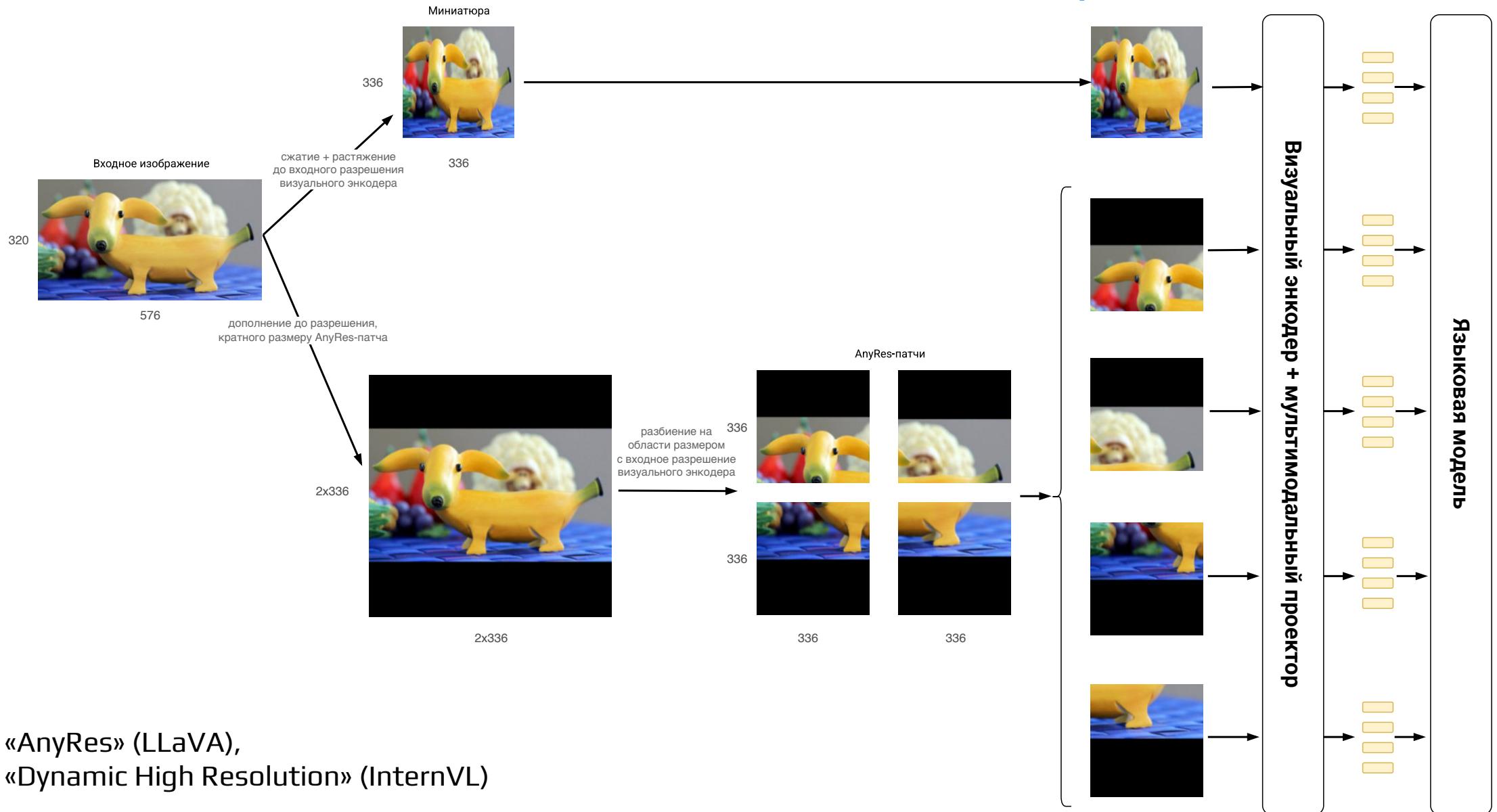
# Обработка high-res изображений

## Проблема базового способа:

- Теряется визуальная информация, что особенно критично для детализированных изображений (в частности, в задаче OCR)

