BERT, GPT, VIT И ДРУГИЕ

Сергей Николенко

СПбГУ — Санкт-Петербург

30 октября 2026 г.





Random facts:

- 30 октября 1888 г. Джон Лауд получил патент на шариковую ручку: она предназначалась для разметки грубой кожи, а не для писем
- 30 октября 1938 г. CBS передала «Войну миров» Орсона Уэллса в форме срочных радиорепортажей; часть слушателей приняла вымысел за новости
- 30 октября 1947 г. 23 страны подписали в Женеве Генеральное соглашение по тарифам и торговле — предшественник системы ВТО
- 30 октября 1961 г. над Новой Землёй испытали «Царь-бомбу»: около 50 мегатонн, самый мощный ядерный взрыв в истории
- 30 октября 1985 г. «Челленджер» отправился в полёт с восемью астронавтами — впервые столько людей стартовало на одном корабле
- 30 октября 1990 г. на Лубянской площади открыли Соловецкий камень — памятник жертвам политических репрессий

Краткое резюме предыдущей серии

- В прошлый раз мы собрали трансформер и разобрали современный рецепт: RoPE, нормализацию, SwiGLU, KV-кэш и FlashAttention.
- Сегодня сначала посмотрим на BERT и семейство GPT — как выбор задачи предобучения меняет применение трансформера.
- Затем заменим слова патчами и перейдём к ViT, Swin и трансформерам для распознавания объектов.

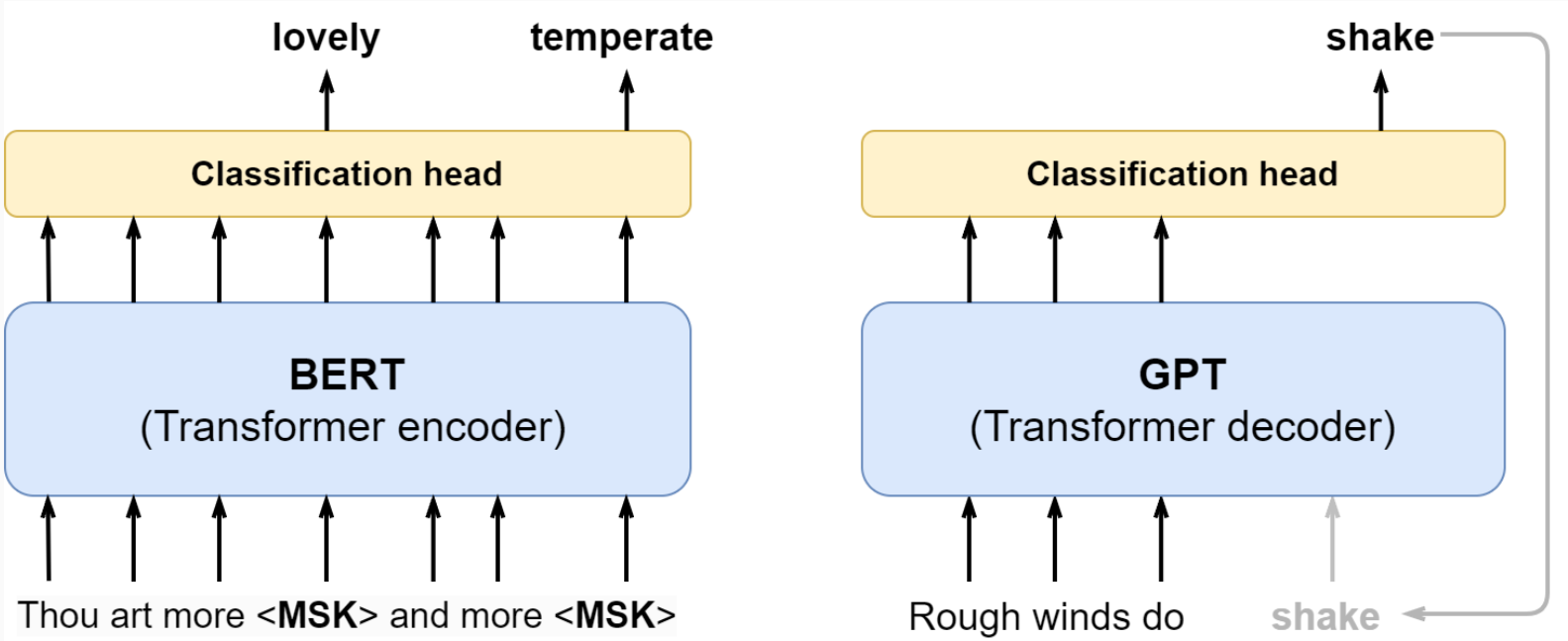
BERT и GPT

BERT

- (Devlin et al., 2018): BERT – Bidirectional Encoder Representations from Transformers
- Тот же Transformer, но теперь только энкодерная часть, и он двунаправленный, с левым и правым контекстом во всех слоях.
- То есть мы обучаем ту же языковую модель, но со словами из обоих направлений и предыдущих слоёв...
- ...подождите, так не должно работать. Как мы его обучаем?

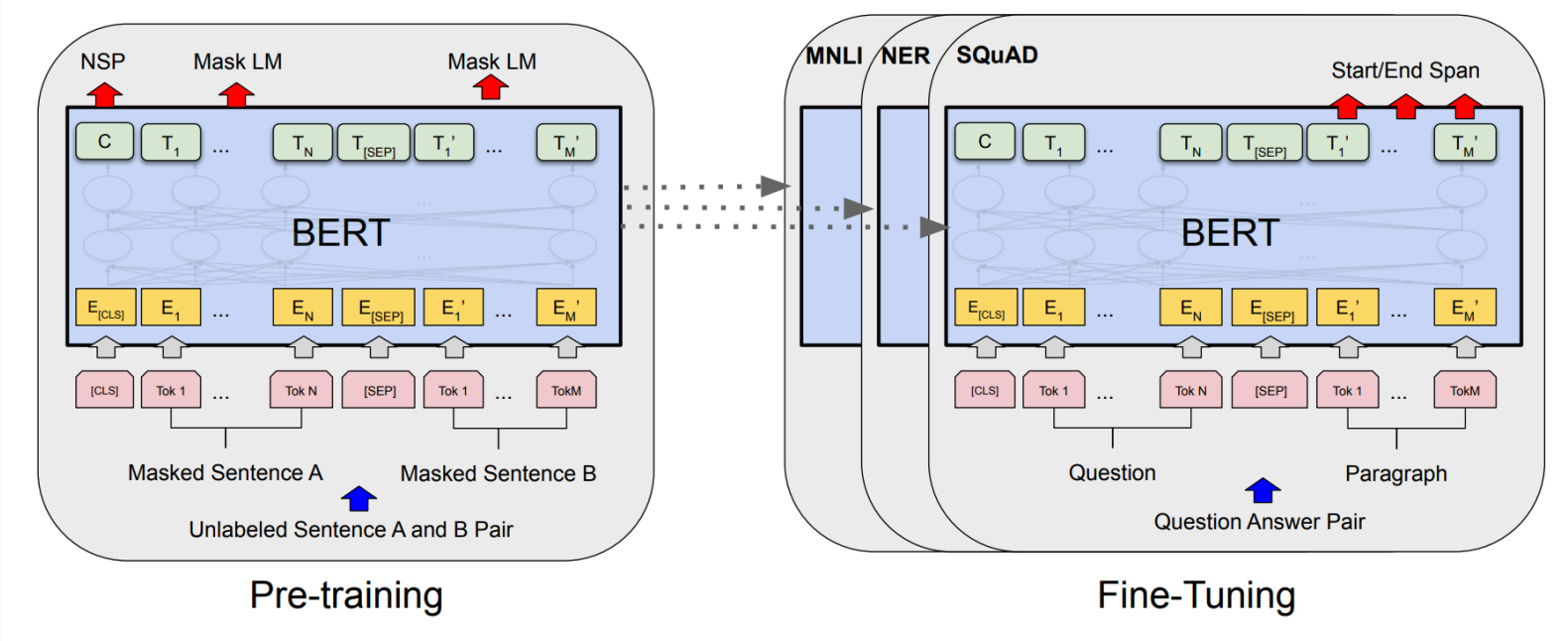
BERT

- Вместо обычной языковой модели BERT маскирует случайные слова и пытается их предсказать.



BERT

- Вторая задача для предобучения – предсказание следующего предложения.



BERT

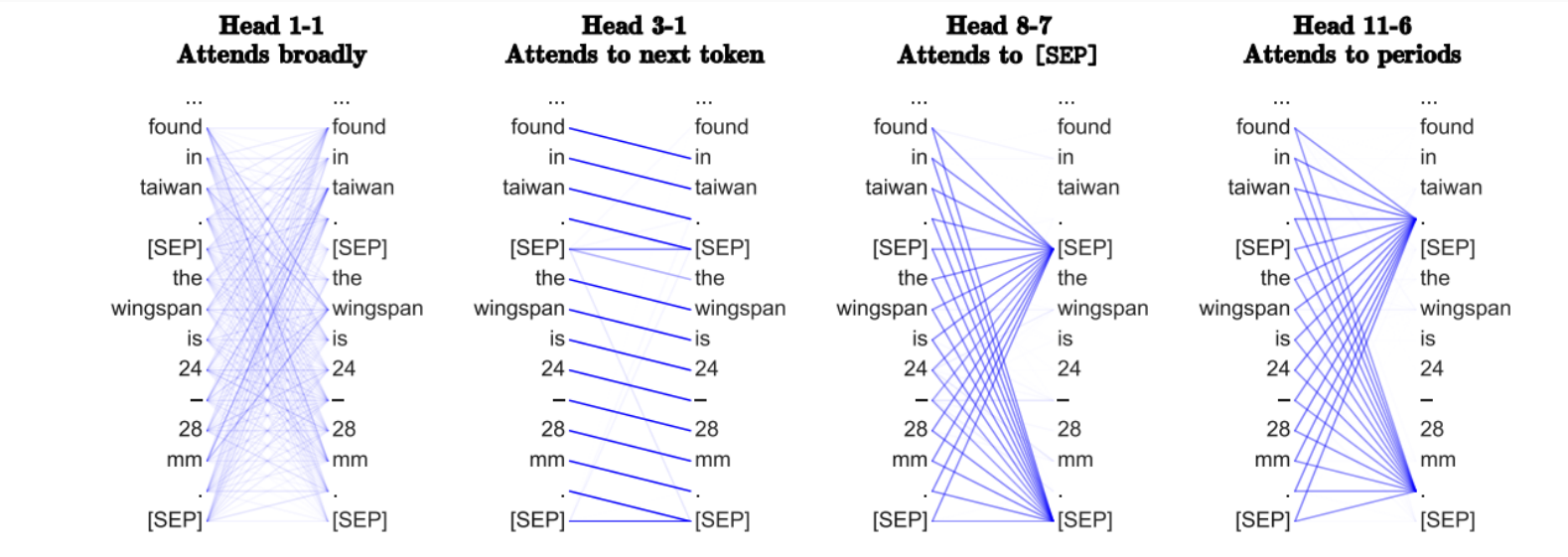
- И это всё, в остальном это просто энкодер Transformer. Опять же, BERT улучшил state of the art для многих задач:

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

- BERT-подобные энкодеры используются для классификации, поиска и получения контекстных эмбеддингов.

