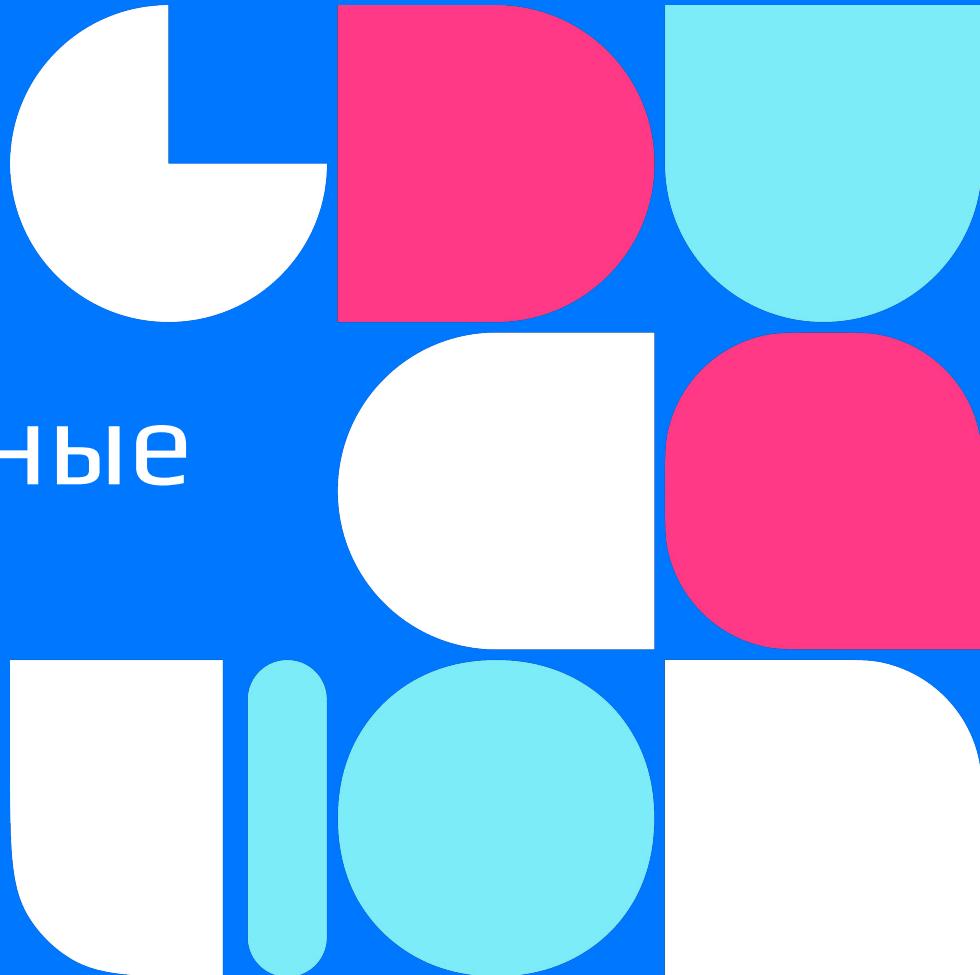




# Введение в мультимодальные модели

Курс "Мультимодальные модели"

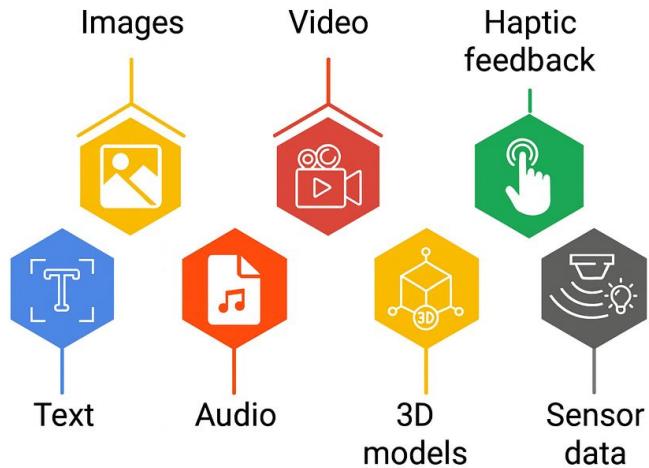


# Цель занятия:

Понять, что такое мультимодальные модели, разобрать их примеры, принципы работы, способы обучения и практическое применение.

# Что такое модальность?

- Изображения — фото, рисунки, диаграммы (2D-пространство)
- Текст — символы, слова, документы (дискретные токены)
- Аудио — речь, музыка, шумы (временной сигнал)
- Видео — последовательность кадров + звук (пространство + время)
- Сенсоры — LiDAR, IMU, EEG, телеметрия (нестандартные сигналы)
- Структурированные данные — табличные, временные ряды



Модальность = тип сигнала / данных

# Мультимодальность как геймченджер

- Больше контекста → глубже понимание.
- Zero-shot и перенос → новые задачи без обучения.
- Робастность → устойчива к шуму и сбоям.
- Унификация → один пайплайн вместо зоопарка моделей.



# Задачи. Image Captioning

Что делает модель?

Интерпретирует визуальные признаки и описывает их словами

Применения:

- Поиск и индексация
- Инклюзия, скринридеры для слабовидящих
- Подписи в социальных сетях

Ограничения:

- Склонность к шаблонным описаниям
- Возможны «галлюцинации»



«На фотографии изображены три человека, сидящие за столом на открытом воздухе. Один из них — женщина с длинными темными волосами, одетая в светлую блузку. Она улыбается и держит в руках несколько карточек с изображениями. Рядом с ней сидит мужчина с кудрявыми волосами, одетый в белую футболку и кепку. Он смотрит на женщину. Третий человек, сидящий справа, носит желтую кепку и сосредоточенно смотрит на стол. На столе расположены карточки и другие предметы, возможно связанные с игрой или мероприятием. На заднем плане видны деревья и палатки, что указывает на то, что действие происходит на фестивале или мероприятии на свежем воздухе»

# Задачи. Text-to-Image

Что делает модель?

Превращает описание на естественном языке в картинку

Применения:

- Генерация иллюстраций, мемов, контента
- Дизайн и прототипирование
- Игры, кино, реклама

Ограничения:

- Галлюцинации и артефакты
- Сложность управления стилем и деталями
- Этические риски (deepfake, авторские права)

## Text-to-Image

(Текст → Изображение)



Генерация изображения по текстовому запросу

Что делает модель?