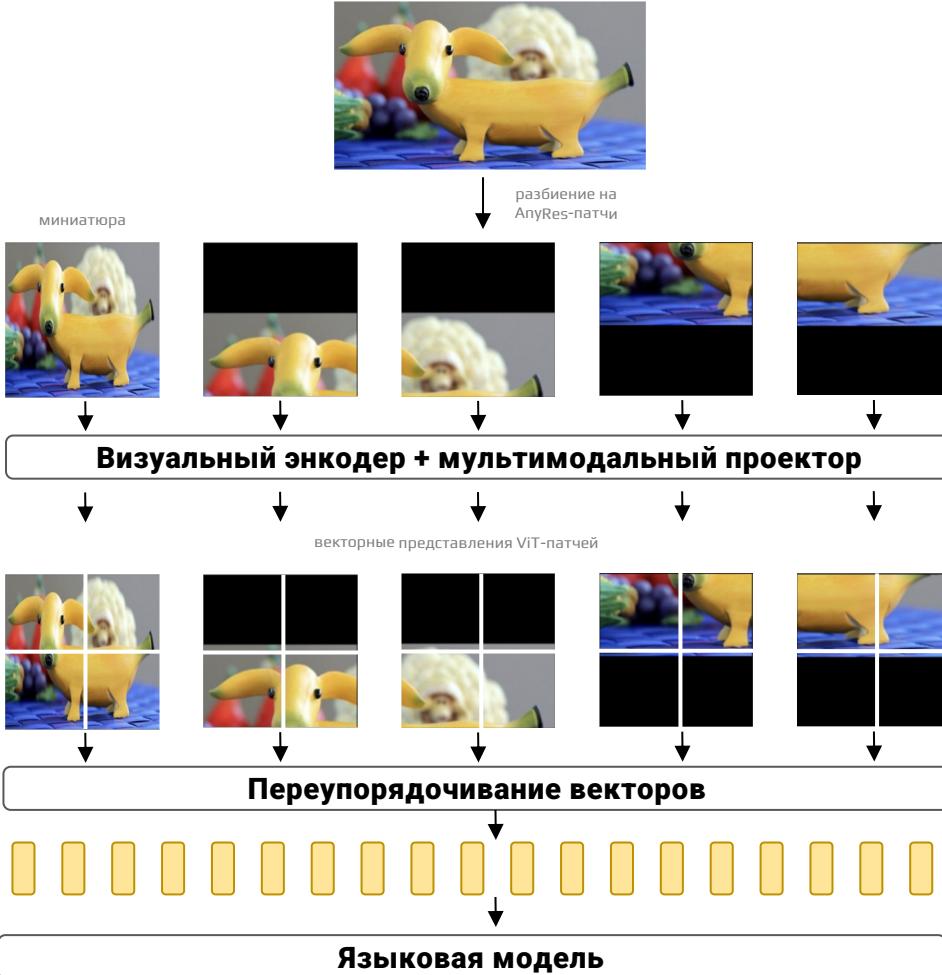


# Разбиение большого изображения на части (AnyRes)



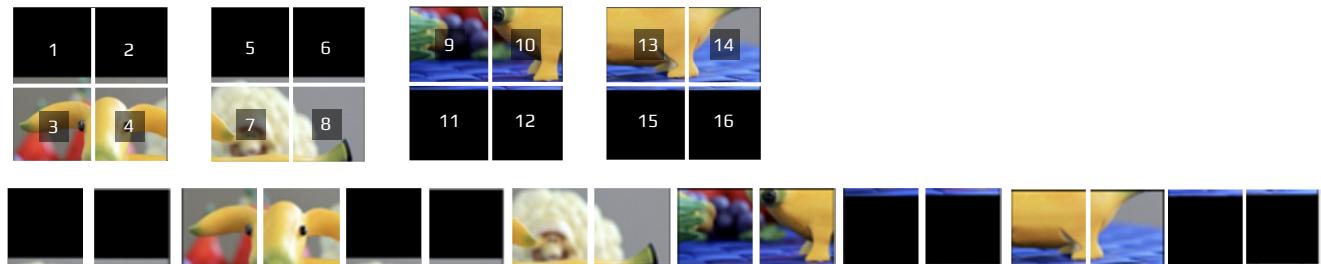
# AnyRes. Порядок визуальных векторов

Каждый именованный квадратик – ViT-патч, для которого энкодер выдаёт эмбеддинг

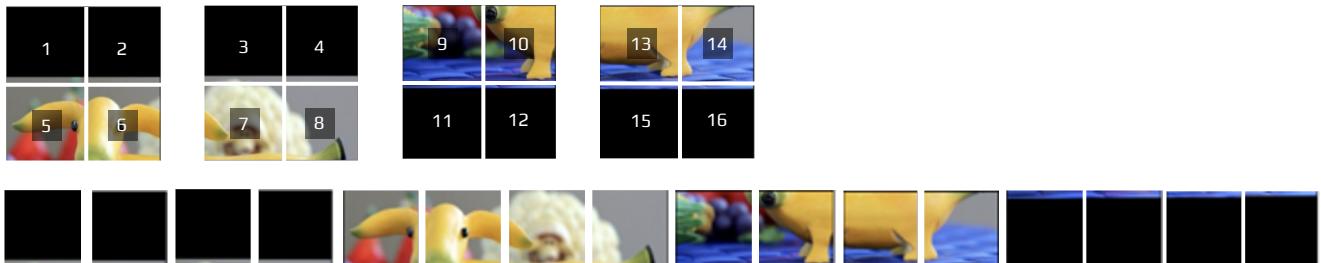


Эмбеддинги миниатюры подаём первыми

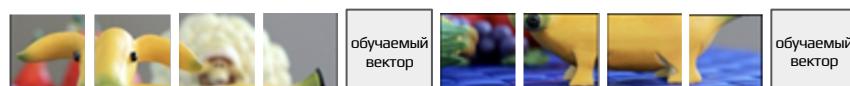
**flat:** как независимые изображения



**spatial:** как если бы это было одно большое изображение

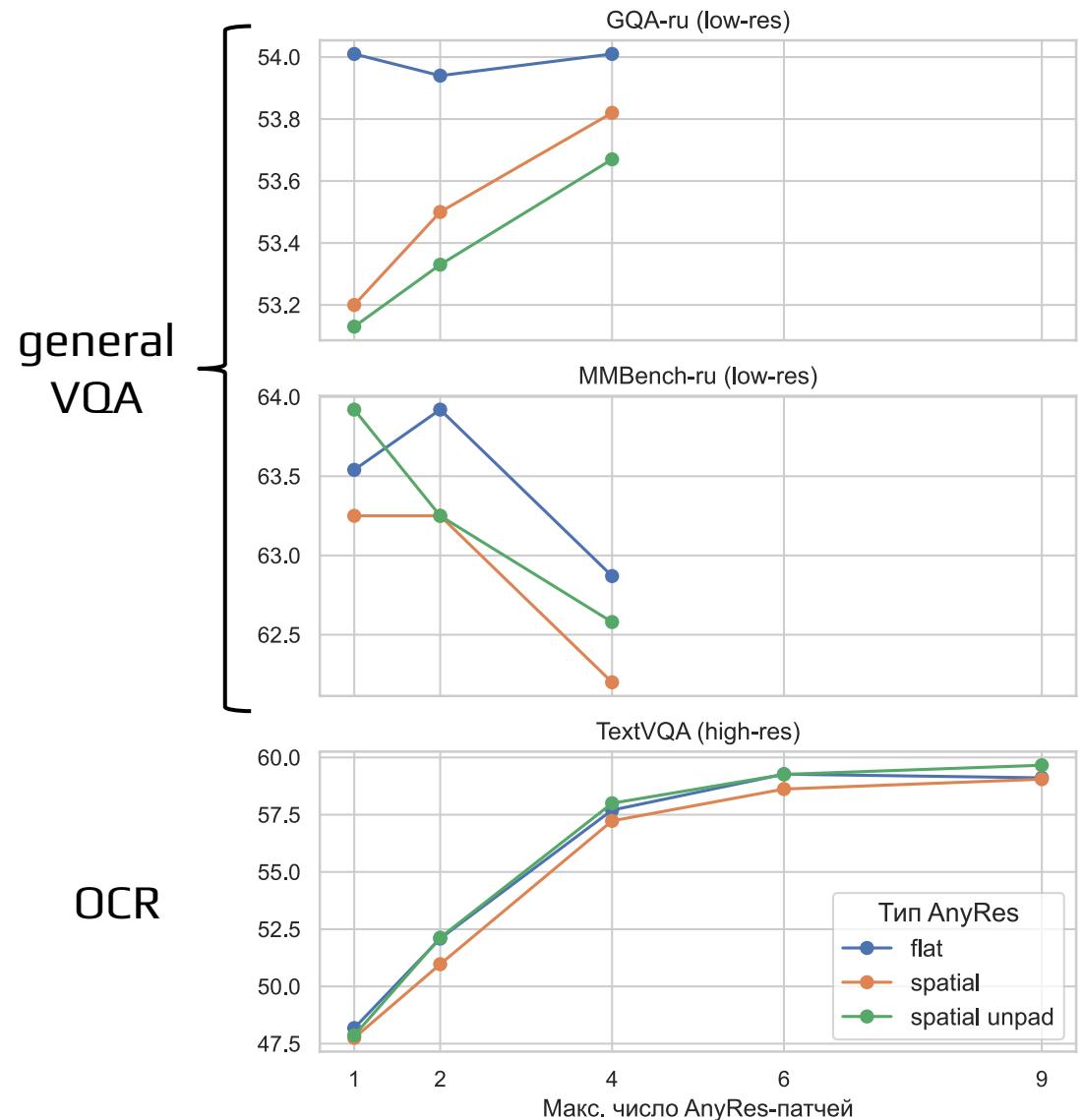


**spatial uprad:** spatial + удаляются патчи, состоящие из padding'a + добавляется обучаемый вектор



# AnyRes. Влияние на качество

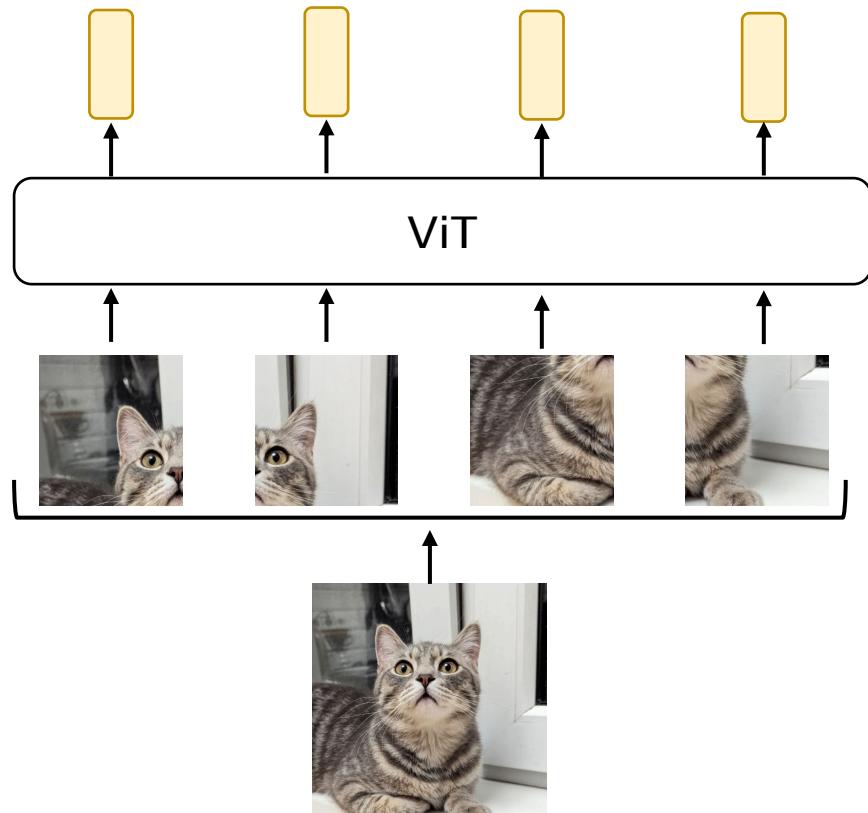
- Для требовательных к пониманию деталей задач (например, OCR) AnyRes значительно улучшает качество. Для более простых – эффект незначителен
- Разница между вариантами подачи эмбеддингов в LLM незначительна