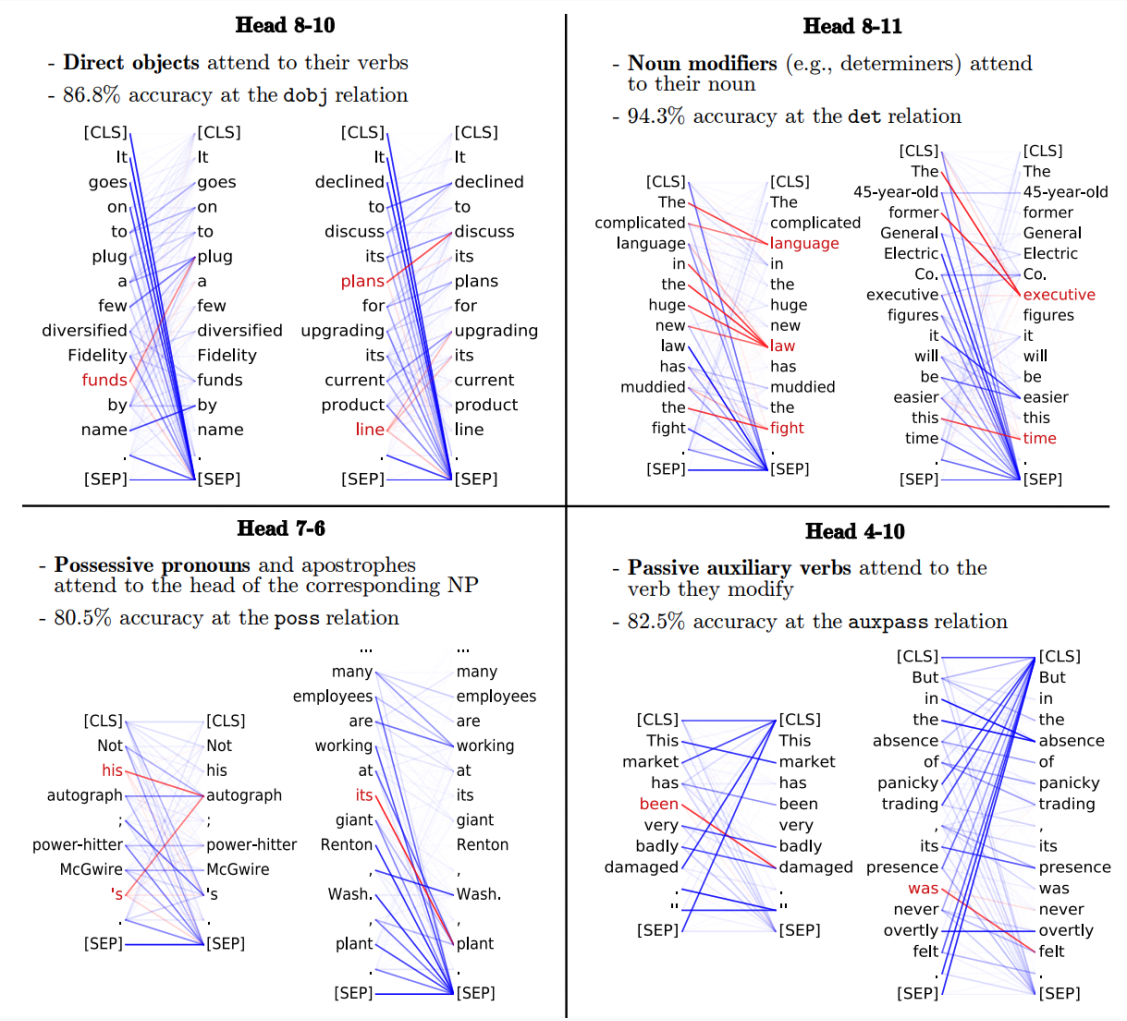
BERT

- Важные детали BERT:
  - wordpiece-эмбеддинги: используем фиксированный словарь подслов размера 30К, условно разбиваем слова на части: electrodynamics → electro ##dynamics
  - можно найти «головы», которые обращают внимание на разные вещи:



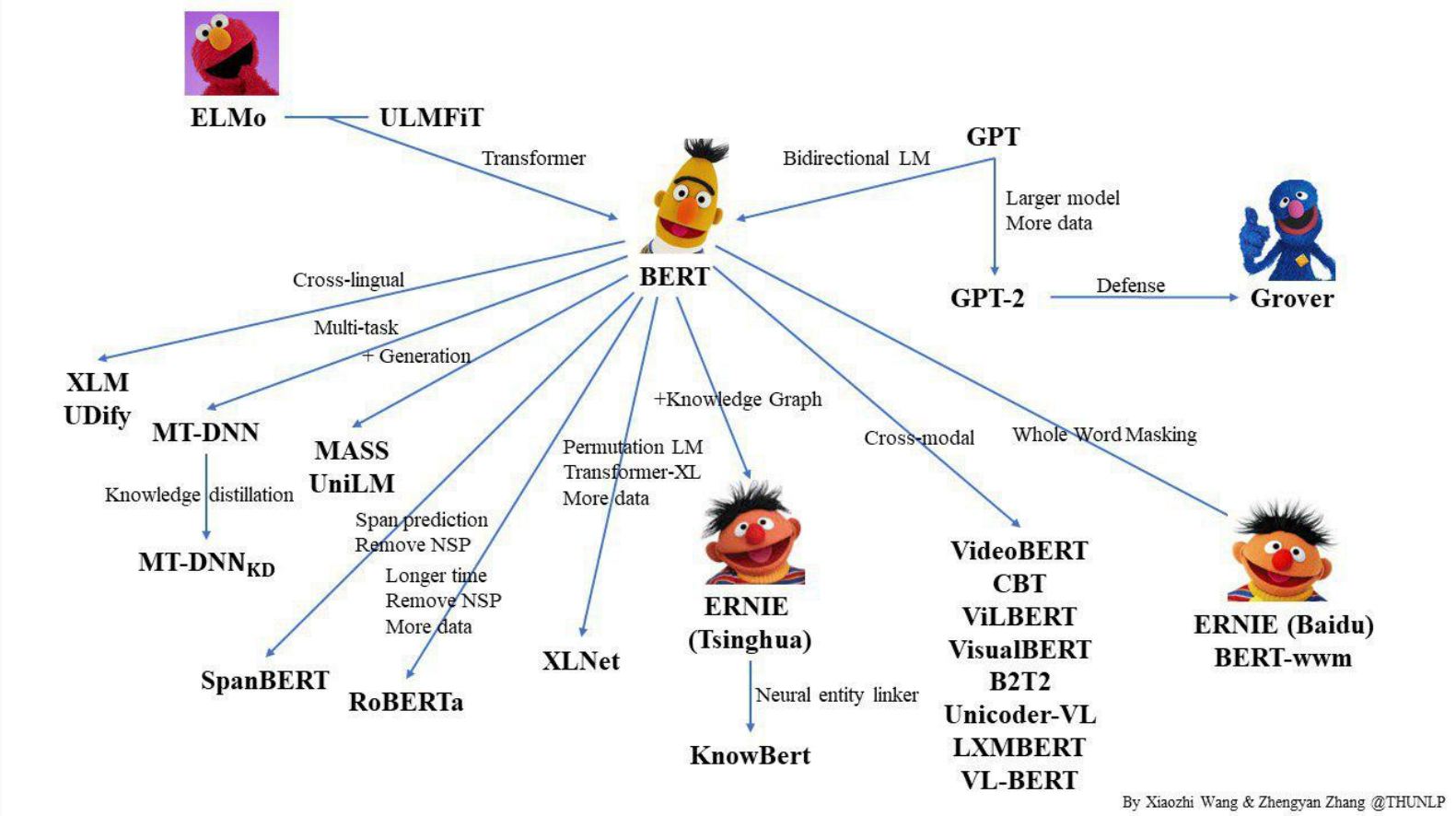
BERT

- Ещё более грамматически (Clark et al., 2019):



BERT

- Существует множество BERT-подобных моделей; это картинка из 2020 года:

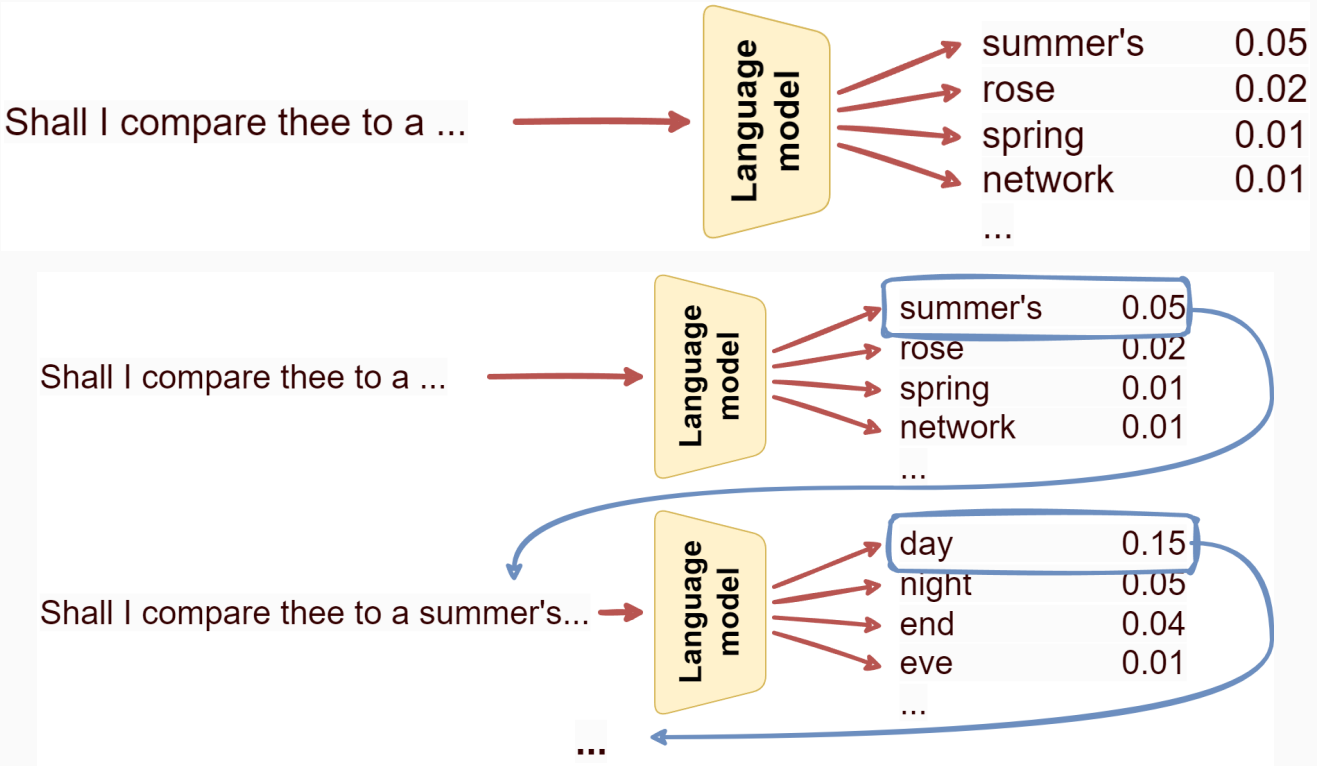


Семейство BERT

- RoBERTa (Liu et al., 2019): больше данных и обучения, динамическое маскирование; отказ от next sentence prediction.
- ALBERT (Lan et al., 2020): факторизация эмбеддингов и общие параметры между слоями.
- DistilBERT (Sanh et al., 2019): маленький энкодер учится воспроизводить поведение большого.
- BART (Lewis et al., 2019): уже энкодер–декодер; восстанавливает испорченный текст и подходит для задач генерации.

GPT

- В прошлый раз мы говорили об оригинальном Transformer
- Следующий шаг – OpenAI GPT (Generative Pretrained Transformer) (Radford et al., 2018)
- *Языковое моделирование*:



GPT

- Мы используем Transformer следующим образом:
  - сначала обучаем обычную языковую модель

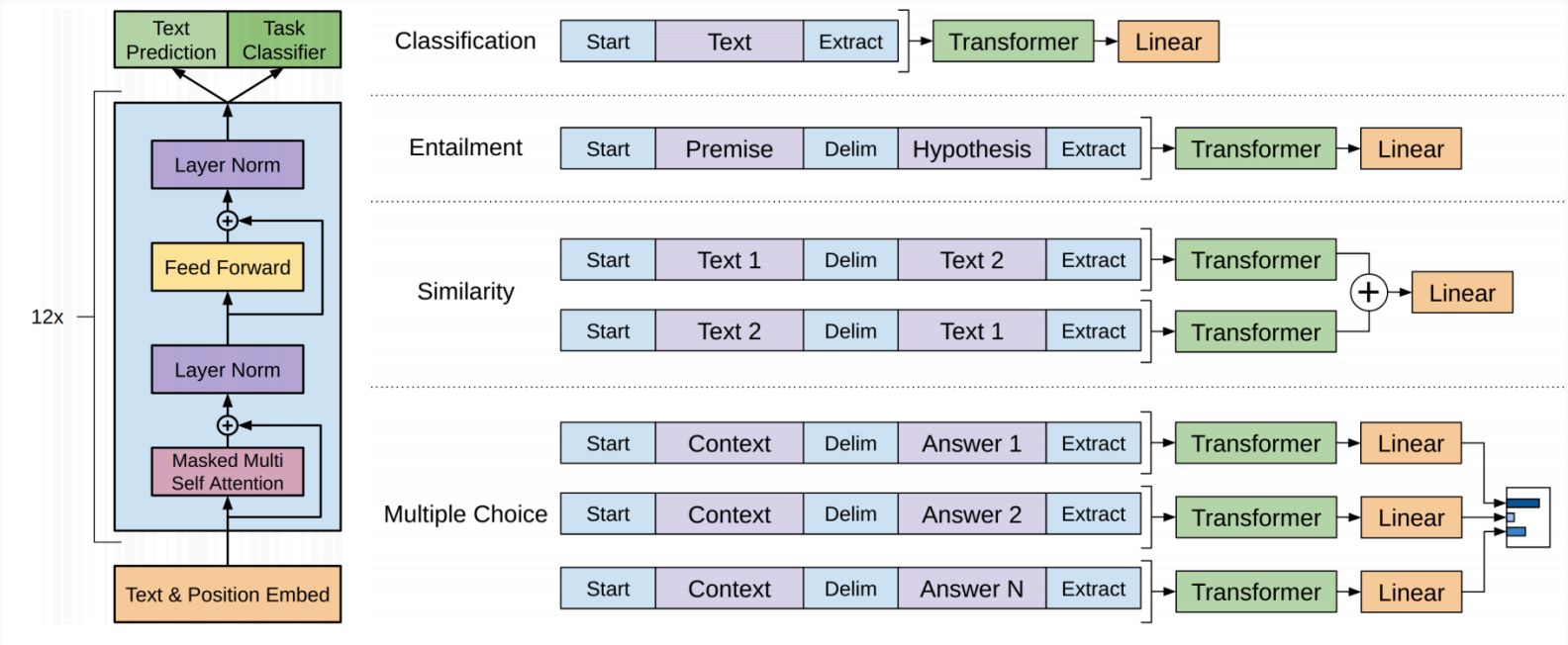
$$L_1(D) = \sum_i \log p(u_i \mid u_{i-k}, \dots, u_{i-1}; \theta);$$