Превращает описание на естественном языке в картинку

Применения:

- Генерация иллюстраций, мемов
- Дизайн и прототипирование
- Игры, кино, реклама

Ограничения:

- Галлюцинации и артефакты
- Сложность управления стилем

Запрос: "A futuristic city at sunset, digital art"



Запрос: Сгенерируй картинку для этого слайда

# Задачи. Cross-modal Retrieval

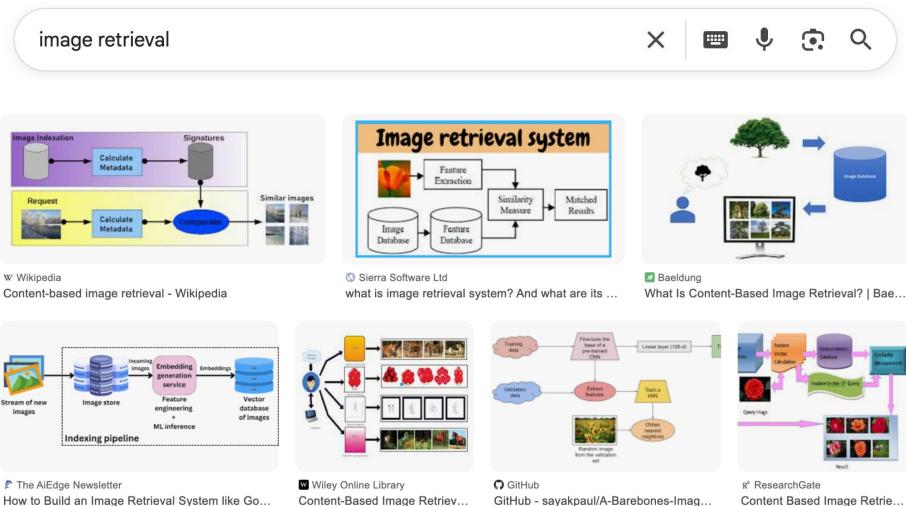
Поиск картинок по тексту / текста по картинке

Как работает?

- Кодируем картинку и текст в общее пространство эмбеддингов
- Сравниваем сходство (расстояние, dot product)
- Можем строить ANN индексы

Применения:

- Поиск изображений и видео в медиасервисах
- Рекомендательные системы
- Модерация и дедупликация контента



# Задачи. Visual Question Answering (VQA)

Ответ на вопрос по изображению

Что делает модель?

- Совмещает визуальные признаки и текст вопроса
- Генерирует текстовый ответ

Применения:

- Ассистенты, отвечающие по картинкам
- Образование и интерактивные системы
- Анализ изображений и видео в медиа/медицине



Запрос:

Что написано на фото?



Запрос:

Это пёс или хлеб?



Запрос:

Где происходят события на фото?

# Задачи. Video Question Answering (Video QA)

Ответ на вопрос по видео

Что делает модель?

- Анализирует последовательность кадров и звук
- Использует временной контекст для ответа

Применения:

- Поиск и Q&A по видеоконтенту
- Медиамониторинг и аналитика
- Образовательные системы (объяснение видеороликов)

Ограничения:

- Длинные видео = высокие вычислительные затраты
- Сложность интеграции аудио + визуального ряда

Запрос: Какой контент демонстрируется, когда на экране появляется обезьяна, и какой аспект интернет-культуры он подчеркивает?

# Задачи. Audio ↔ Text

ASR (Automatic Speech  
Recognition)

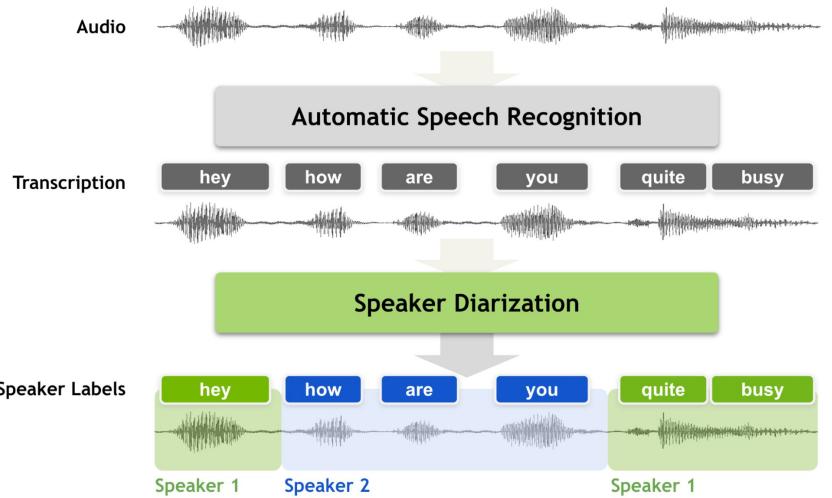
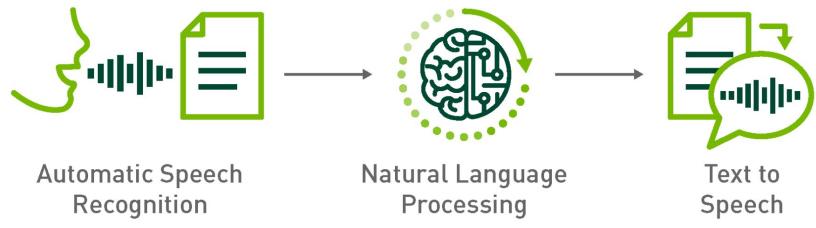
Распознавание речи в текст

TTS (Text-to-Speech)

Озвучивание текста голосом

Audio-Visual задачи

- Синхронизация речи и губ
- Определение говорящего
- Эмоциональная окраска речи



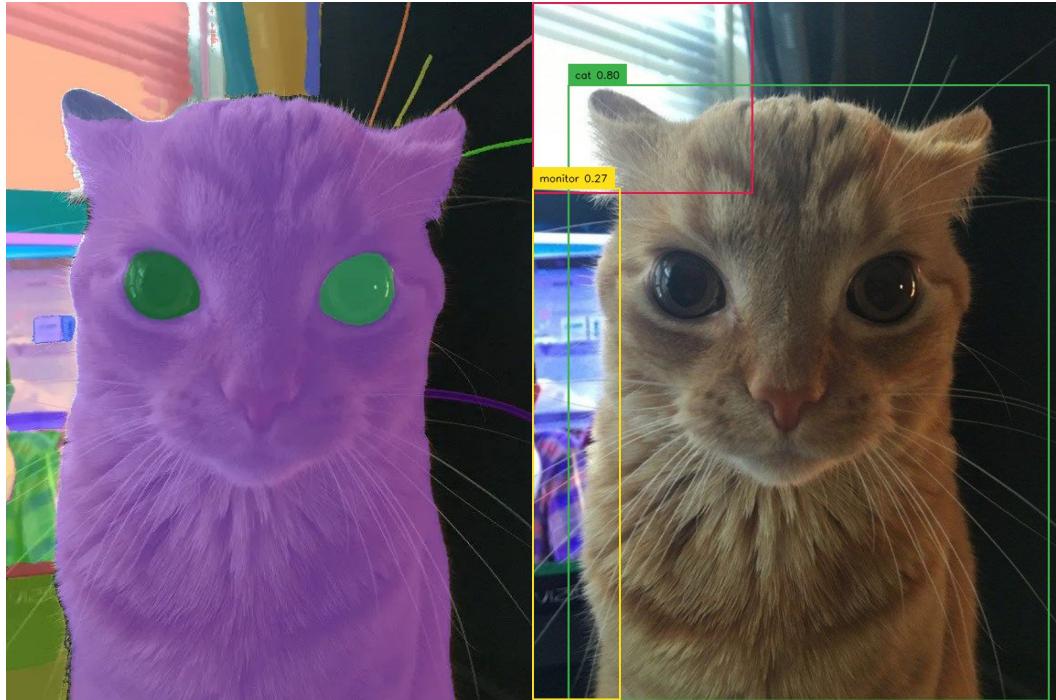
# Задачи. Grounding

Что делает модель?