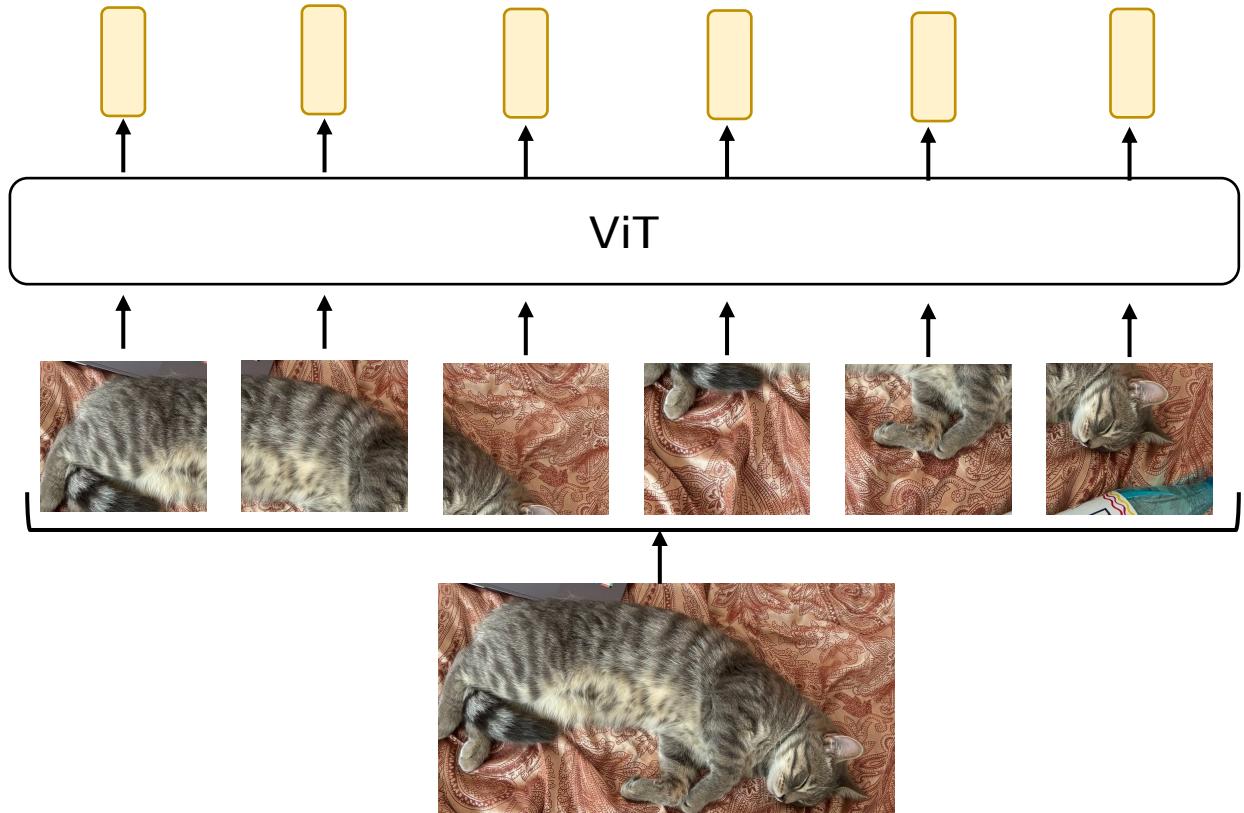
# Подача произвольного числа патчей в ViT ([Qwen2-VL, 2024](#))

## Предобученный ViT:

→ Обучался под конкретное разрешение  
(фикс. кол-во патчей)



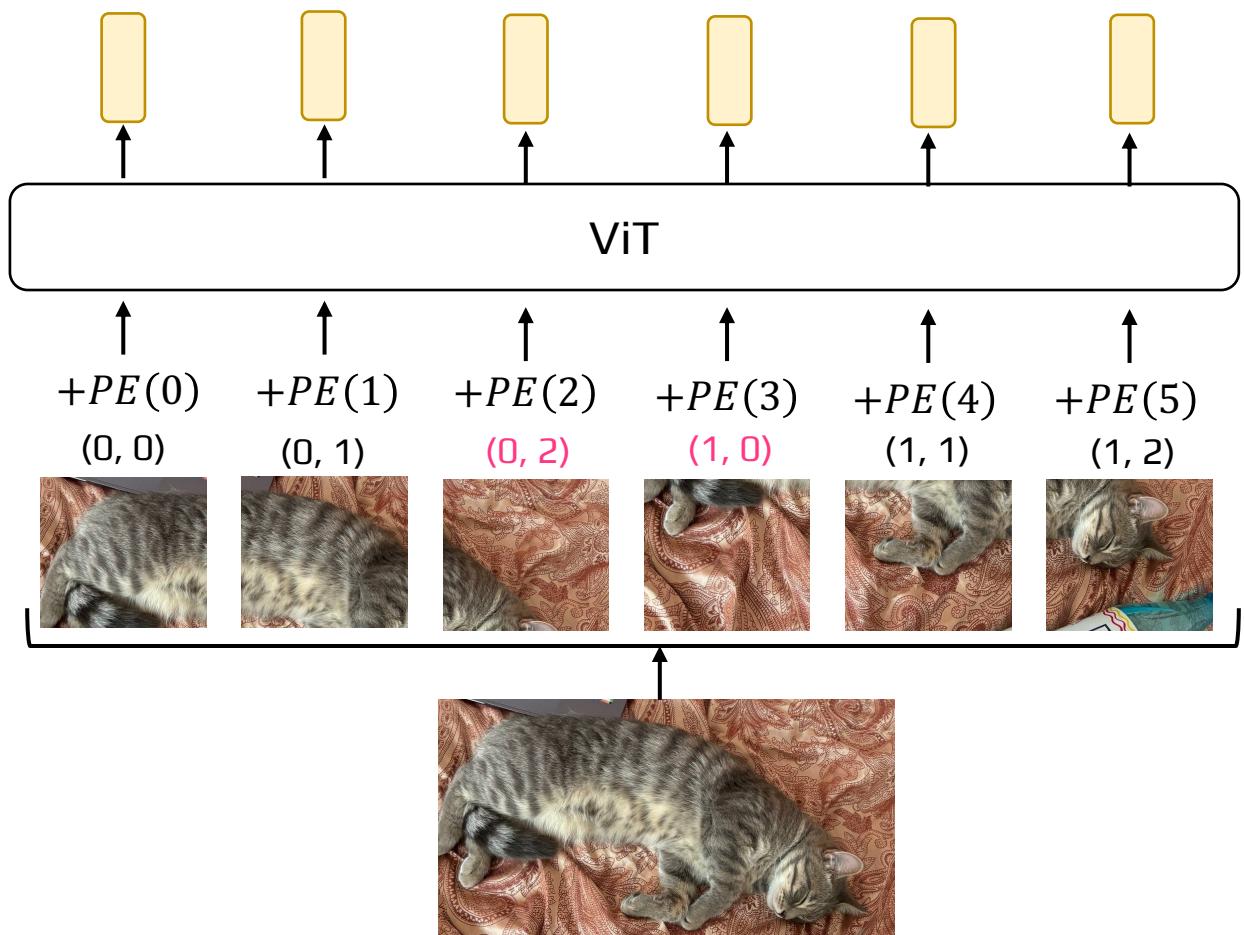
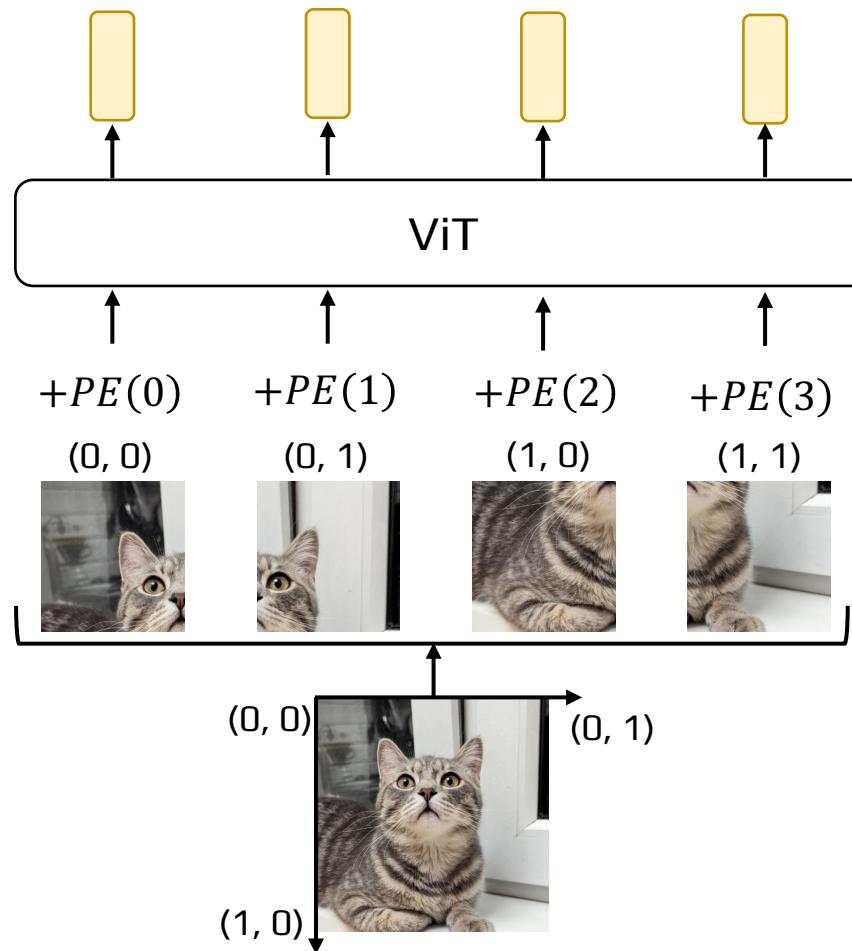
→ Но мы можем подавать в него  
произвольное кол-во патчей