  - затем добавляем новый линейный слой для каждой задачи и делаем дообучение с учителем:

$$L(C, D) = \sum_{(\mathbf{x}, y)} \log p(y \mid \mathbf{x}) + \lambda L_1(D).$$

GPT

- Идея в том, чтобы переиспользовать одну и ту же модель для множества разных задач:



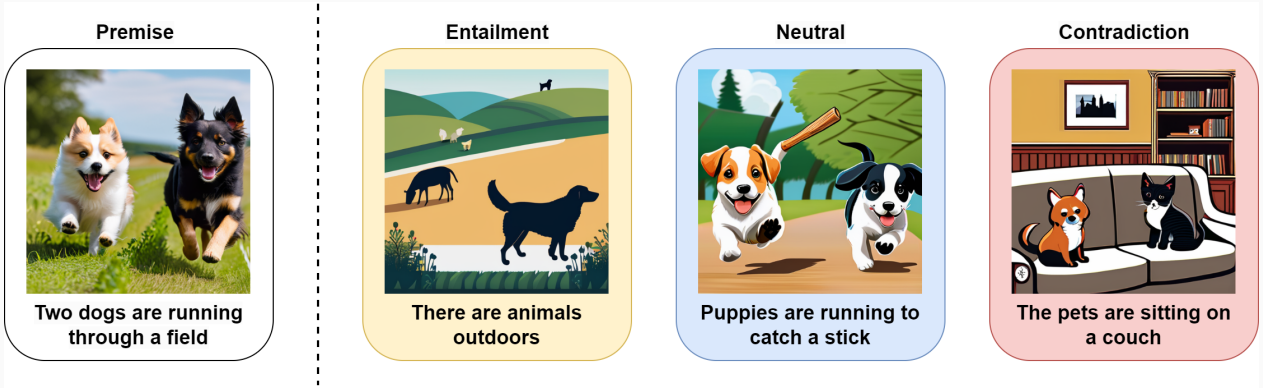
- GPT установил state of the art для многих задач NLP.

GPT-2: продолжать текст без дообучения

- Следующий шаг – GPT-2 (Radford et al., 2019). Тот же GPT, но:
  - значительно больше: 1.5B параметров (в GPT-1 было 117M, в BERT-Large – 340M);
  - обучен на огромном датасете: WebText – тексты страниц по ссылкам с Reddit с кармой ≥ 3, 40GB текста;
  - без дообучения и без учителя, тестирование качества в zero-shot режиме.

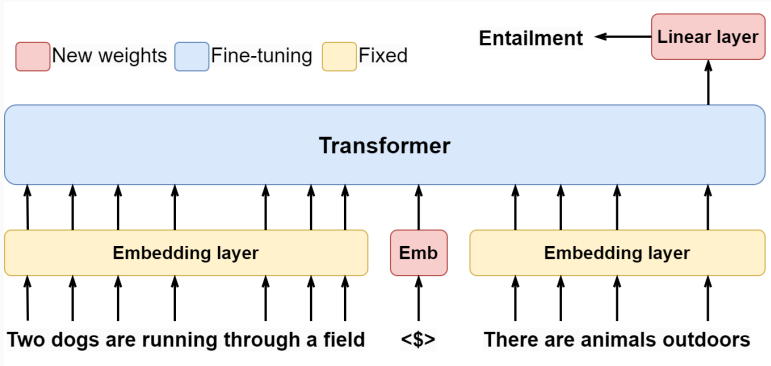
GPT-2: продолжать текст без дообучения

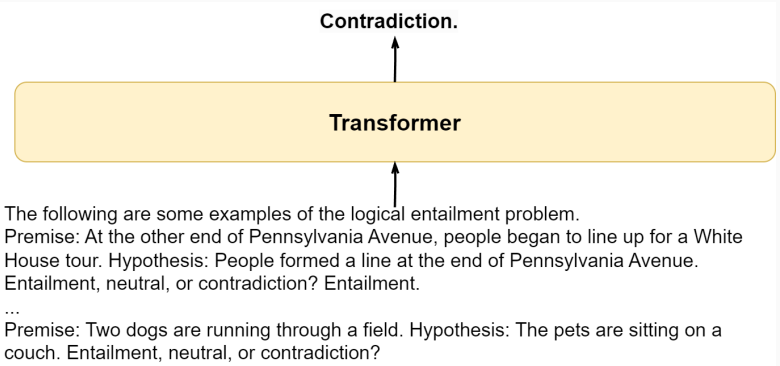
- В качестве примера применения рассмотрим задачу логического следования; на картинке
  - «*На улице есть животные*» получает метку «*Следование*», то есть вытекает из посылки,
  - «*Щенки бегут, чтобы поймать палку*» помечено как «*Нейтрально*», поскольку прямого противоречия нет, но собаки могут быть или не быть щенками, и палки тоже может не быть,
  - а «*Питомцы сидят на диване*» – явное «*Противоречие*», то есть посылка это исключает.



GPT-2: продолжать текст без дообучения

- Вот как оригинальный GPT решал задачу следования по сравнению с GPT-2 и далее:





GPT-2: продолжать текст без дообучения

- Результаты:

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)	PTB (PPL)	enwik8 (BPB)	text8 (BPC)	WikiText103 (PPL)	1BW (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14	46.54	0.99	1.08	18.3	**21.8**
117M	**35.13**	45.99	**87.65**	**83.4**	**29.41**	65.85	1.16	1.17	37.50	75.20
345M	**15.60**	55.48	**92.35**	**87.1**	**22.76**	47.33	1.01	**1.06**	26.37	55.72
762M	**10.87**	**60.12**	**93.45**	**88.0**	**19.93**	**40.31**	**0.97**	**1.02**	22.05	44.575
1542M	**8.63**	**63.24**	**93.30**	**89.05**	**18.34**	**35.76**	**0.93**	**0.98**	**17.48**	42.16

- Это zero-shot примеры из маленького контекста!
- Источник: Radford et al., *Language Models are Unsupervised Multitask Learners* (2019).

Семейство GPT: масштабирование

- GPT-1 (2018): 12 слоёв, 117M параметров, предобучение на BookCorpus и дообучение под конкретные задачи.
- GPT-2 (2019): до 1.5B параметров, WebText; zero-shot постановка — задачу задаём контекстом.
- GPT-3 (2020): до 175B параметров; few-shot learning через примеры в контексте без обновления весов.
- Целевая функция остаётся простой: предсказывать следующий токен по предыдущим.

BERT и GPT: чему учим трансформер?

- BERT: двунаправленный энкодер, восстановление маскированных токенов. Удобен для понимания и представления текста.
- GPT: причинная маска, предсказание следующего токена. Удобен для последовательной генерации.
- Архитектура внимания и целевая функция определяют, какую информацию можно использовать.
- А что, если токенами будут не части слов, а части картинки?

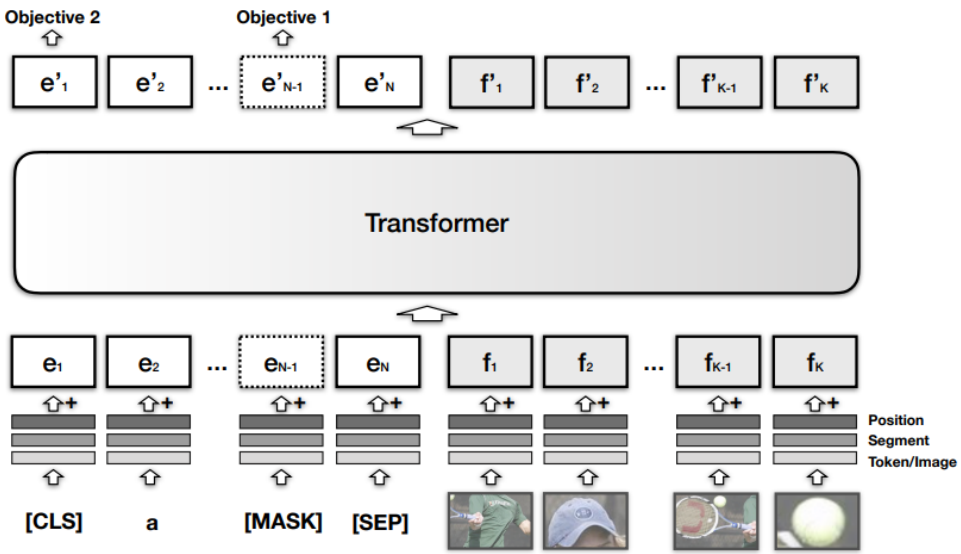
Vision Transformers

Visual BERT

- Картинки — это же не последовательности? Ну да, но...
- Visual BERT: давайте вместе моделировать картинки и подписи к ним
- Используем предобученный object detection вроде Faster R-CNN, вырезаем объекты, строим их вложения через CNN и positional embeddings, подаём в единый трансформер с подписью