- Находит объект на изображении по текстовому описанию
- Соотносит слова и визуальные регионы

Применения:

- Робототехника и ассистенты («возьми красную кружку»)
- Интерактивные интерфейсы (редактирование картинок по тексту)
- Визуальный поиск (указание точного объекта)



# Задачи. Документы и OCR

Что делает модель?

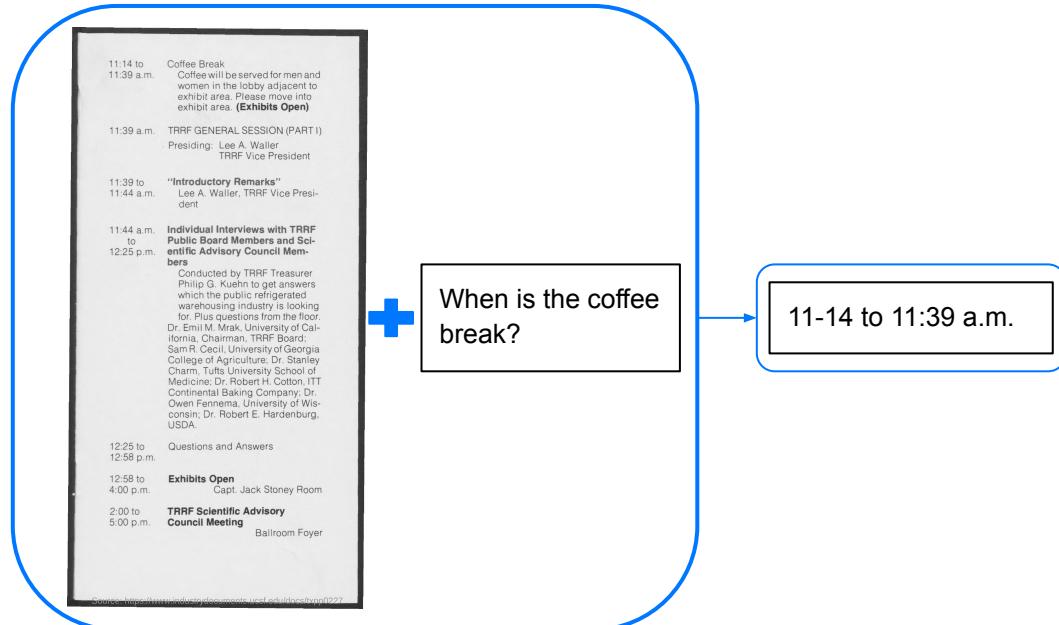
- Считывает текст с изображений (OCR)
- Понимает структуру документа (поля, таблицы, подписи)
- Отвечает на вопросы по содержанию документа

Применения:

- Юридические и финансовые документы
- Автоматизация бэк-офиса (счета, накладные, анкеты)
- Медицина (анализ отчётов, рецептов, историй болезни)

Особенности:

- OCR ≠ мультимодальная модель (нужна связка OCR + VLM)
- Важен контекст: текст сам по себе недостаточен без визуальной структуры



# Архитектурные подходы

## 1. Early Fusion (feature-level fusion)

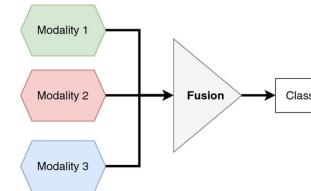
Все модальности сначала преобразуются в признаки низкого уровня.

Эти признаки конкатенируются/суммируются → подаются в один общий энкодер.

Пример: мультимодальный sentiment analysis (текст + аудио одновременно в сеть).

Плюс: сразу учится сквозная репрезентация.

Минус: взрывается размерность, шум легко ломает всю модель.



## 2. Late Fusion (decision-level fusion)

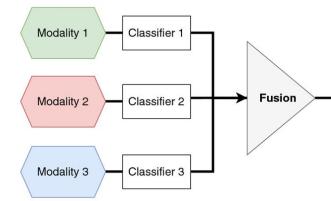
Каждая модальность обрабатывается своим энкодером.

На выходе мы объединяем только решения (например, усреднение вероятностей или метаклассификатор).

Пример: отдельные модели для аудио и видео, затем усреднение вероятностей «спорт» vs «интервью».

Плюс: модульность, легко добавлять/убирать модальности.

Минус: модель не учится глубоким связям между модальностями.



## 3. Shared Embedding Space (dual-encoder / contrastive)

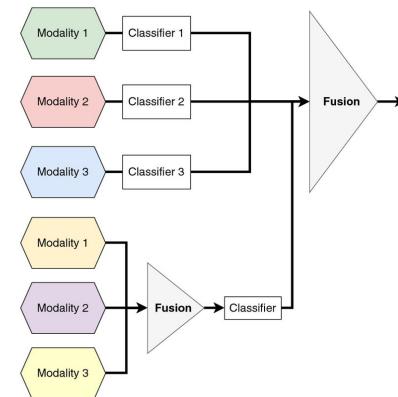
Для каждой модальности — свой энкодер.

Энкодеры учатся проектировать данные в общее пространство эмбеддингов.

Наиболее известный пример — CLIP (текстовый энкодер + визуальный энкодер).

Плюс: хорошо работает для retrieval, масштабируется на большие коллекции.

Минус: нет сложного взаимодействия, только сравнение эмбеддингов.



## 4. Cross-Encoder / Cross-Attention

Один энкодер принимает токены всех модальностей вместе или через cross-attention.

Текстовые токены «смотрят» на визуальные токены и наоборот.

Пример: Flamingo, LXMERT, BLIP.

Плюс: глубокая интеграция → reasoning и генерация.

Минус: вычислительно тяжело, плохо масштабируется на большие коллекции.

# Contrastive Loss

Contrastive Loss (InfoNCE / Softmax)

- Положительные пары ближе
- Отрицательные пары дальше
- Основа CLIP

Sigmoid / Binary Loss

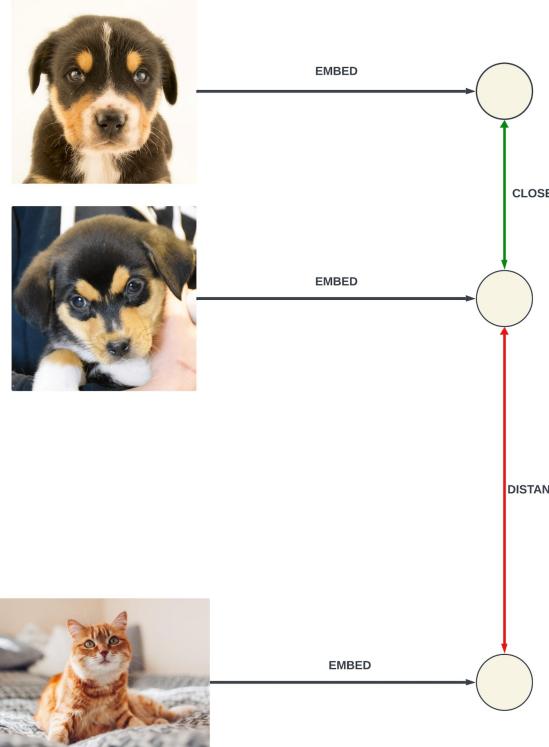
- Каждая пара рассматривается независимо
- Нет нормализации по батчу (пример: SigLIP)