# Подача произвольного числа патчей в ViT. Positional Encoding

## Проблема:

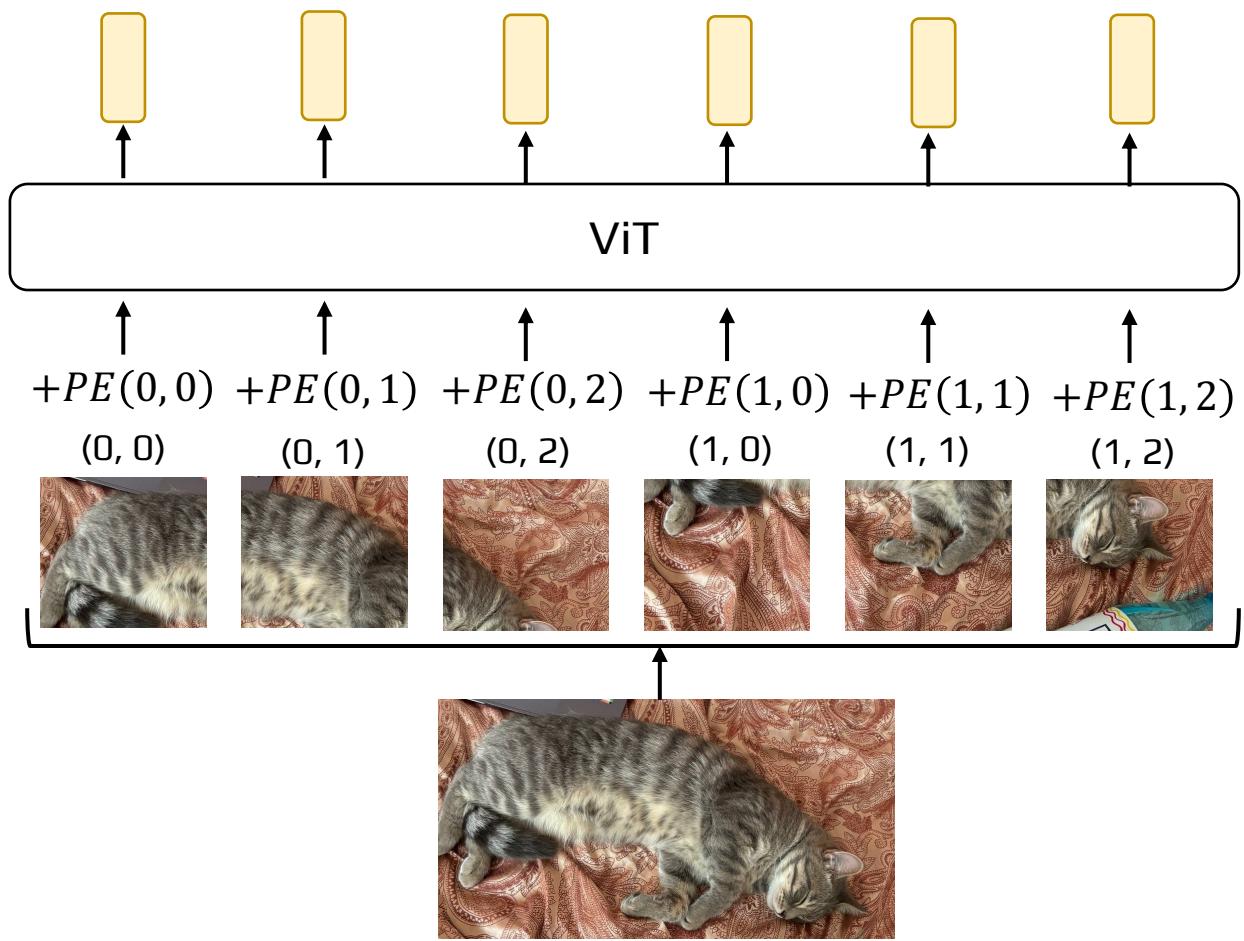
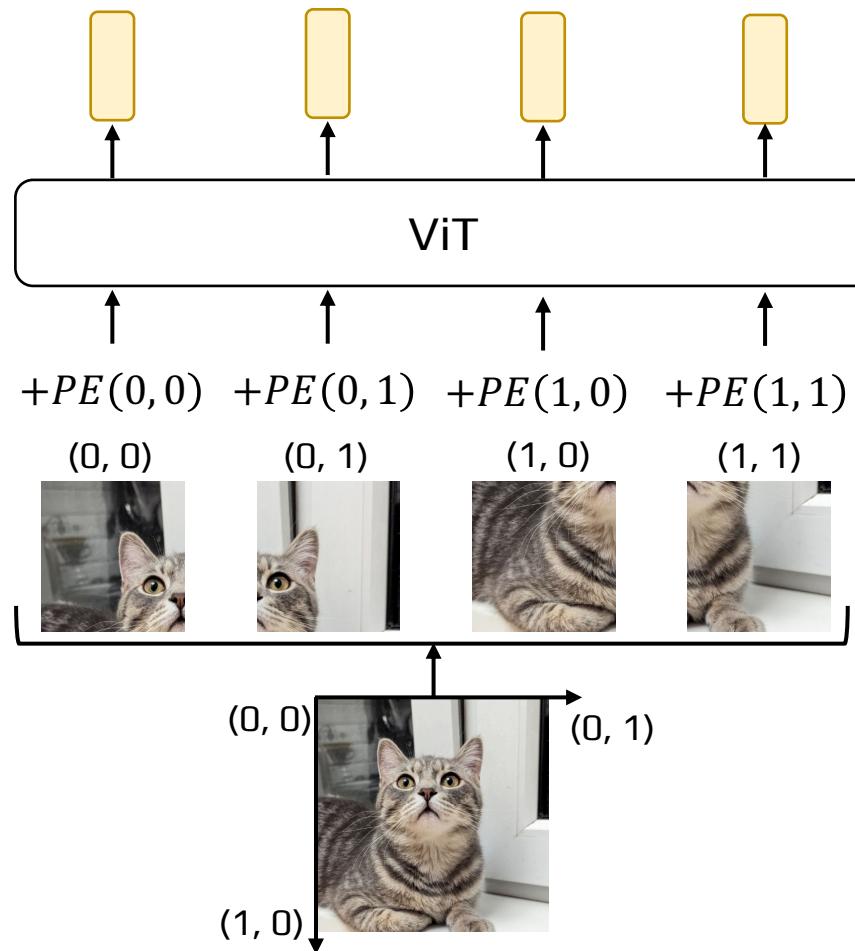
→ В оригинальном ViT позиционные эмбеддинги являются функцией от индекса патча → будут соответствовать разным областям изображения для разных входных изображений