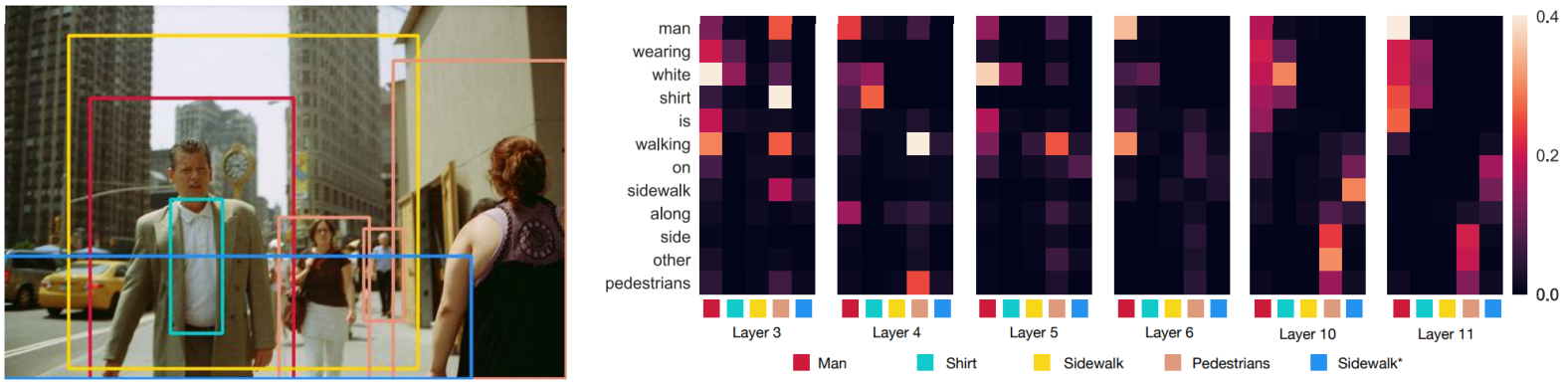


A person hits a ball with a tennis racket

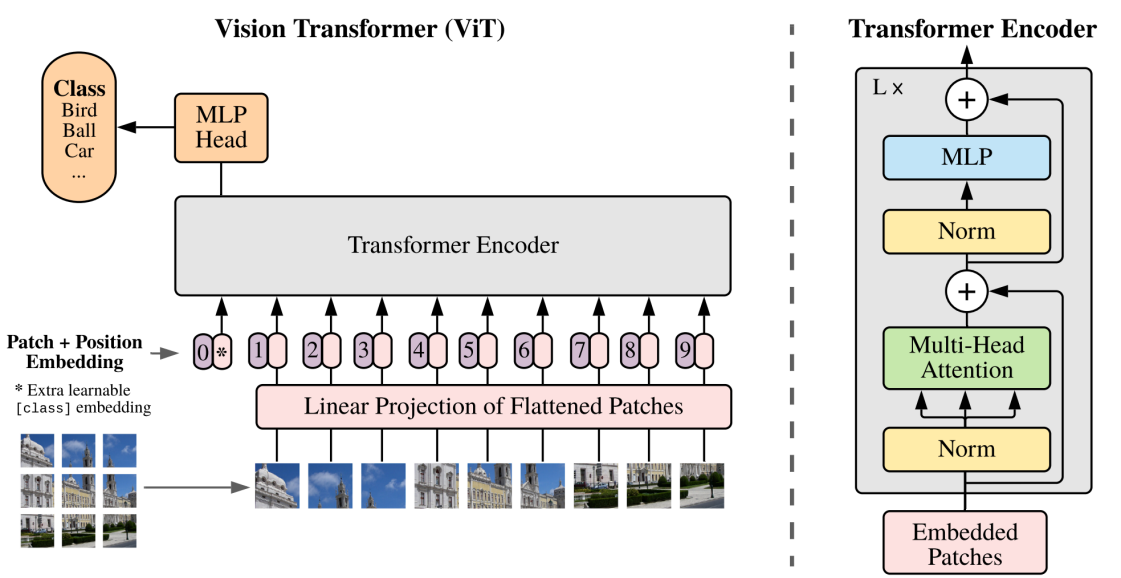


Visual BERT

- Предобучение: masked language modeling + sentence-image prediction (подходит ли подпись к рисунку?)
- Примеры голов внимания показывают, как слова из подписи «смотрят» на соответствующие объекты

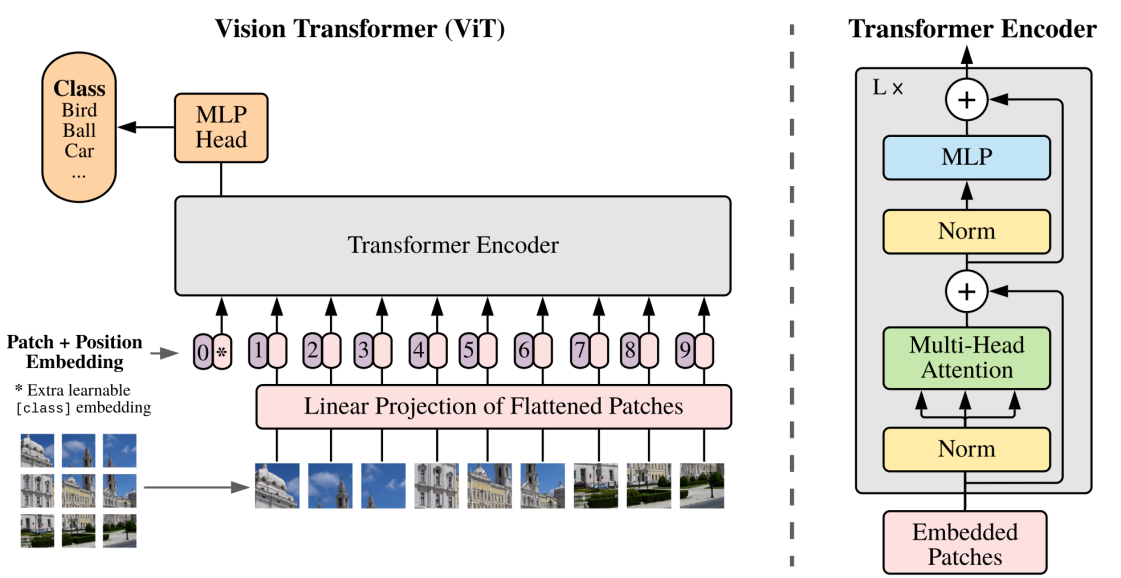


Vision Transformer (ViT)



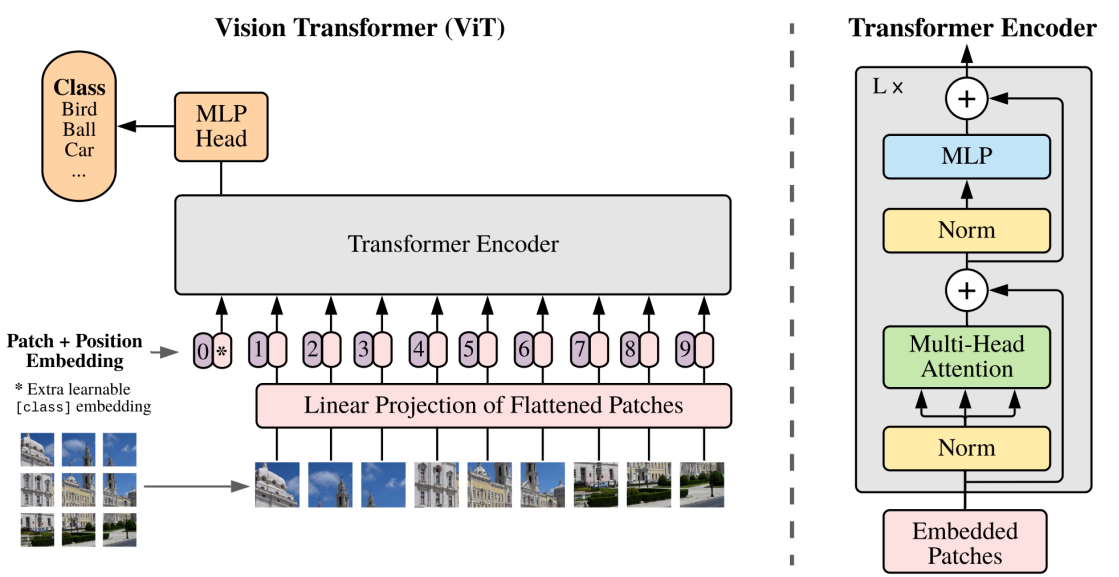
- Классическая Transformer-архитектура для картинок: *Vision Transformer* (ViT; Dosovitsky et al., 2020)
- ViT — прямолинейная модификация BERT, но теперь на входе токены из изображения, а не текста

Vision Transformer (ViT)



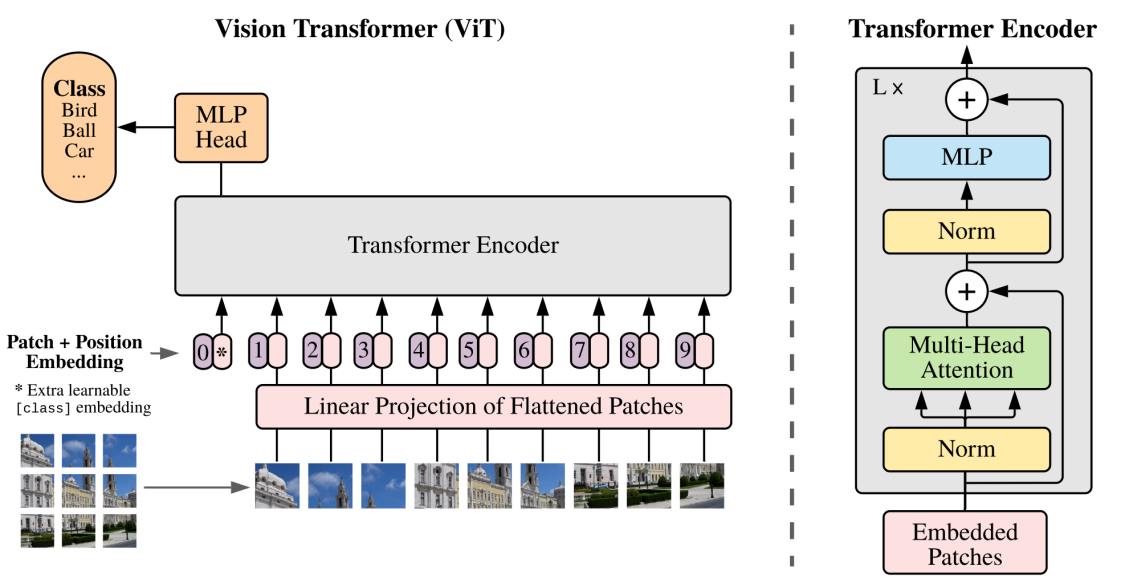
- Вход режется на маленькие патчи: $H \times W$ картинка с C каналами $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ превращается в последовательность патчей $\mathbf{x}_p \in \mathbb{R}^{N \times (P^2 C)}$, где $N = HW/P^2$ — число $P \times P$ патчей

Vision Transformer (ViT)



- Патчи превращаются во вложения через обычную линейную проекцию, последовательность векторов подаётся в кодировщик трансформера
- В основном эксперименте ViT предобучается с учителем на классификации; итоговый [CLS]-токен подаётся в классификатор

Vision Transformer (ViT)



- ViT использует обучаемые одномерные позиционные эмбеддинги.
- Dosovitsky et al. экспериментировали с 2D-кодировками, но никакого улучшения не нашли...

ViT: цена последовательности

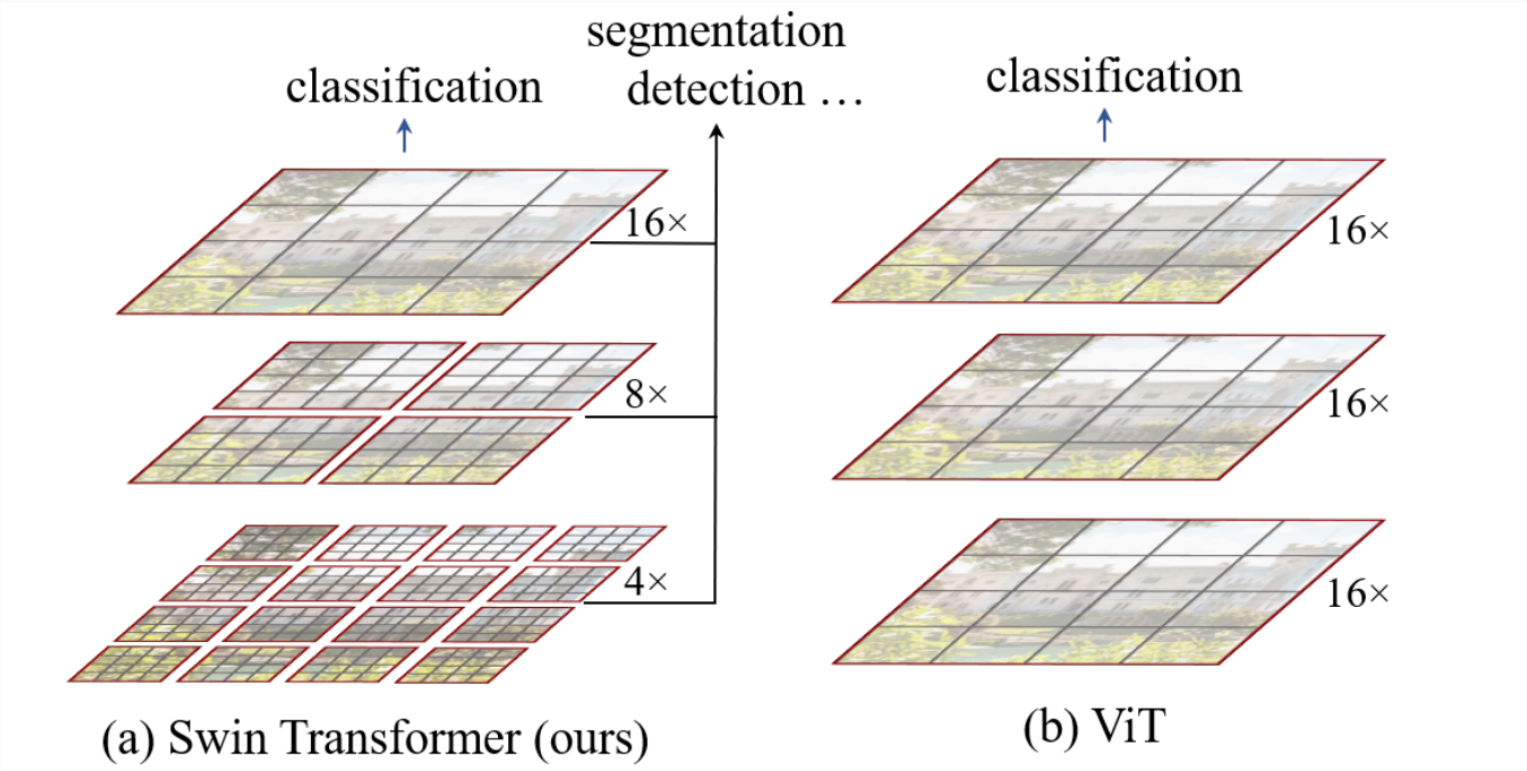
- Для картинки 224×224 и патча 16×16: $14^2 = 196$ патчей и один [CLS]-токен.
- Каждый RGB-патч — вектор из $16^2 \cdot 3 = 768$ чисел; общая линейная проекция превращает его в эмбеддинг.
- Если удвоить обе стороны изображения, токенов станет в 4 раза больше, а пар в полном внимании — в 16 раз.
- В исходном ViT меньше встроенных предположений о локальности и переносах, чем у CNN: особенно важен масштаб данных.