Matching Loss (ITM)

- Классификация: «пара соответствует / не соответствует»

Masked Modeling

- Маскируем часть данных (текст/изображение)
- Модель восстанавливает недостающее (пример: ALBEF, BLIP)



## Contrastive Loss. Formal

$$L = \mathbf{1}[y_i = y_j] \|x_i - x_j\|^2 + \mathbf{1}[y_i \neq y_j] \max(0, \epsilon - \|x_i - x_j\|^2)$$

- Если  $y_i = y_j \rightarrow$  минимизируем расстояние между эмбеддингами
- Если  $y_i \neq y_j \rightarrow$  увеличиваем расстояние до порога  $\epsilon$

# CLIP (Contrastive Language–Image Pretraining)

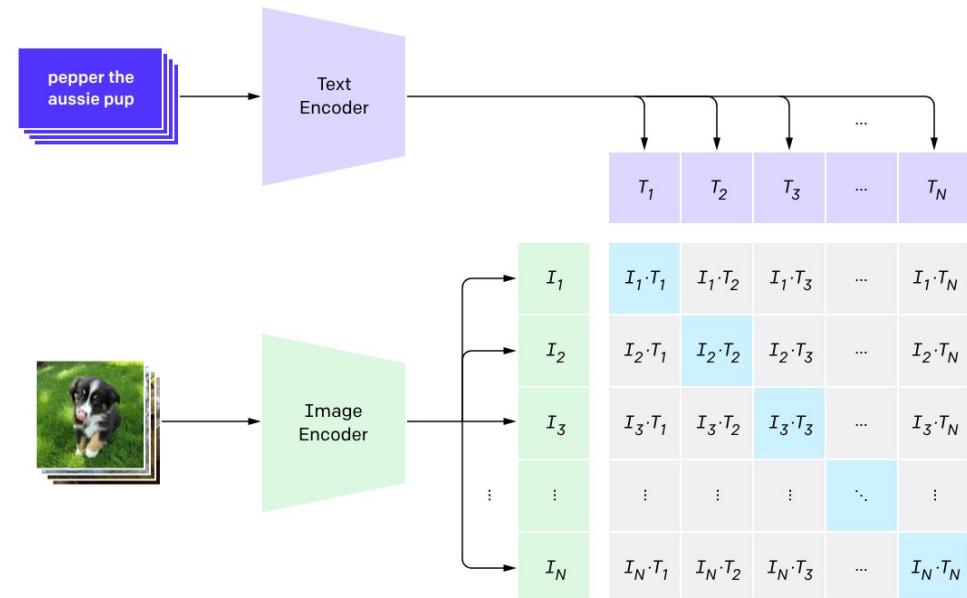
Два энкодера:

- Image Encoder (ResNet / ViT)
- Text Encoder (Transformer)

Проекция в общее  
пространство эмбеддингов

Контрастивное обучение:

- Сближаем правильные пары
- Отталкиваем неправильные



# CLIP. Код

```
● ○ ●

# Псевдокод CLIP-лосса
import torch
import torch.nn.functional as F

# img_emb, txt_emb: [batch, dim]
img_emb = F.normalize(model.encode_image(images), dim=-1)
txt_emb = F.normalize(model.encode_text(texts), dim=-1)

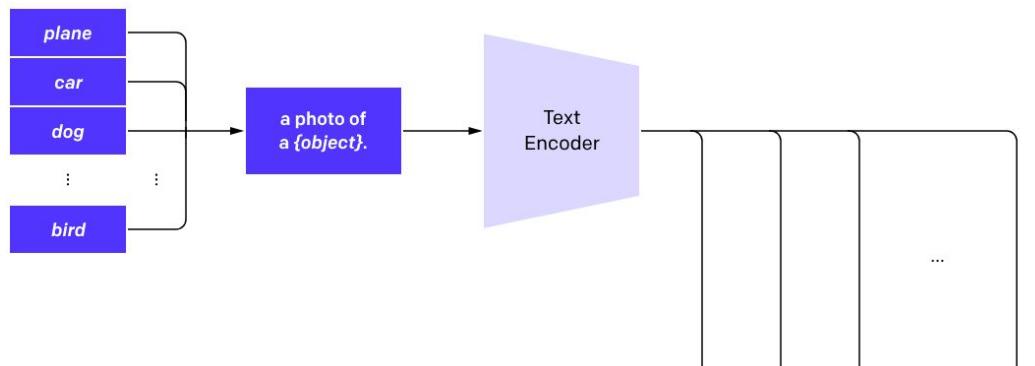
# Матрица сходств (batch x batch)
logits = img_emb @ txt_emb.T # косинусное сходство

# Кросс-энтропия: каждая картинка должна матчиться со своим текстом
labels = torch.arange(len(logits)).to(logits.device)
loss_i2t = F.cross_entropy(logits, labels)      # image → text
loss_t2i = F.cross_entropy(logits.T, labels)    # text → image
loss = (loss_i2t + loss_t2i) / 2
```

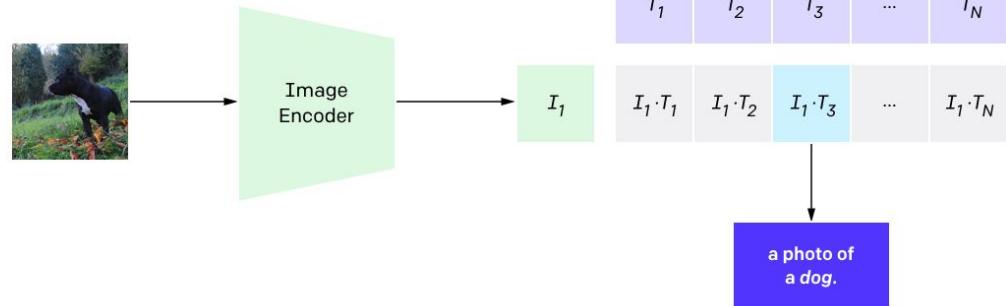
# Zero-shot классификация с CLIP

- Создаём текстовые представления классов
- Формируем псевдо-лейблы «*a photo of a {object}*»
- Кодируем их через Text Encoder
- Кодируем изображение через Image Encoder
- Сравниваем сходства между эмбеддингом картинки и текстов
- Выбираем ближайший класс
- Не нужно дообучать на новых классах — достаточно описать их словами.