# Подача произвольного числа патчей в ViT. Positional Encoding

**Решение:**

→ Новые позиционные эмбеддинги как функция от двумерных координат патча (в Qwen-2VL – 2D-RoPE)



1. Общая архитектура современных VLM
2. Вариации архитектуры VLM:
  - а) обучение без разморозки LLM
  - б) VLM без визуального энкодера
3. Обработка изображений
  - а) high-res
  - б) агрегация визуальных токенов

## Агрегация визуальных токенов

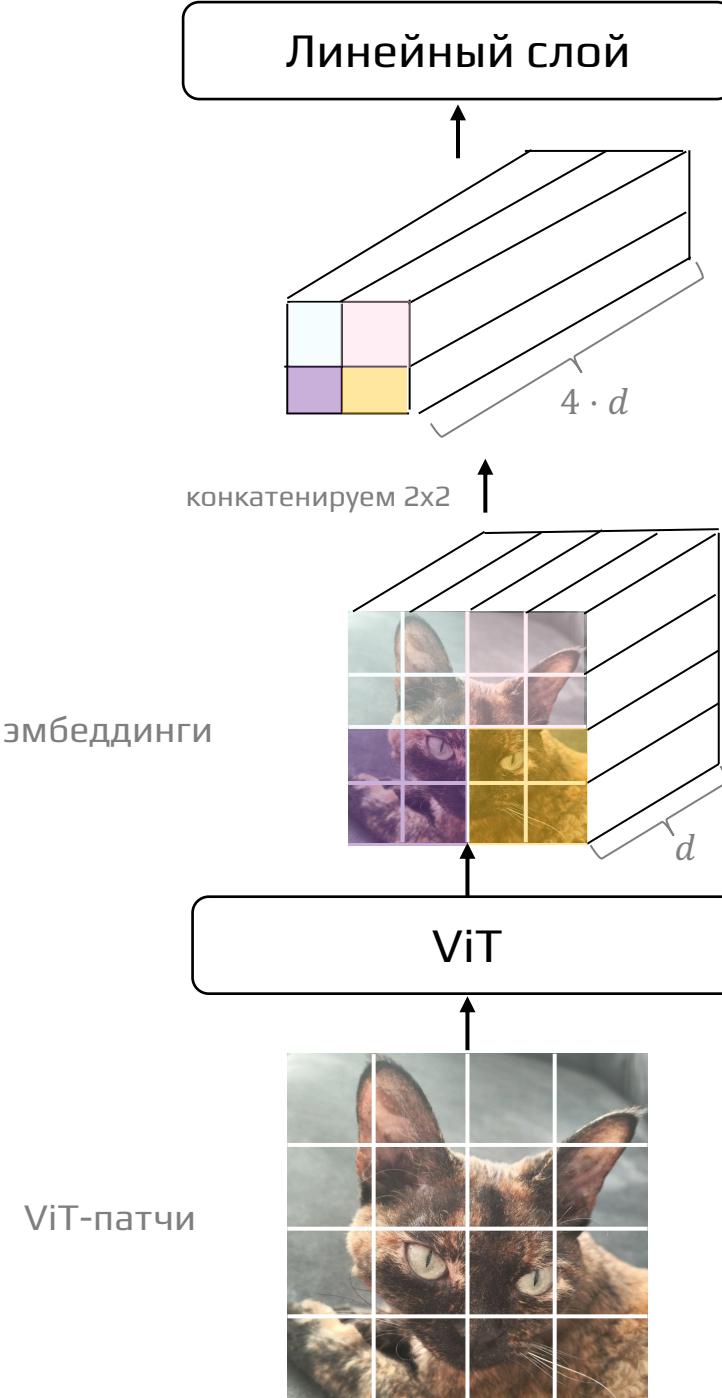
### Проблема:

- Для high-res изображений получаем слишком много визуальных токенов

### Решение:

- Применять некоторую агрегацию к визуальным токенам

«Pixel Shuffle» в InternVL:

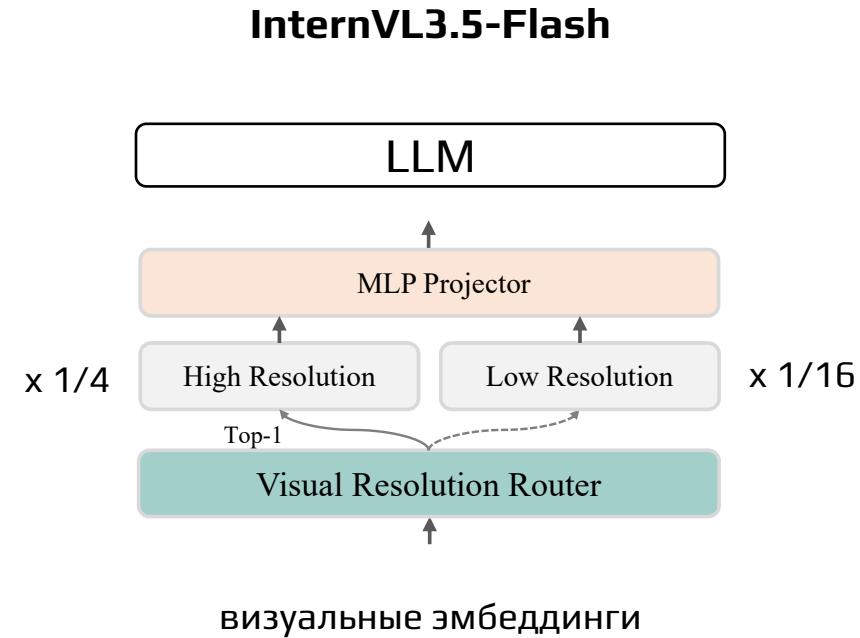


# Динамическая агрегация визуальных токенов. InternVL3.5 (Август 2025)

→ В InternVL3.5-Flash

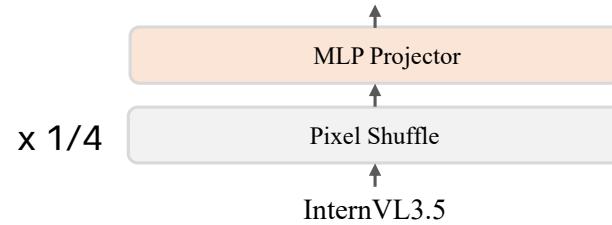
используют 2 агрегатора:  
один выдаёт в 4 раза меньше  
эмбеддингов, чем другой.