Предобучение на изображениях

- Оригинальный ViT: в основном классификация с учителем на большом датасете, затем перенос на целевую задачу.
- BERT-подобное маскирование — отдельная линия: BEiT предсказывает дискретные визуальные токены, MAE восстанавливает пиксели скрытых патчей.
- Не следует смешивать архитектуру ViT с конкретной задачей предобучения.

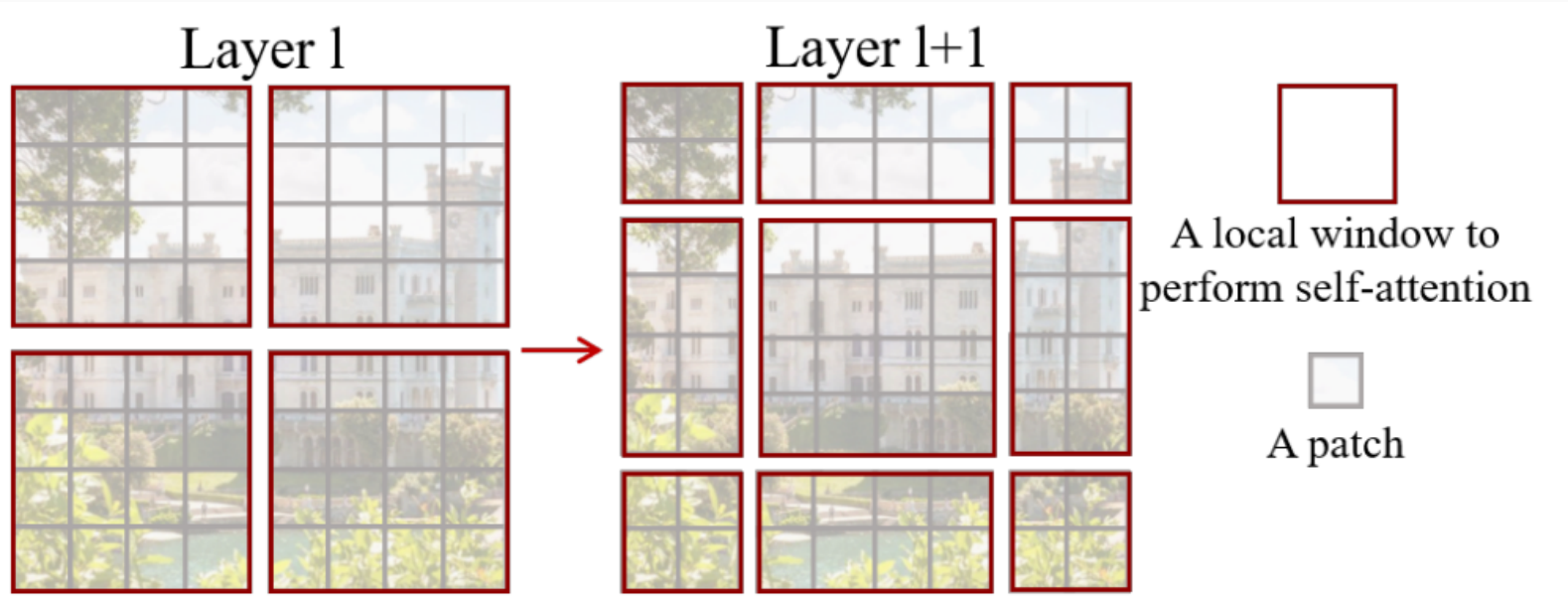
Swin Transformers

- *Swin Transformer* (Liu et al., 2021; **s**hifted **win**dows): похож на ViT, но иерархический



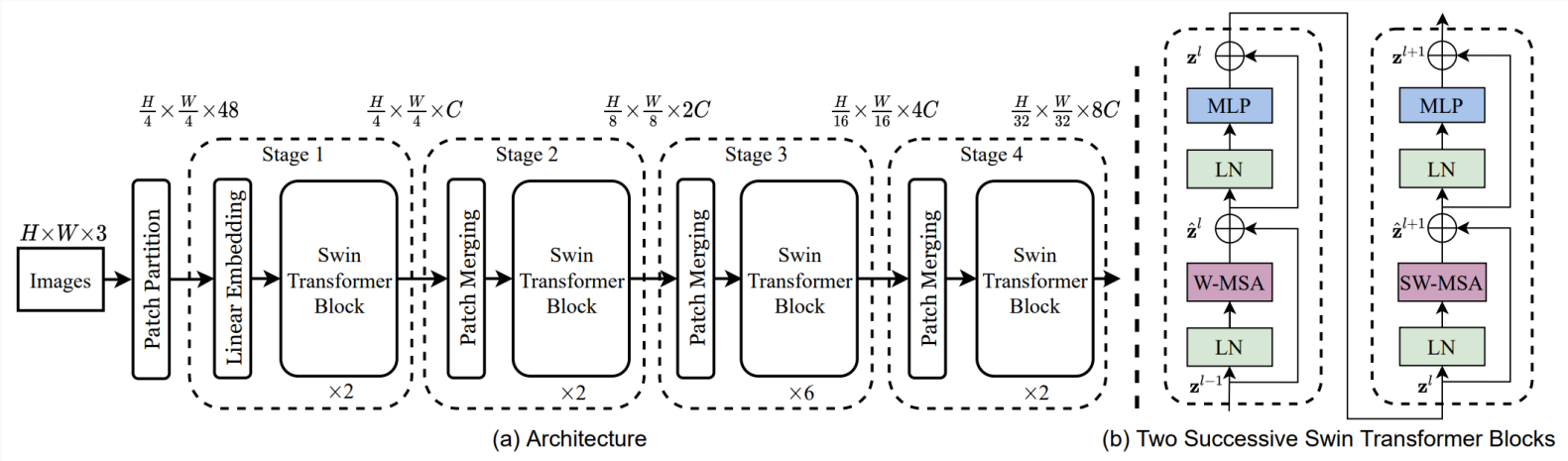
Swin Transformers

- Обрабатывает патчи на нескольких масштабах, вычисляя самовнимание по патчам с CNN-подобной архитектурой
- Окна сдвигаются между слоями, и это даёт связь между частями изображения



Swin Transformers

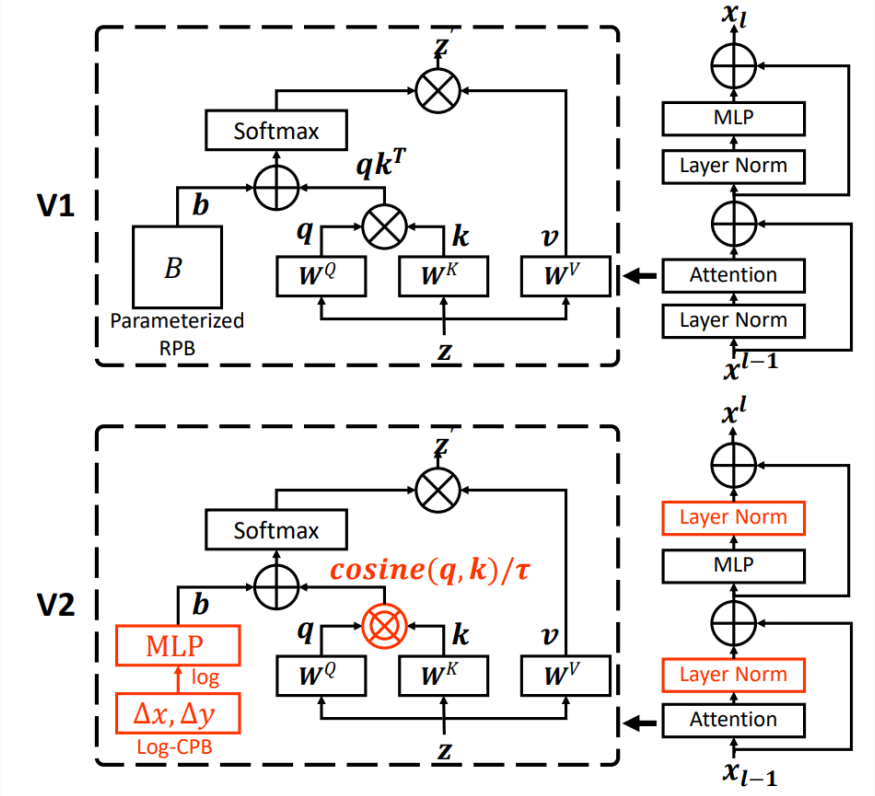
- Сама архитектура сокращает геометрию, как классические CNN backbones:



- Swin может масштабироваться до высоких разрешений и может использоваться для задач распознавания объектов и сегментации

Swin Transformers

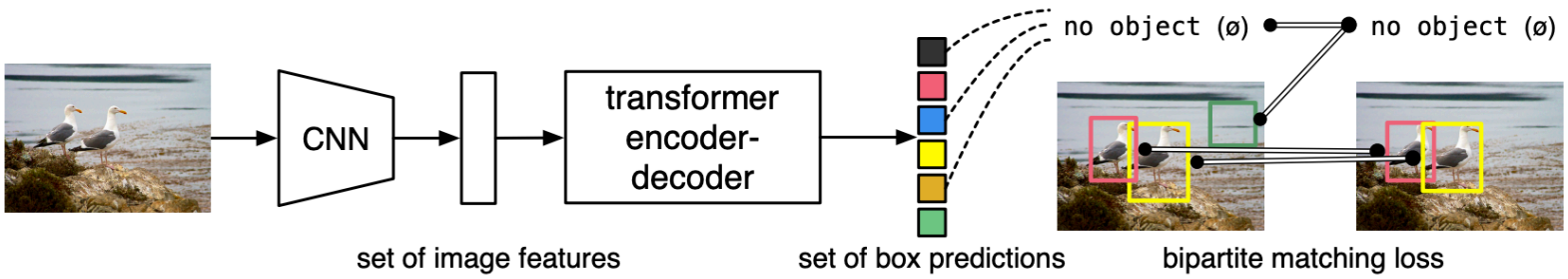
- Потом вышел *Swin Transformer v2* (Liu et al., 2022) — это Swin Transformer, отмашстабированный до 3B параметров, на картинках до 1536×1536 пикселей, что ещё сильнее улучшает обработку изображений во всех задачах



Распознавание объектов с трансформерами

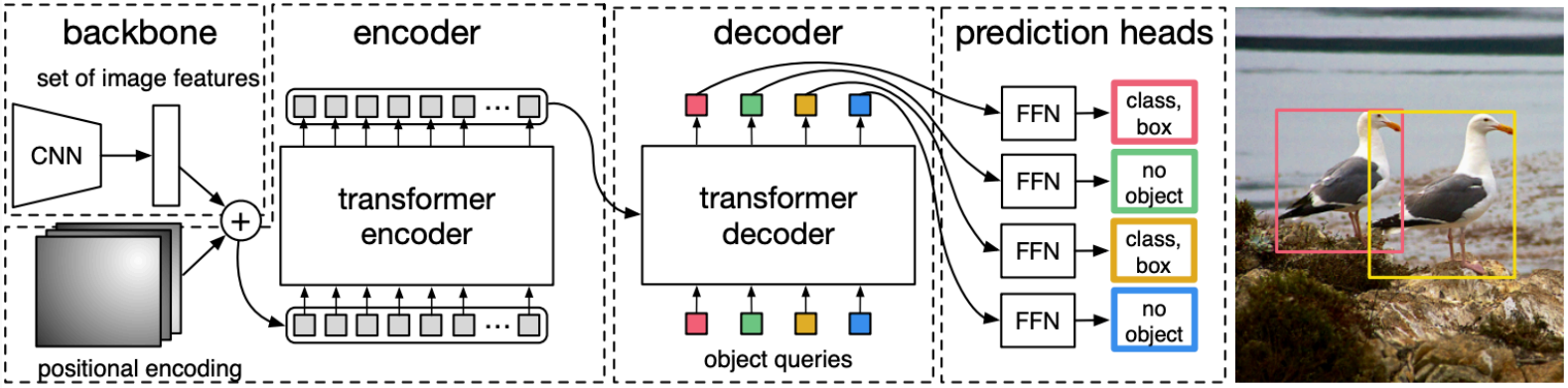
DETR и аналоги

- DETR (DEtection TRansformer; Carion et al., 2020) – прямая адаптация трансформеров
- Распознавание объектов — задача предсказания множества; DETR добавляет основанную на множествах глобальную функцию потерь, которая даёт уникальные предсказания через паросочетания, и архитектуру кодировщик-декодировщик на базе трансформера