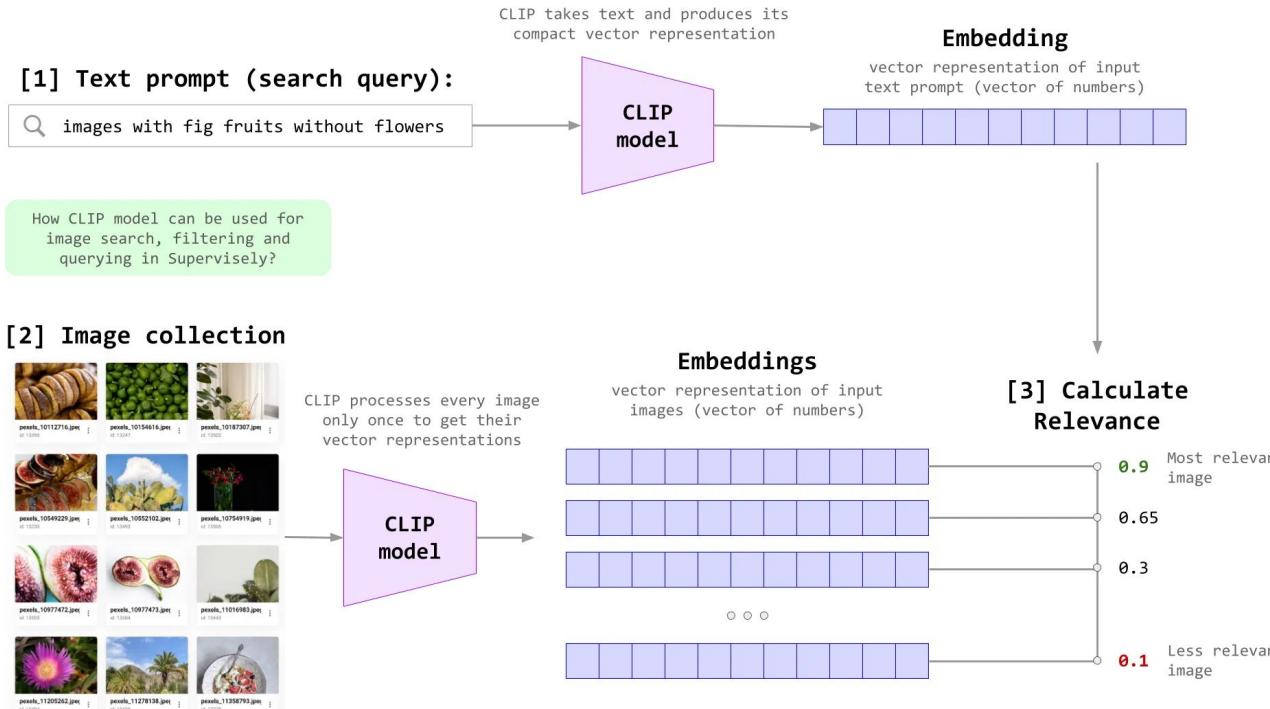
2. Create dataset classifier from label text



3. Use for zero-shot prediction



# Multimodal Retrieval



[How to run OpenAI CLIP with UI for Image Retrieval](#)

# Sigmoid Loss for Language–Image Pretraining

- Ванильный CLIP имеет набор нюансов, например:

- Обучение зависит от размера батча, мало «негативов» → слабый сигнал
- Предполагает один позитив в строке → мульти-позитивы подавляются
- При этом в батче часто есть ложные негативы, синонимы, похожие подписи
- Оптимизируем «классификацию в батче», а используем для retrieval

- Идея SigLIP

- Рассматриваем каждую пару (изображение., текст) независимо, получаем бинарную классификацию
- Можем иметь несколько позитивов для одного изображения/текста
- Нет нормализации по батчу
- Лучше калибровка под retrieval

# SigLIP

$\mathbf{x}_i, \mathbf{y}_i$  — эмбеддинги изображения и текста

$t$  — learnable temperature (масштаб логитов)

$b$  — bias

$z_{ij}$  — метка: +1 для «пара подходит», -1 для «пара не подходит»

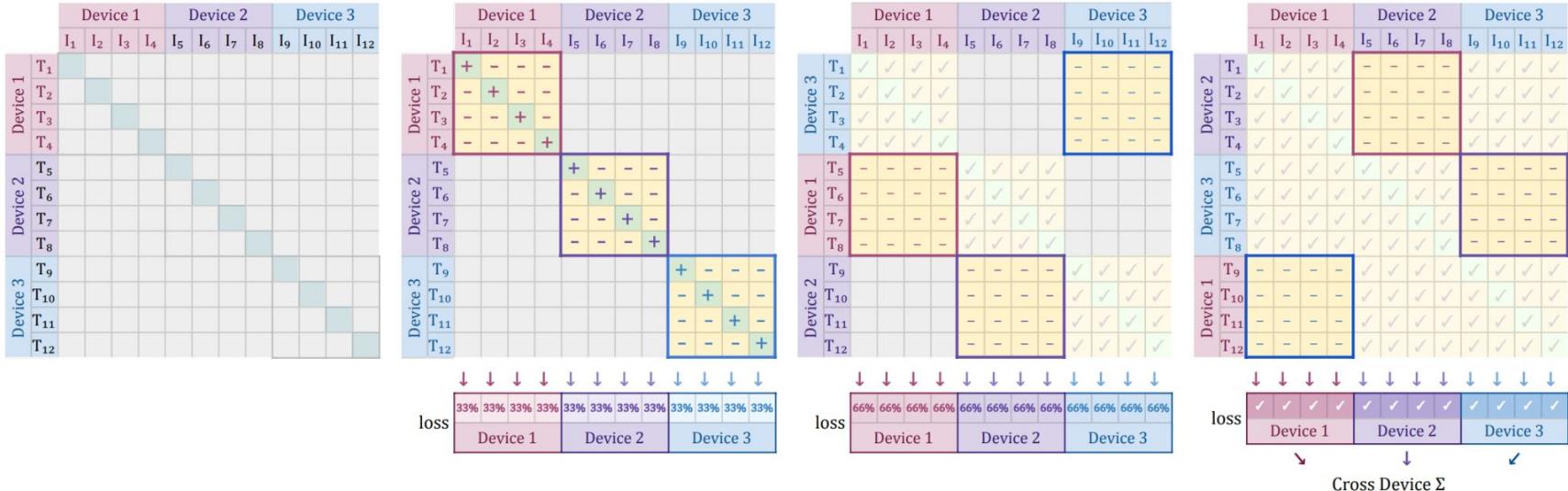
CLIP

$$-\frac{1}{2|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \left( \overbrace{\log \frac{e^{t\mathbf{x}_i \cdot \mathbf{y}_i}}{\sum_{j=1}^{|\mathcal{B}|} e^{t\mathbf{x}_i \cdot \mathbf{y}_j}} + \log \frac{e^{t\mathbf{x}_i \cdot \mathbf{y}_i}}{\sum_{j=1}^{|\mathcal{B}|} e^{t\mathbf{x}_j \cdot \mathbf{y}_i}}}^{\text{image} \rightarrow \text{text softmax}} \right)$$

SigLIP

$$-\frac{1}{|\mathcal{B}|} \sum_{i=1}^{|\mathcal{B}|} \sum_{j=1}^{|\mathcal{B}|} \underbrace{\log \frac{1}{1 + e^{z_{ij}(-t\mathbf{x}_i \cdot \mathbf{y}_j + b)}}}_{\mathcal{L}_{ij}}$$

# SigLIP. Efficient loss implementation



Изначально каждый девайс держит по 4 изображения и 4 текста.  
Чтобы посчитать полный лосс, нужно обмениваться представлениями с другими GPU.

Каждое устройство считает свой компонент лосса (выделенные клетки), включая позитивные пары.