→ Выбор между ними — по  
изображению



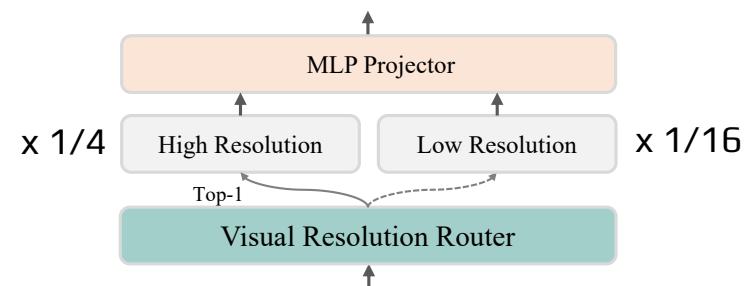
# Обучение Visual Resolution Router

1: Обзывают VLM стандартным образом  
→ далее её используют как reference



2: Добавляют 2-й агрегатор. Для каждого изображения выбирают степень сжатия случайно. Дистиллируют reference model

$$\text{Loss: } \mathbb{E}_{c \sim [\frac{1}{4}, \frac{1}{16}]} [\text{KL} (\pi_{\theta_{\text{ref}}} || \pi_{\theta, c})]$$



3: Добавляют ViT. Обзывают только его на бинарную классификацию

$$r_i = \frac{\text{KL} \left( \pi_{\theta_{\text{ref}}} || \pi_{\theta, \frac{1}{16}} \right)}{\text{KL} \left( \pi_{\theta_{\text{ref}}} || \pi_{\theta, \frac{1}{4}} \right)}$$

Чем больше  $r_i$ , тем хуже работает большее сжатие для данного изображения

$$\tau = \text{median}(r_i)$$

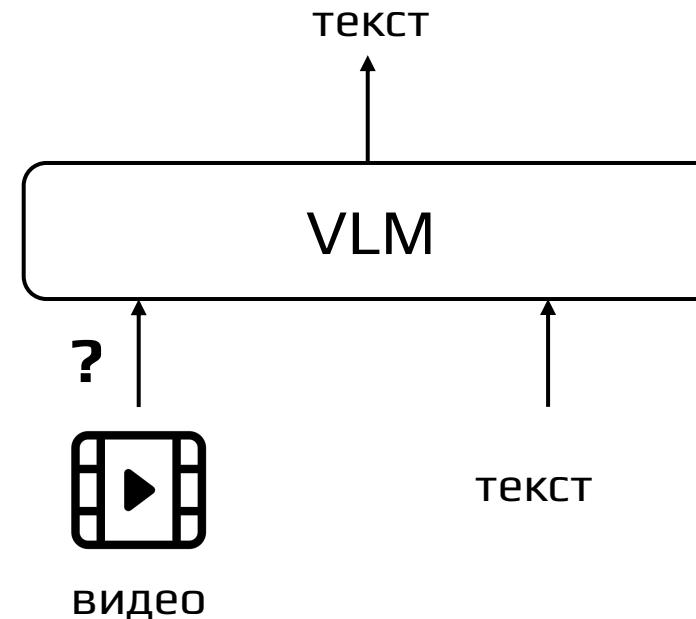
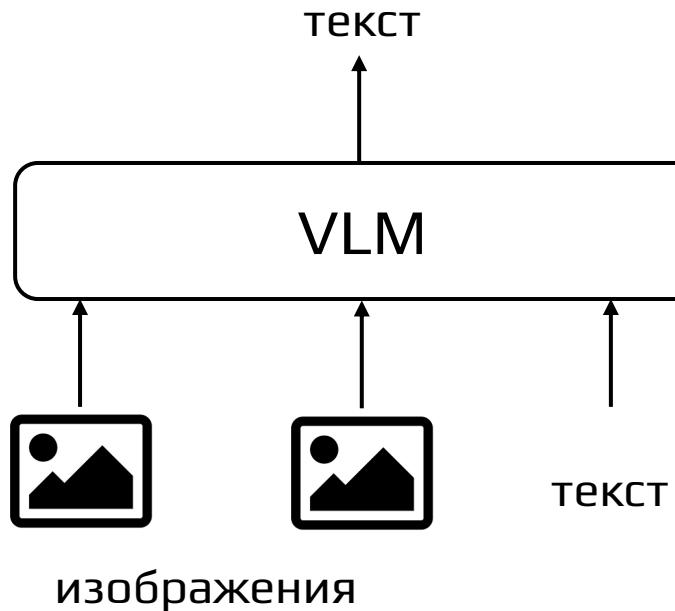
$$\text{target} = \begin{cases} \frac{1}{16}, & r_i < \tau \\ \frac{1}{4}, & r_i \geq \tau \end{cases}$$

(большее сжатие не сильно хуже меньшего)  
(большее сжатие сильно хуже меньшего)

- 1. Общая архитектура современных VLM**
- 2. Вариации архитектуры VLM:**
  - а) обучение без разморозки LLM**
  - б) VLM без визуального энкодера**
- 3. Обработка изображений**
  - а) high-res**
  - б) агрегация визуальных токенов**
- 4. Обработка видео**

# Работа с видео

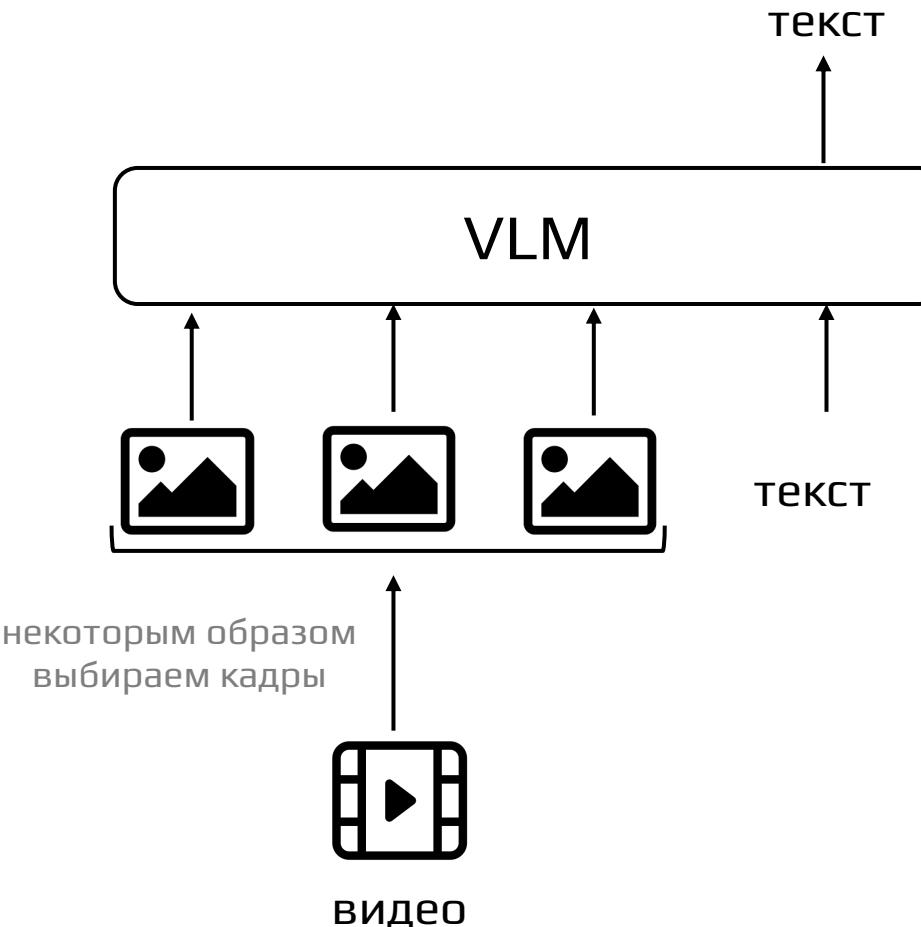
**Вопрос:** как подать видео в VLM, принимающую на вход только изображения?