DETR и аналоги

- DETR использует традиционный сверточный backbone (ResNet у Carion et al., но можно взять что угодно)
- Затем DETR выравнивает признаки, добавляет позиционное кодирование и перемешивает признаки с помощью кодировщика
- Декодировщик берёт на вход небольшое число *объектных запросов*, обучаемых позиционных вложений, с вниманием на выход кодировщика
- Выход проходит через маленькую feedforward сеть, которая предсказывает класс и bbox (или “нет объекта”)



DETR и аналоги

- Функция потерь происходит из паросочетания:
  - для множества предсказаний $\hat{\mathbf{y}} = \{\hat{y}_i\}_{i=1}^N$ и истинных объектов $\mathbf{y} = \{y_i\}_{i=1}^N$ (плюс $\emptyset$, т.к. N должно быть большим), ищем перестановку $\sigma_* = \arg\min_\sigma \sum_{i=1}^N \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$ с помощью венгерского алгоритма, где $\mathcal{L}_{\text{match}}$ – это стоимость сопоставления;
  - $\mathcal{L}_{\text{match}}$ объединяет точность предсказаний для $y_i = (c_i, b_i)$, где c_i – метка класса (возможно, $\emptyset$), а $b_i \in \mathbb{R}^4$ – bbox:

$$\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}) = -[c_i \neq \emptyset]\hat{p}_{\sigma(i)}(c_i) + [c_i \neq \emptyset]\mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)});$$

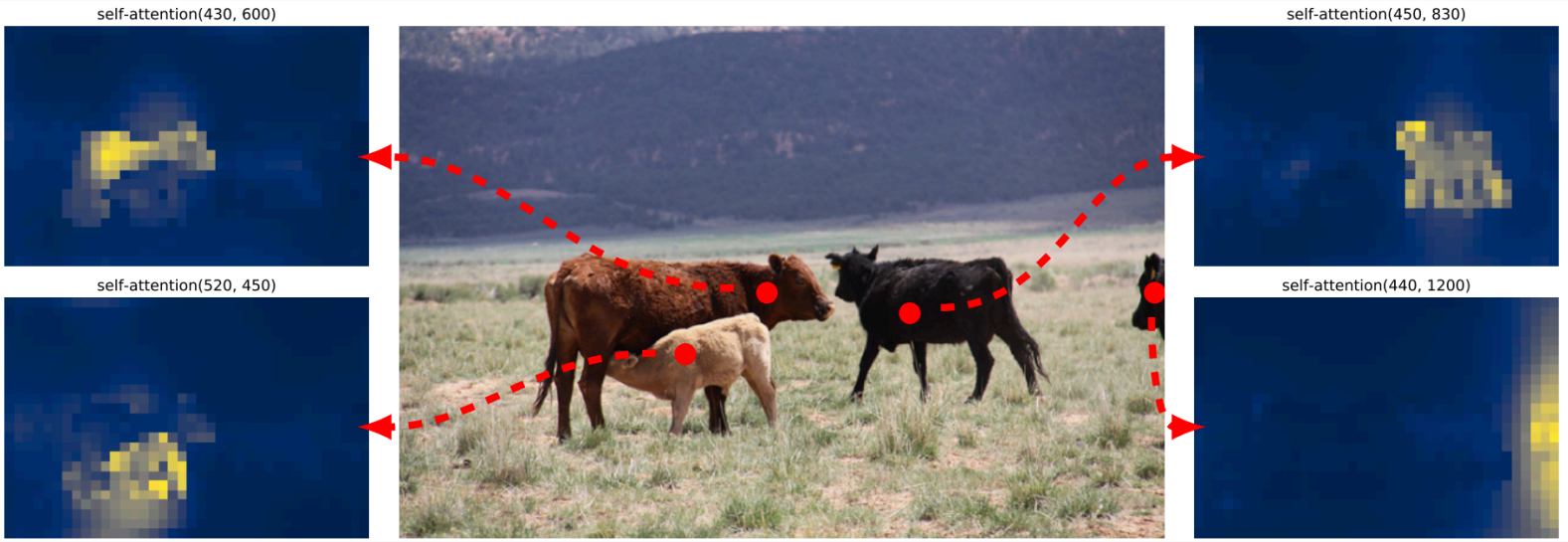
    здесь $\mathcal{L}_{\text{box}}$ – комбинация функций потерь от IoU и L_1;
  - после сопоставления фактическая функция потерь

$$\mathcal{L}_{\text{Hungarian}}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^N \left[-\log \hat{p}_{\sigma_*(i)}(c_i) + [c_i \neq \emptyset]\mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)})\right];$$

    теперь мы предсказываем и $c_i = \emptyset$, но на практике они получают меньший вес.

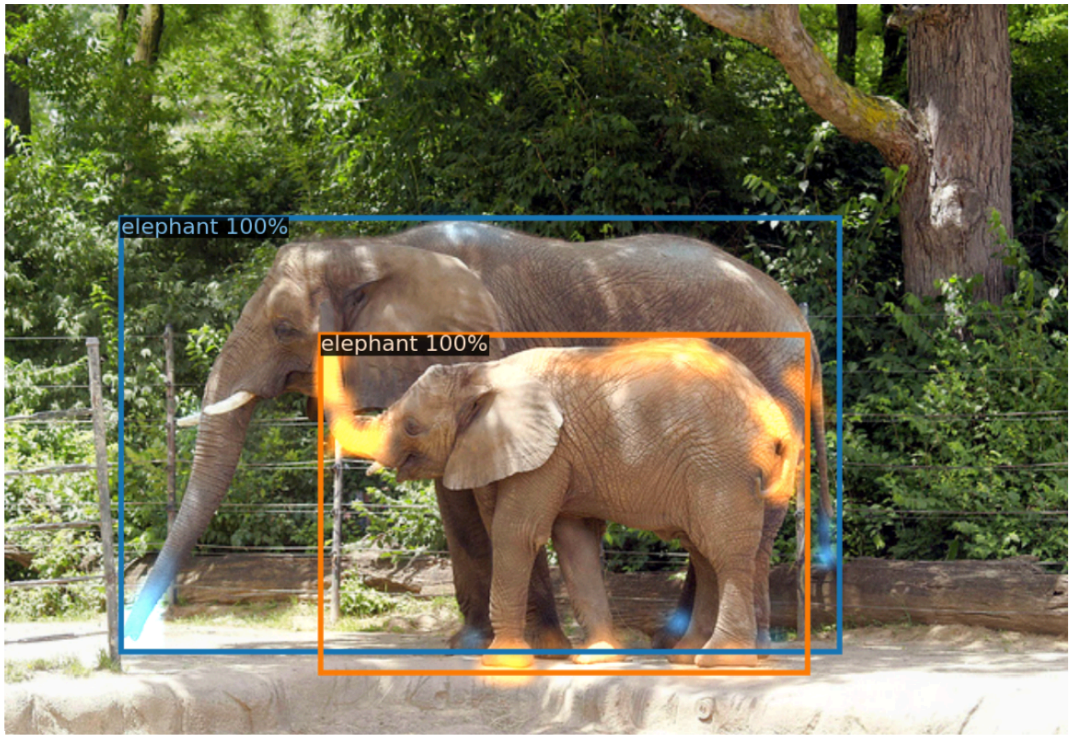
DETR и аналоги

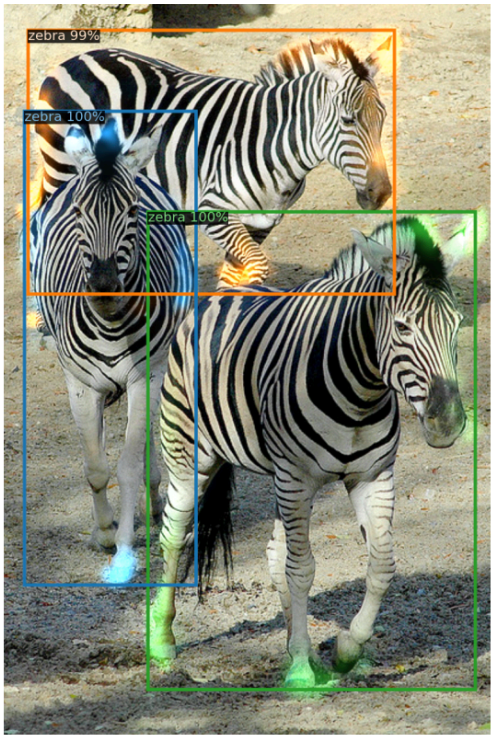
- Хорошие результаты, и DETR способен распределять экземпляры объектов по головам внимания:



DETR и аналоги

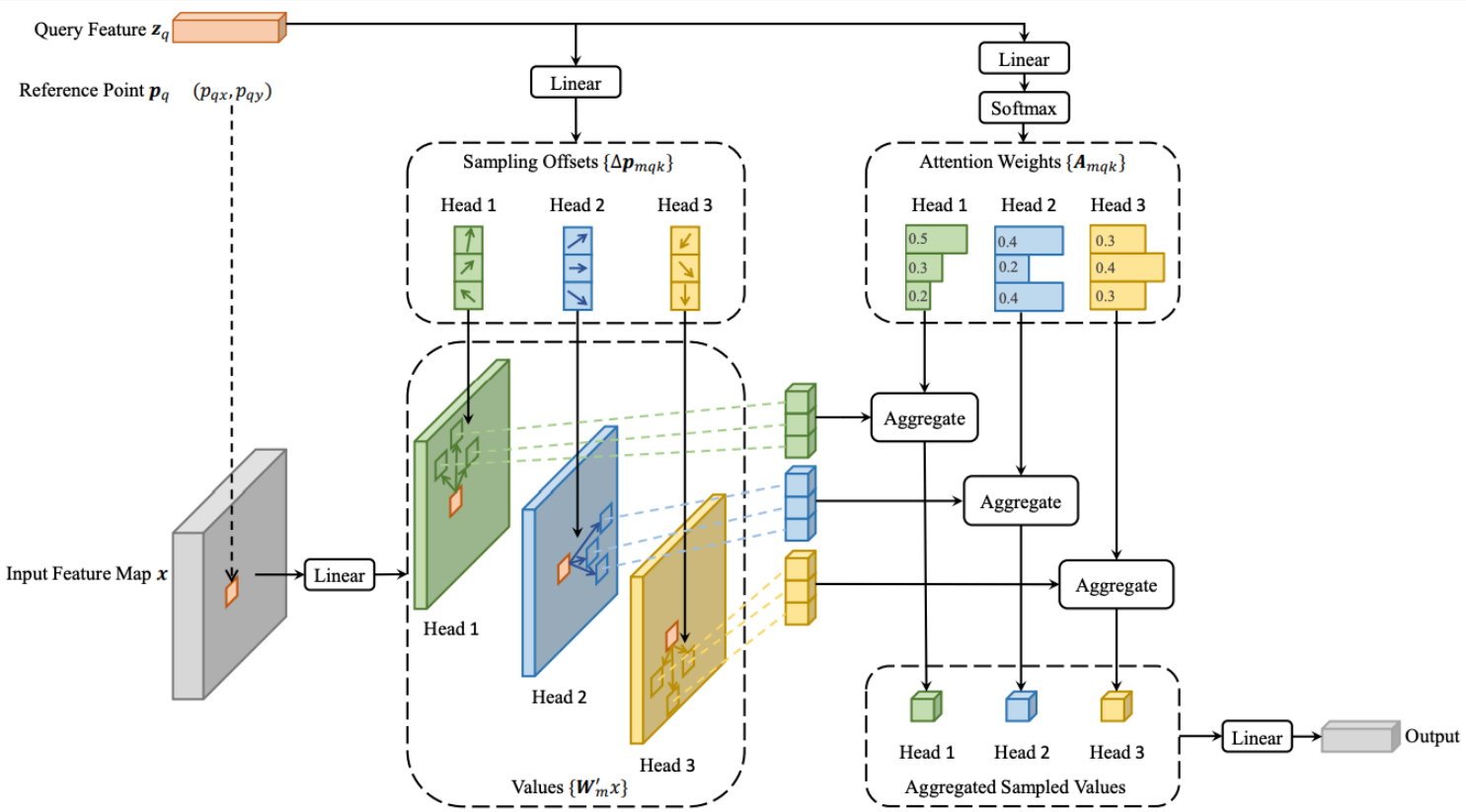
- Внимание декодера для каждого объекта локализовано на границах
- Скорее всего, DETR разделяет экземпляры глобальным вниманием в кодировщике, а декодеру остается только выяснить классы и границы объектов