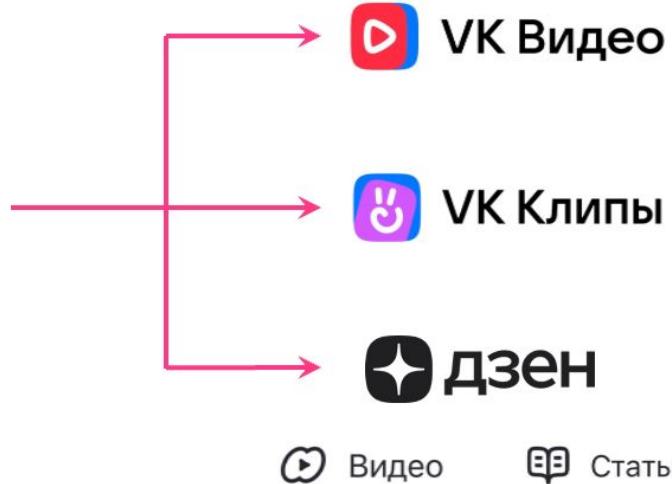
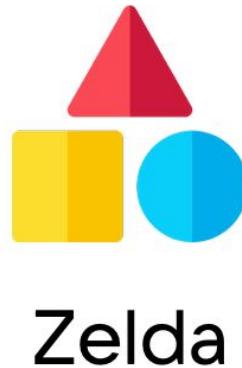
Тексты «меняются местами» между устройствами: теперь устройство 1 содержит изображения, которые изначально и были на нем, а тексты, которые были на устройстве 2.

Этот процесс повторяется, пока все пары «изображение–текст» не будут учтены.  
В конце все компоненты суммируются (Cross Device  $\Sigma$ ).

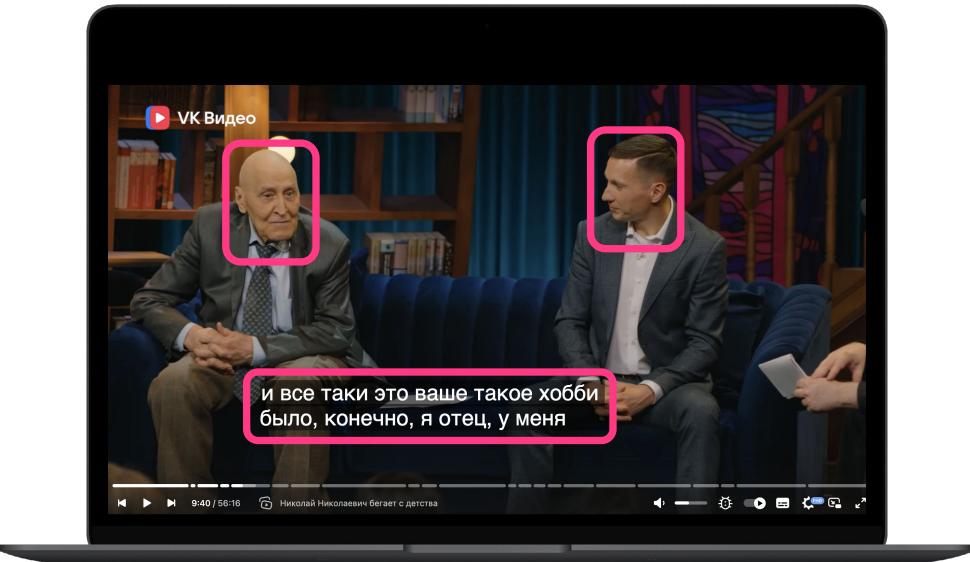
# Zelda: фреймворк и семейство моделей

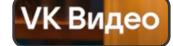
[Евгений Астафуров | Как мы обучали контентный видео-энкодер не используя размеченные данные](#)

[Илья Алтухов | Разработка мультимодальных контентных моделей для рекомендаций](#)

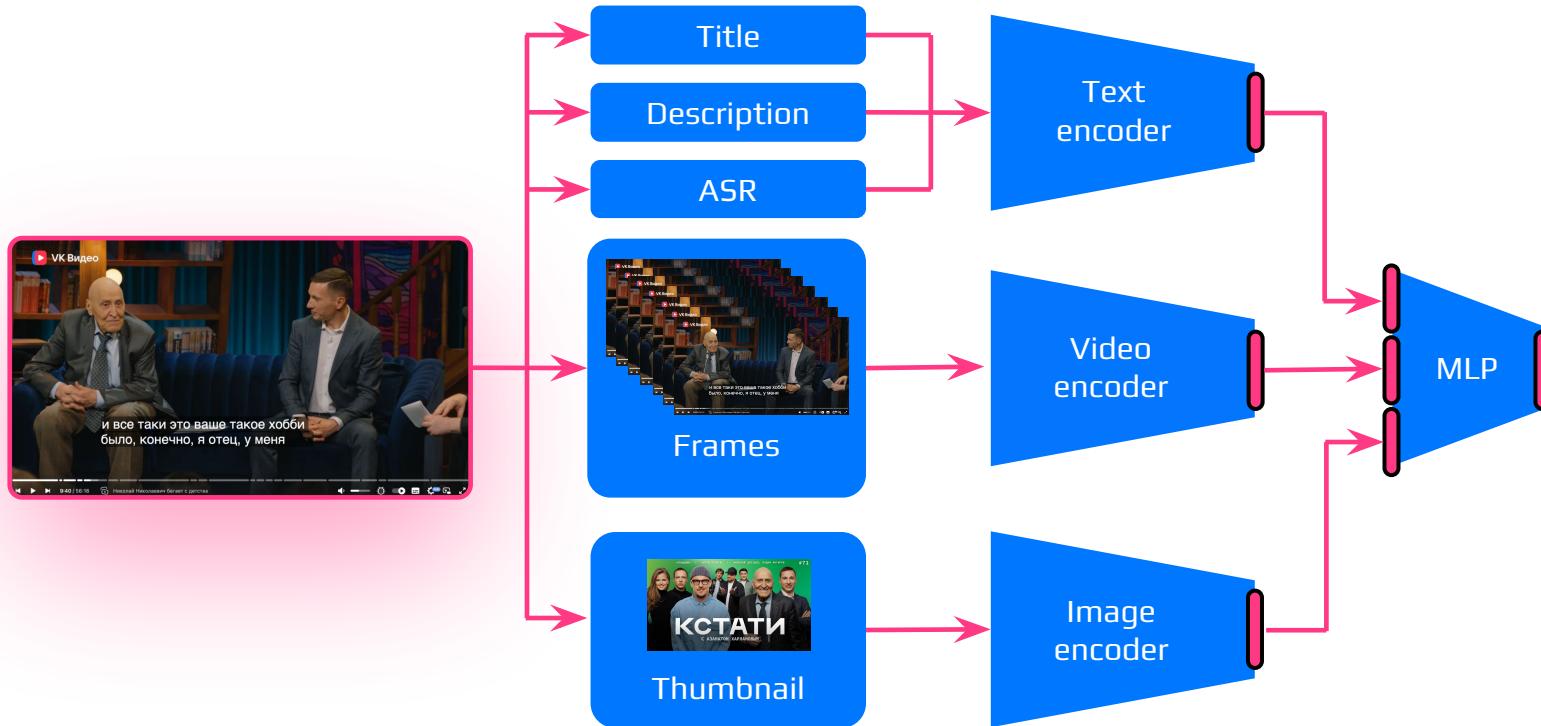


# Какая информация содержится в контенте



- Обложка ..... 
- Аудио ..... 
- Люди ..... 
- Текст ..... 
- Видео ..... 
- Субтитры ..... 
- Название, описание, автор...