# Работа с видео

**Вопрос:** как подать видео в VLM, принимающую на вход только изображения?

**Ответ:** как несколько кадров



1. Общая архитектура современных VLM
2. Вариации архитектуры VLM:
  - а) обучение без разморозки LLM
  - б) VLM без визуального энкодера
3. Обработка изображений
  - а) high-res
  - б) агрегация визуальных токенов
4. Обработка видео
5. Решение задач компьютерного зрения

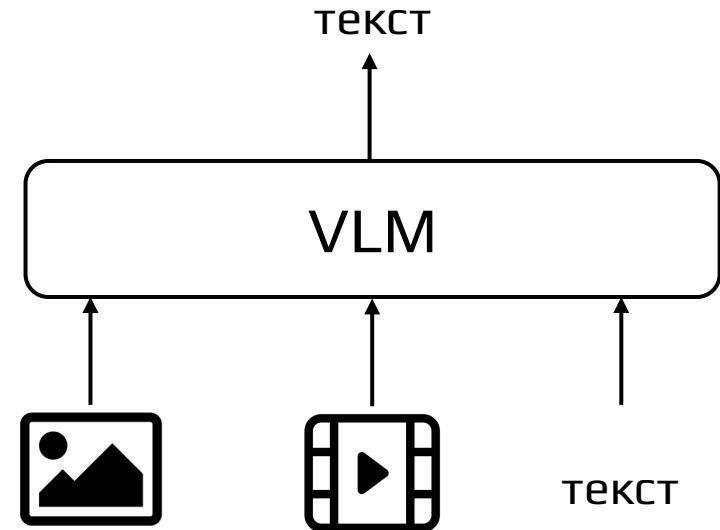
# Решение задач компьютерного зрения с помощью VLM

## Задачи, которые решаем хорошо:

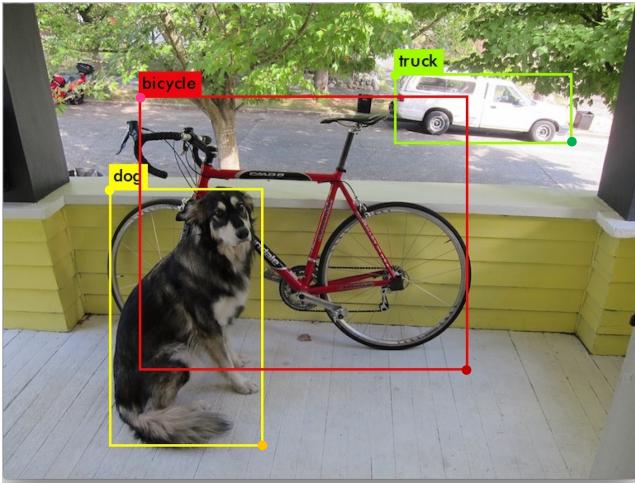
- Классификация
- Image Captioning
- Visual Question Answering
- OCR

## Задачи, с которыми могут возникнуть трудности:

- Детекция
- Сегментация



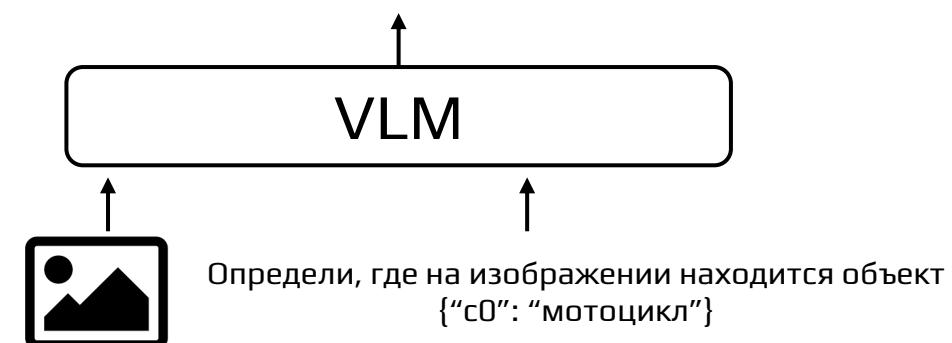
# Представление ответа в виде текста



```
{  
    "c0": [108, 193, 268, 463],  
    "c1": [410, 75, 590, 150],  
    "c2": [140, 95, 485, 385],  
}
```

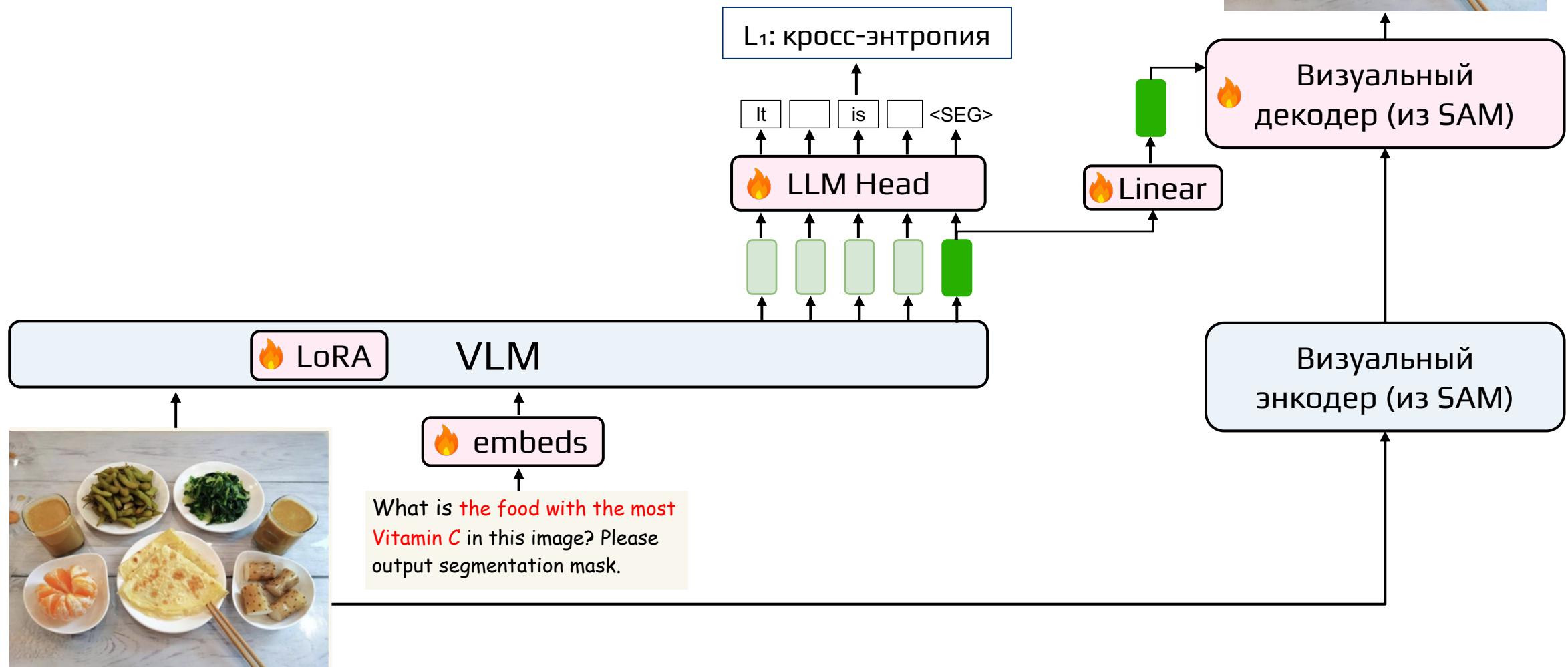


```
[(<c0>, 135.3, 95.7, 123.4, 53.4, 84.9, 57.6, 66.8,  
60.5, 60.1, 72.3, 34.2, 71.4, ..., 124.9, 119.3)].
```