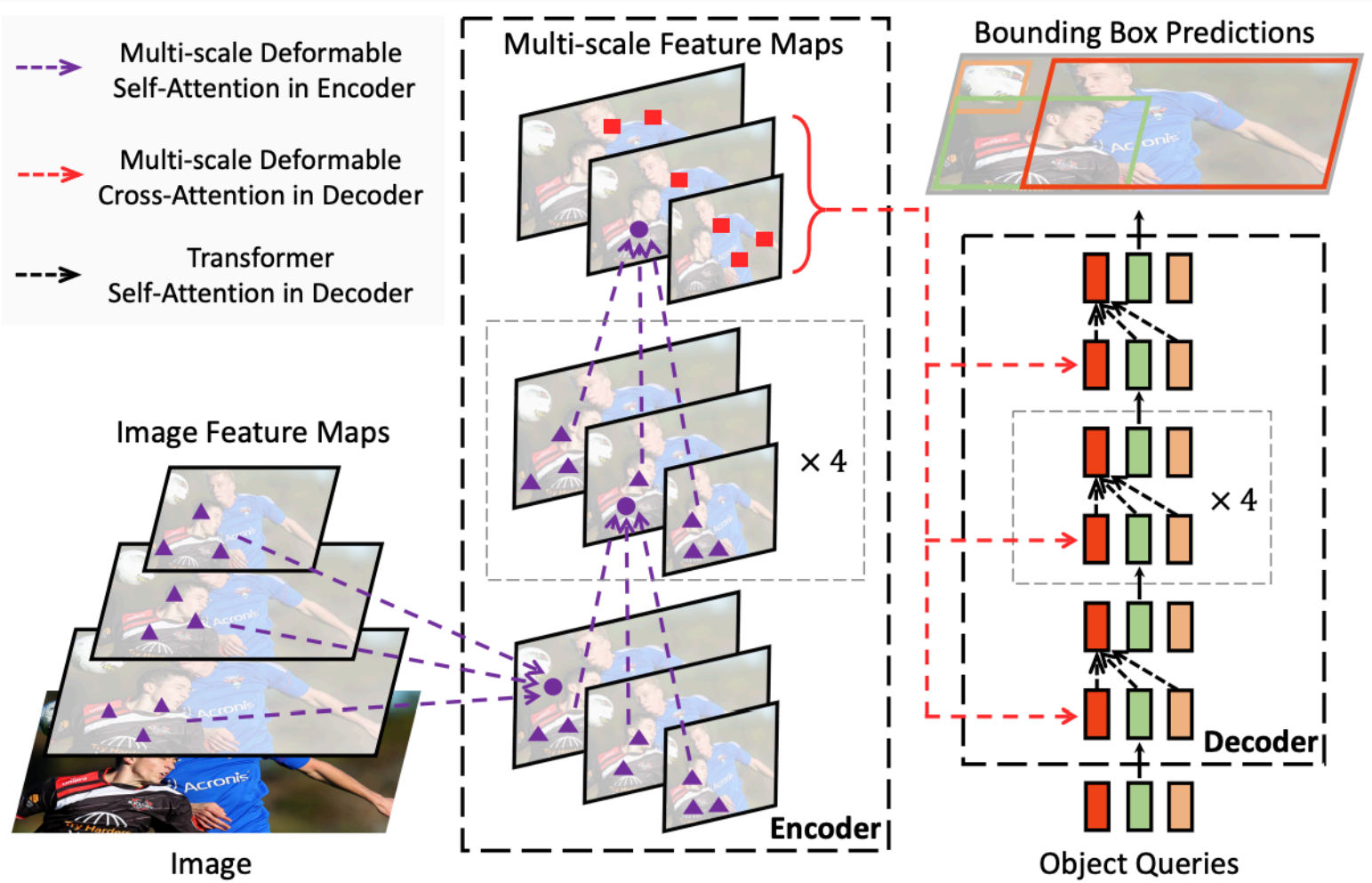
DETR и аналоги

- DETR работает хорошо, но не очень хорошо на маленьких объектах, а квадратичная сложность делает мультимасштабное распознавание очень сложной задачей
- Deformable DETR (Zhu et al., 2021): давайте используем деформируемое внимание и будем обращать внимание только на несколько пикселей



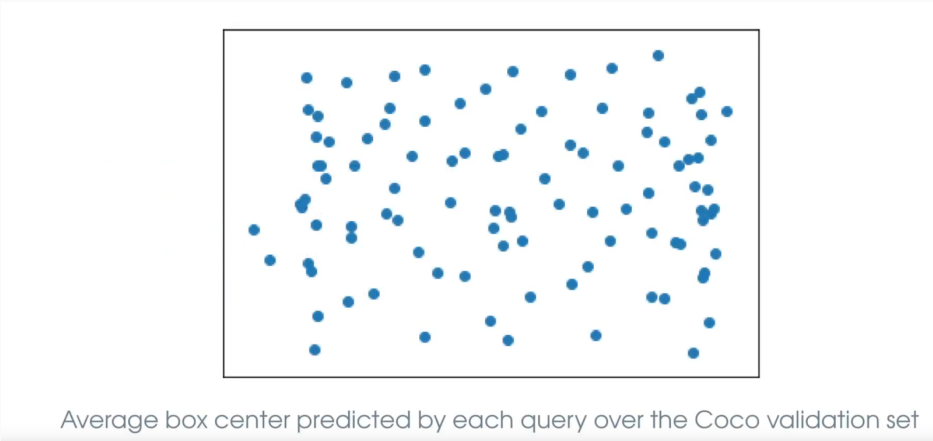
DETR и аналоги

- Общая архитектура:



DETR и аналоги

- Позиционные запросы DETR – это в основном гибкие обучаемые якоря для ограничивающих прямоугольников; они обучаются глобально и не зависят от текущего изображения

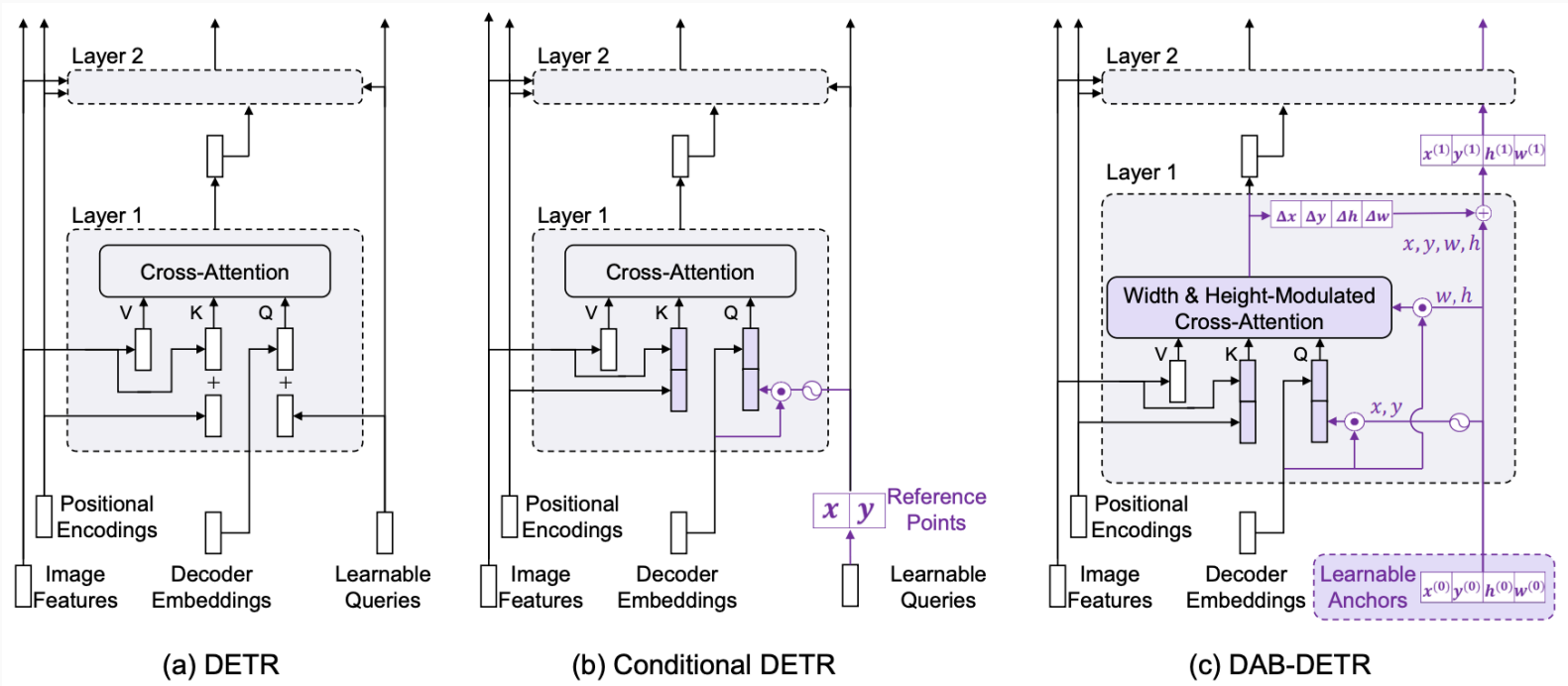


Average box center predicted by each query over the Coco validation set

- Обучение пространственных запросов на данных (без каких-либо фиксированных anchor boxes) – важная часть DETR, но DETR обучается очень, очень медленно, и пространственные запросы – большой фактор в этом
- Как мы можем это улучшить?