# Zelda. Архитектура



# Zelda. Обучение

Между релевантными  
парами

$X_{1,1}$	$X_{1,2}$	$X_{1,3}$
$X_{2,1}$	$X_{2,2}$	$X_{2,3}$
$X_{3,1}$	$X_{3,2}$	$X_{3,3}$

Документ 2

Документ 1

$$infoNCE = -\log \frac{\exp(sim(x_i, y_i)/\tau)}{\sum_k \exp(sim(x_i, y_k)/\tau)}$$

$$\mathcal{L} = \alpha \cdot \frac{1}{N} \sum infoNCE_{ij} + \beta \cdot infoNCE_{pairs}$$

Между модальностями одного  
документа

$X_{1,1}$	$X_{1,2}$	$X_{1,3}$	$X_{1,4}$	$X_{1,5}$	$X_{1,6}$
$X_{2,1}$	$X_{2,2}$	$X_{2,3}$	$X_{2,4}$	$X_{2,5}$	$X_{2,6}$
$X_{3,1}$	$X_{3,2}$	$X_{3,3}$	$X_{3,4}$	$X_{3,5}$	$X_{3,6}$
$X_{4,1}$	$X_{4,2}$	$X_{4,3}$	$X_{4,4}$	$X_{4,5}$	$X_{4,6}$
$X_{5,1}$	$X_{5,2}$	$X_{5,3}$	$X_{5,4}$	$X_{5,5}$	$X_{5,6}$
$X_{6,1}$	$X_{6,2}$	$X_{6,3}$	$X_{6,4}$	$X_{6,5}$	$X_{6,6}$

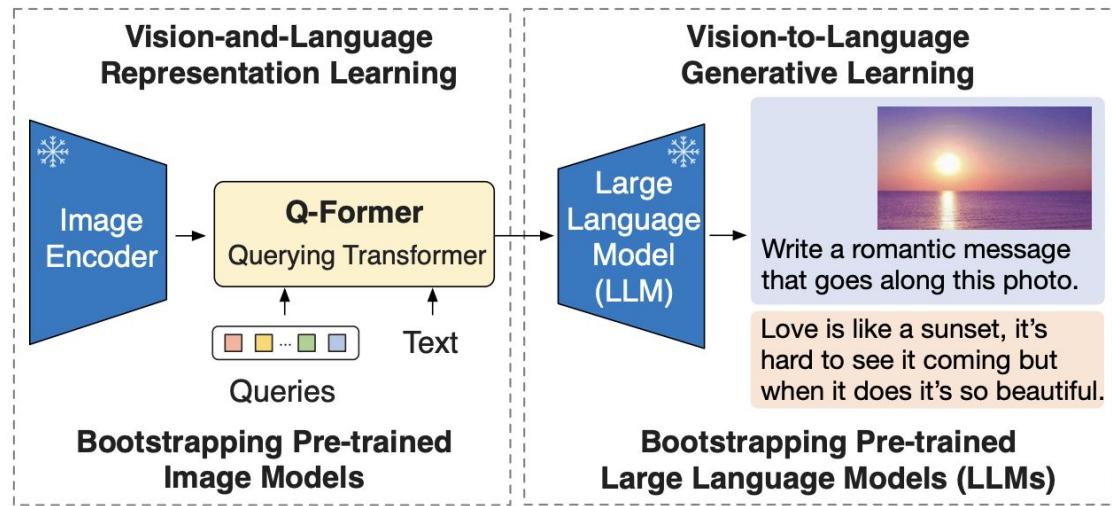
Модальность 1

Модальность 2

# BLIP (Bootstrapping Language–Image Pretraining)

## Главная идея

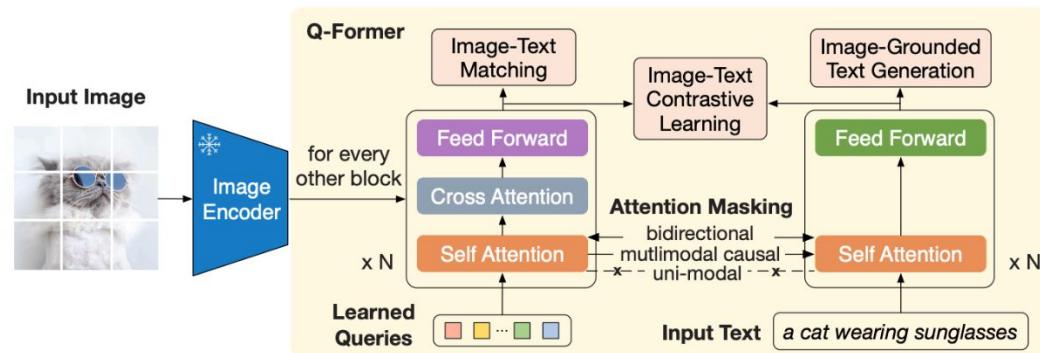
- Соединить **Vision Encoder** (ViT, замороженный) и **Large Language Model (LLM, замороженный)**
- Вставить **Q-Former (Query Transformer)** как «мост» между зрением и языком
- Обучать только Q-Former → дешево и эффективно



# BLIP. Q-Former

- В DETR (Detection Transformer) ввели «object queries» — этообучаемые векторы, которые **не являются данными картинки, а параметры модели.**
- Каждый такой query токен через cross-attention «смотрит» на все признаки картинки и вытягивает из них информацию.
- Представим, что у нас есть 200 фотографий (патчи ViT), и мы хотим объяснить их кому-то (LLM). Мы не отдаём все 200, а сначала задаёшь 32 вопроса вроде:
  - ❖ «Что тут главное?»
  - ❖ «Какие объекты на переднем плане?»
  - ❖ «Какие есть цвета/текстуры?»

И только эти 32 ответа говорим. Это и есть роль query токенов.



**Q-Former (Query Transformer):**

- **Query токены (learned queries)** — фиксированное число параметров, которые «задают вопросы» картинке.
- **Self-Attention** — связи между query токенами.
- **Cross-Attention** — query токены «смотрят» на эмбеддинги картинки от ViT.

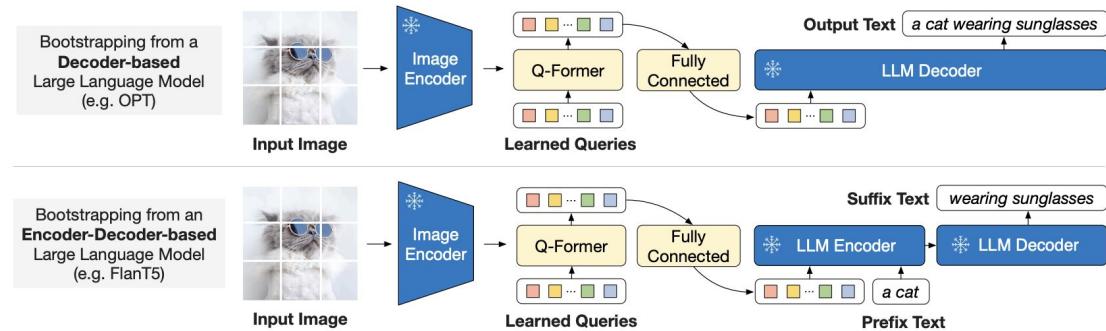
# BLIP. Vision-to-Language Generative Pre-training