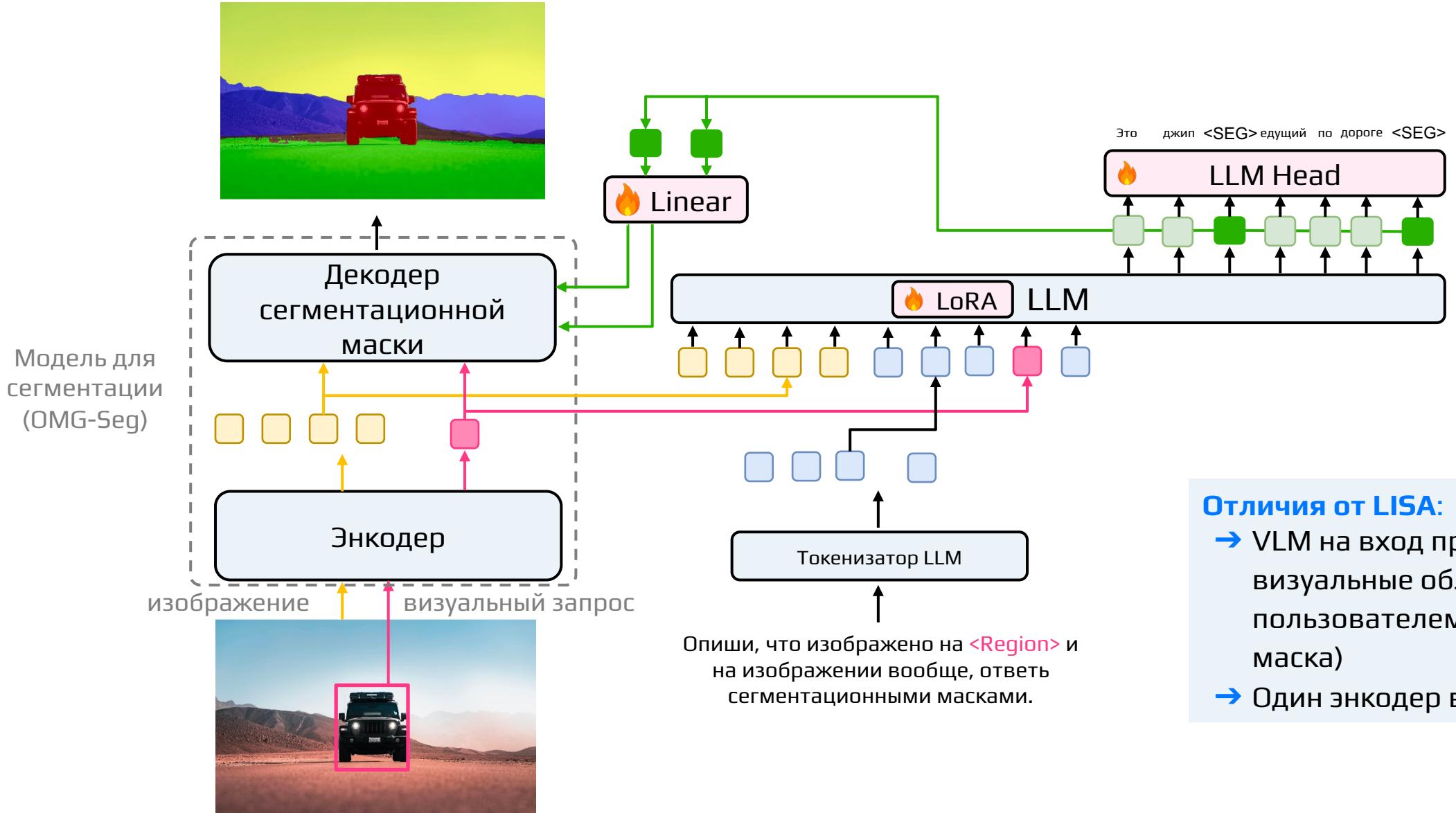


## LISA (2024).

**Идея:** с помощью VLM генерировать ответ в виде эмбеддинга  
(т.н. dense представление вместо sparse)



# OMG-LLaVA (2024).



1. Общая архитектура современных VLM
2. Вариации архитектуры VLM:
  - а) обучение без разморозки LLM
  - б) VLM без визуального энкодера
3. Обработка изображений
  - а) high-res
  - б) агрегация визуальных токенов
4. Обработка видео
5. Решение задач компьютерного зрения
6. Текущее состояние области

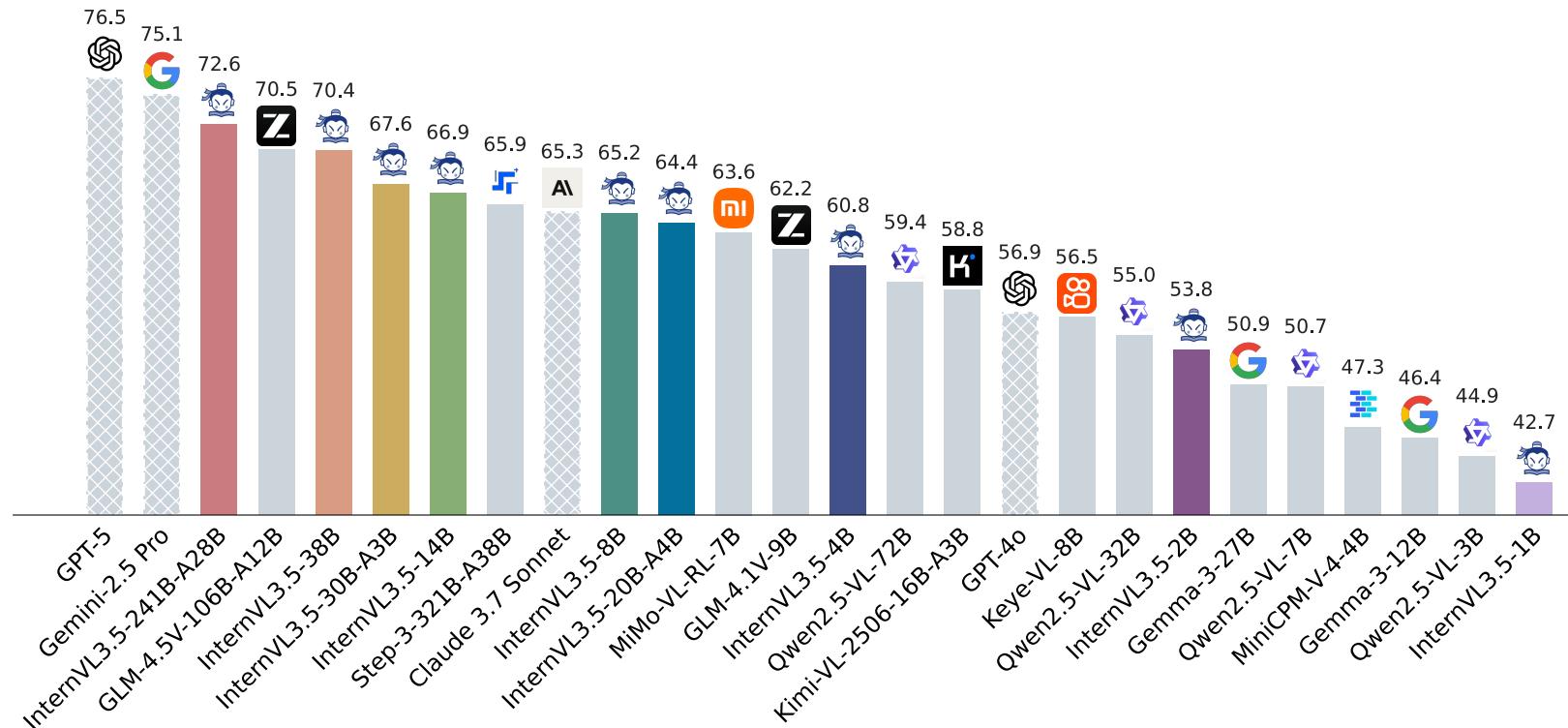
# Текущее состояние области

Передовые open-source модели:

- InternVL, GLM, QwenVL

За счёт чего улучшают качество:

- RLHF
- Увеличение объёма и разнообразия данных для обучения



Взято из статьи InternVL-3.5  
(например, нет Claude 4 Sonnet)

Спасибо  
за внимание!