**Bootstrapping = использование замороженных LLM**

- LLM (OPT, FlanT5 и др.) остаётся неизменным
- Обучается только Q-Former + FC слой

**Как работает:**

1. Image Encoder → эмбеддинги
2. Q-Former → query токены
3. FC layer → преобразует размерность
4. LLM → генерирует текст (caption, answer, диалог)



**Два сценария:**

- Decoder-only LLM (например, OPT): Q-Former токены идут прямо в **декодер** как префикс
- Encoder–Decoder LLM (например, FlanT5): Q-Former токены идут в **энкодер**, затем декодер генерирует ответ

# OWL-ViT: Open-Vocabulary Object Detection

Идея:

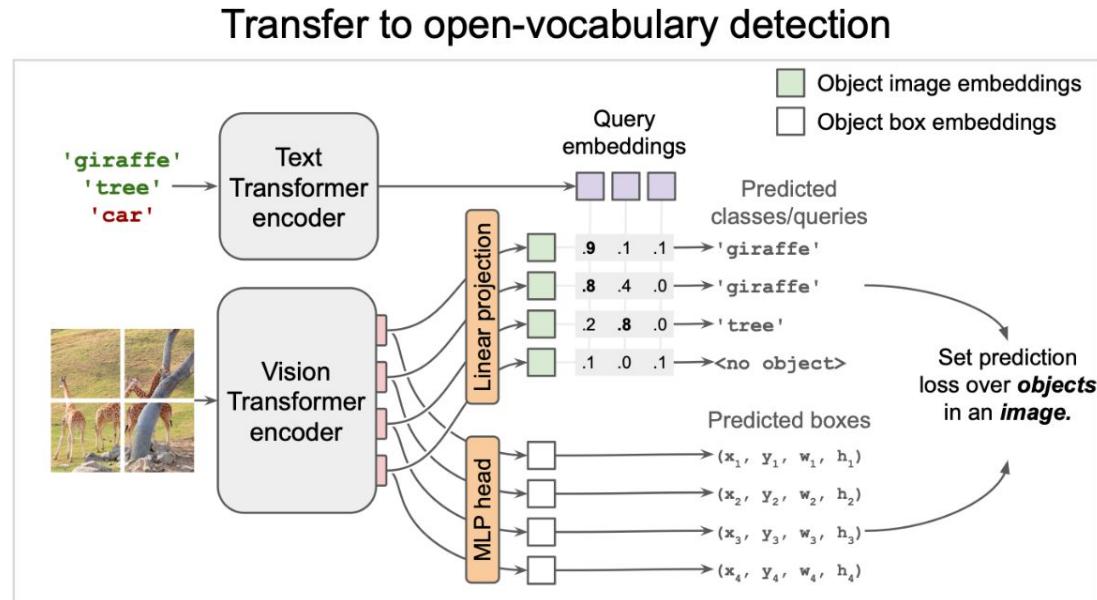
- Объединить CLIP-style обучение и детекторы объектов
- Вместо фиксированного словаря классов → поиск объектов по **произвольному текстовому запросу**

Как работает:

- Vision Transformer извлекает признаки
- Text Encoder кодирует запросы
- Сравнение эмбеддингов → bounding box + confidence

Компоненты функции потерь:

- Lcls — Для каждого object embedding выбирается лучший текстовый запрос
- L1 — регрессия боксов ( $x, y, w, h$ )
- LGIoU — Учитывает пересечение предсказанного и реального бокса, устойчив к непересекающимся боксам

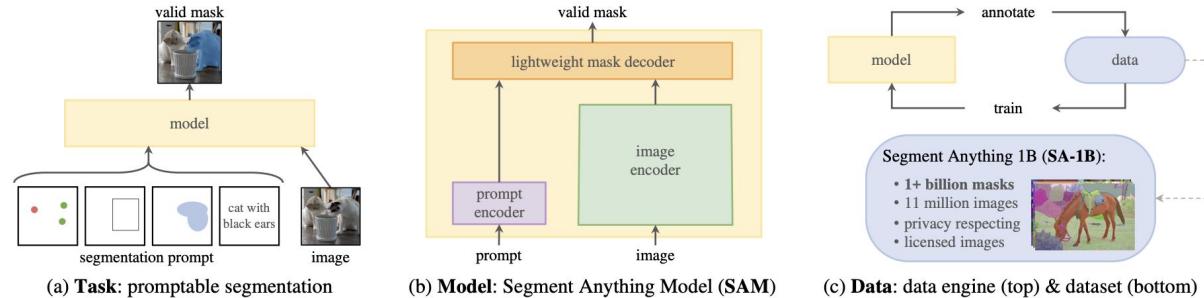


$$\mathcal{L} = \lambda_{\text{cls}} \mathcal{L}_{\text{cls}} + \lambda_{\text{L1}} \mathcal{L}_{\text{L1}} + \lambda_{\text{GIoU}} \mathcal{L}_{\text{GIoU}}$$

# SAM (Segment Anything Model)

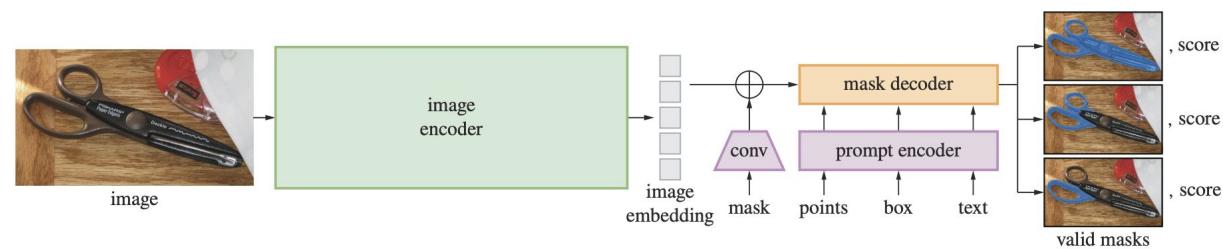
## Что такое SAM

- Foundation-модель для сегментации (Meta AI, 2023)
- Работает в zero-shot режиме: сегментирует объекты, которых не видела
- Promptable: выделяет объект по подсказке пользователя (точка, бокс, грубая маска)



## Архитектура

1. **Image Encoder** → патч-эмбеддинги
2. **Prompt Encoder** → токены для точки/бокса/маски
3. **Mask Decoder (Transformer + MLP)**  
cross-attention: prompt tokens ↔ image tokens, выдаёт несколько масок (обычно 3) + оценку качества (IoU score)



# Челленджи и будущее мультимодальных моделей



## Generalist multimodal models

один пайплайн для captioning, VQA, OCR, detection, segmentation



## Мультимодальные агенты

диалог и действия: текст + картинка + видео + сенсоры

## Grounding и интерактивность

понимание физического мира (AR/VR, робототехника)



## Open-source экосистема

LLaVA, InternVL, Grounded-SAM, QWEN VL



## Этика и ответственность

контроль качества, прозрачность данных, интерпретация



## Данные

шумные интернет-пары, bias, ограниченные доменные датасеты, дороговизна разметки

Спасибо  
за внимание!