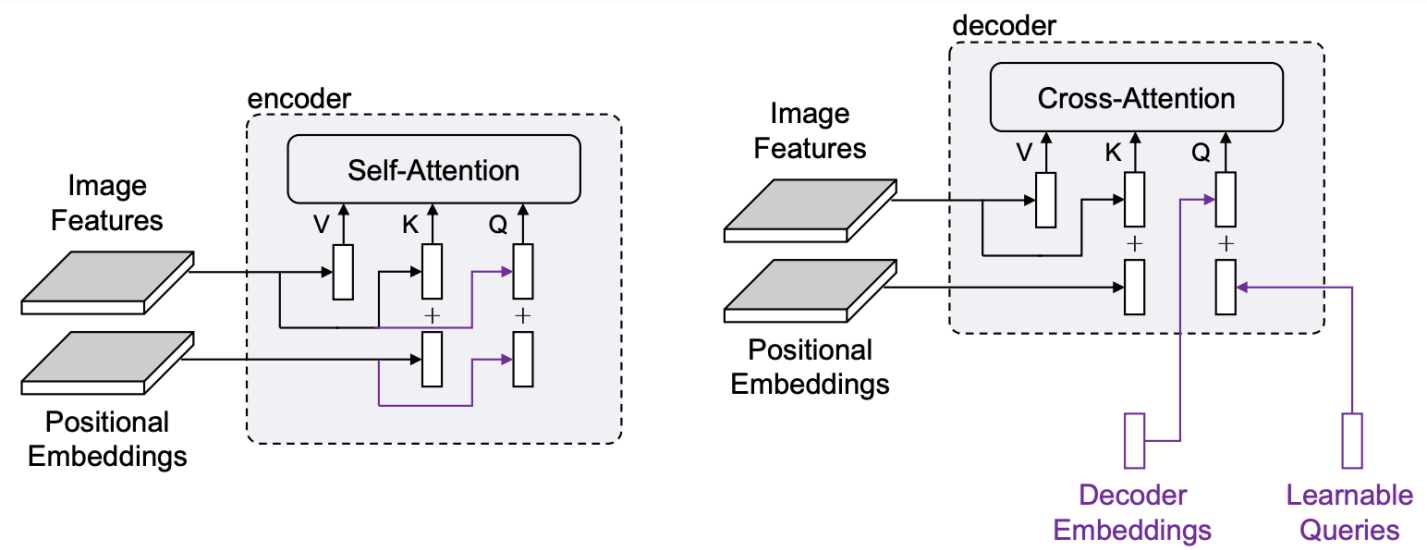
DETR и аналоги

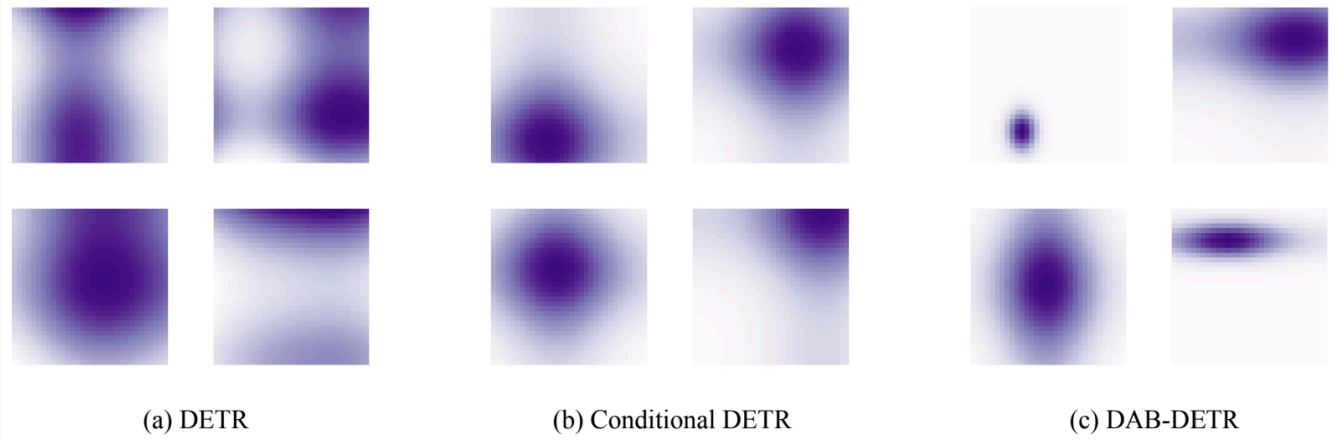
- DAB-DETR (Dynamic Anchor Box DETR; Liu et al., 2022): давайте вернемся к идее anchor boxes
- Обучаемые якоря постепенно уточняются от слоя к слою в декодере, и теперь они имеют определенную семантику:



DETR и аналоги

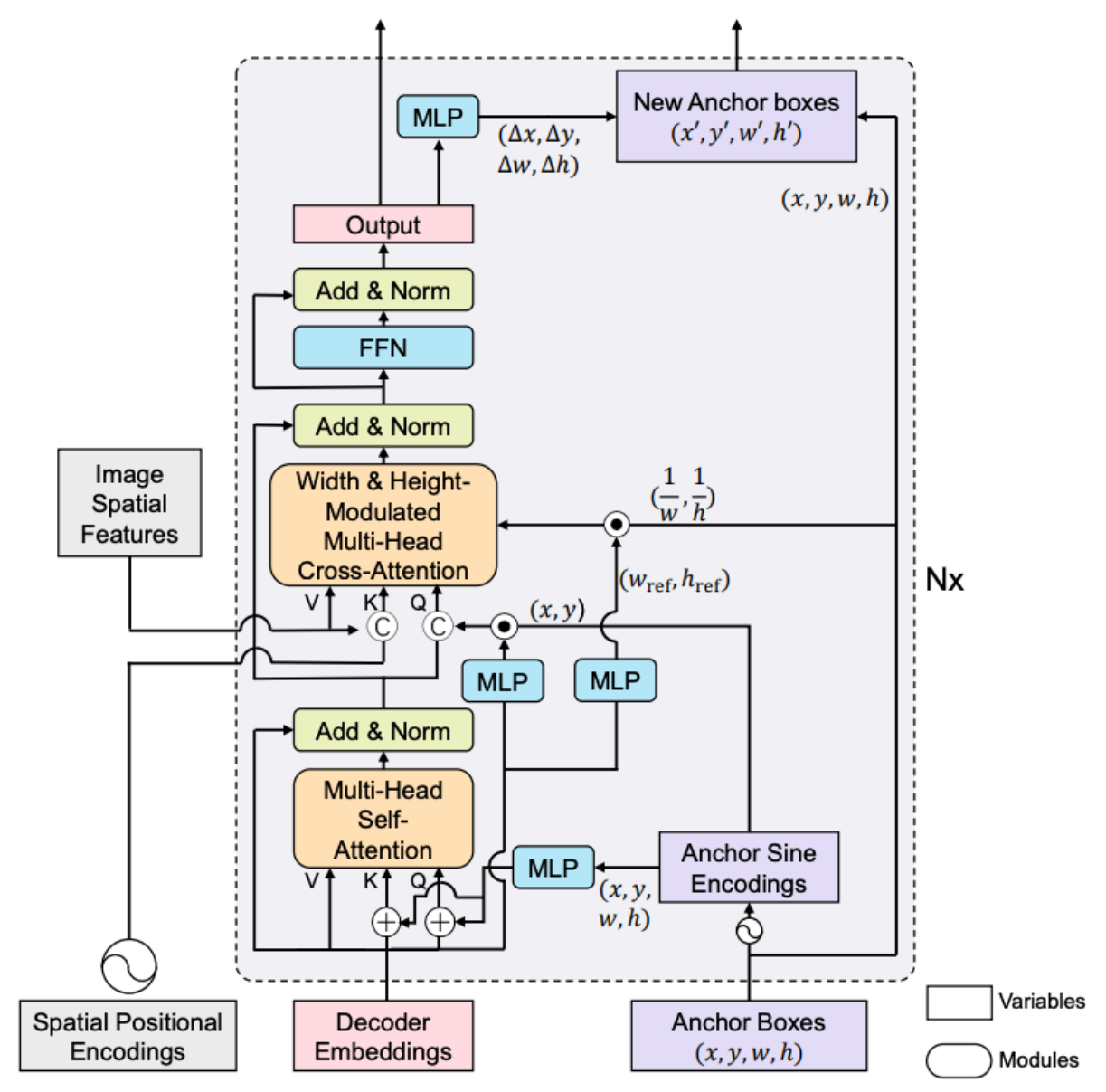
- Ширина и высота anchor box используются для модуляции перекрестного внимания, поэтому внимание становится гораздо лучше локализованным, и это помогает ускорить обучение





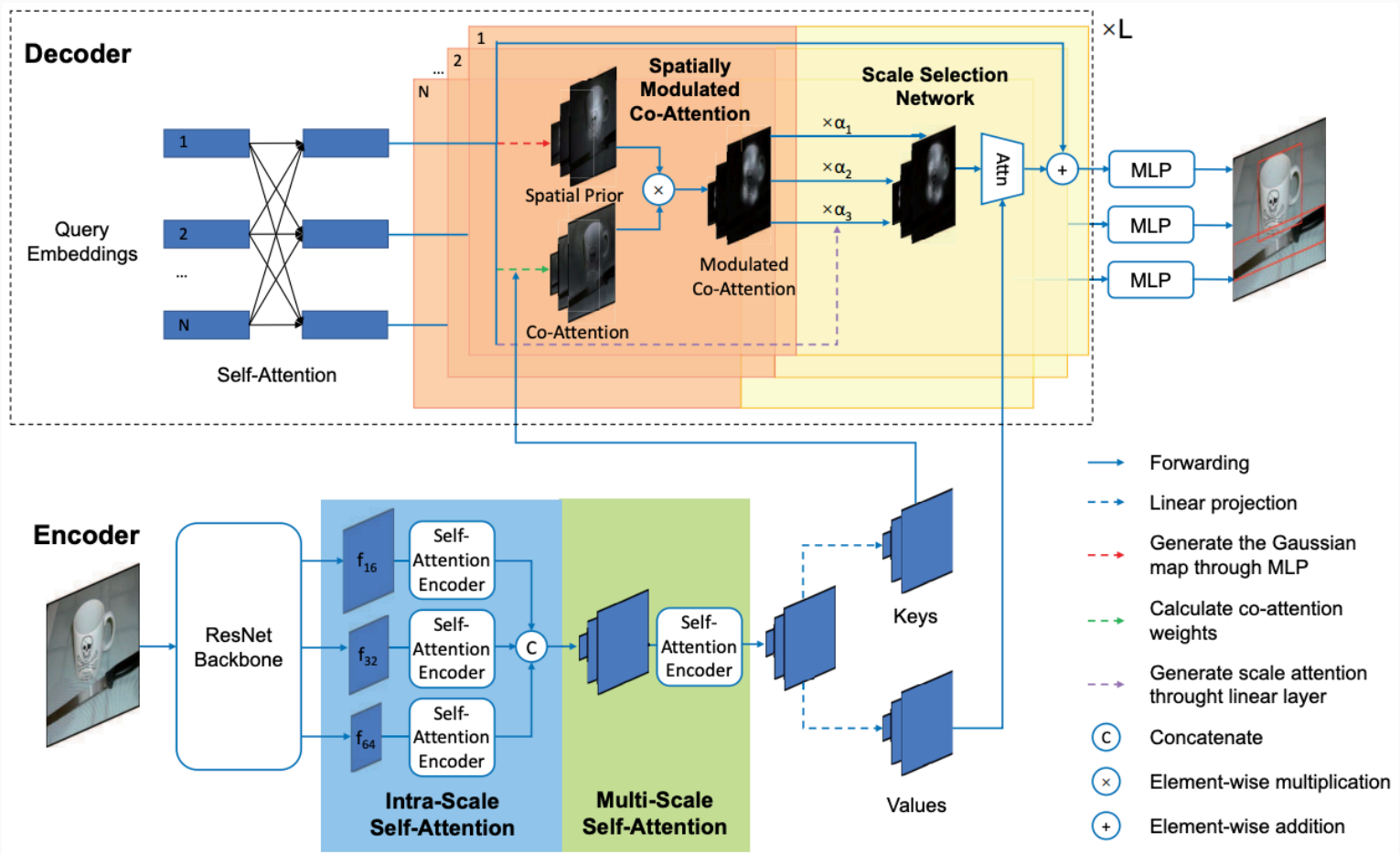
DETR и аналоги

- Обзор:



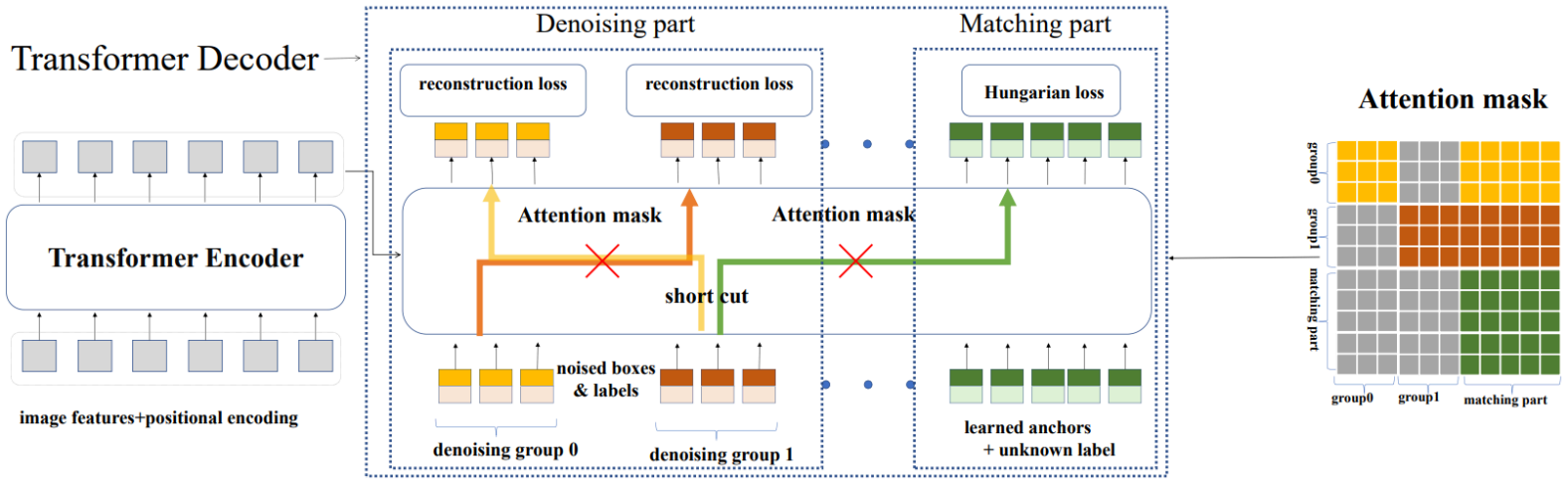
DETR и аналоги

- SMCA DETR (Spatially Modulated Co-Attention DETR; Gao et al., 2021) модулирует веса внимания через пространственные априорные распределения; то, что рядом, более важно: пространственные веса на основе предсказанных смещений относительно опорных точек



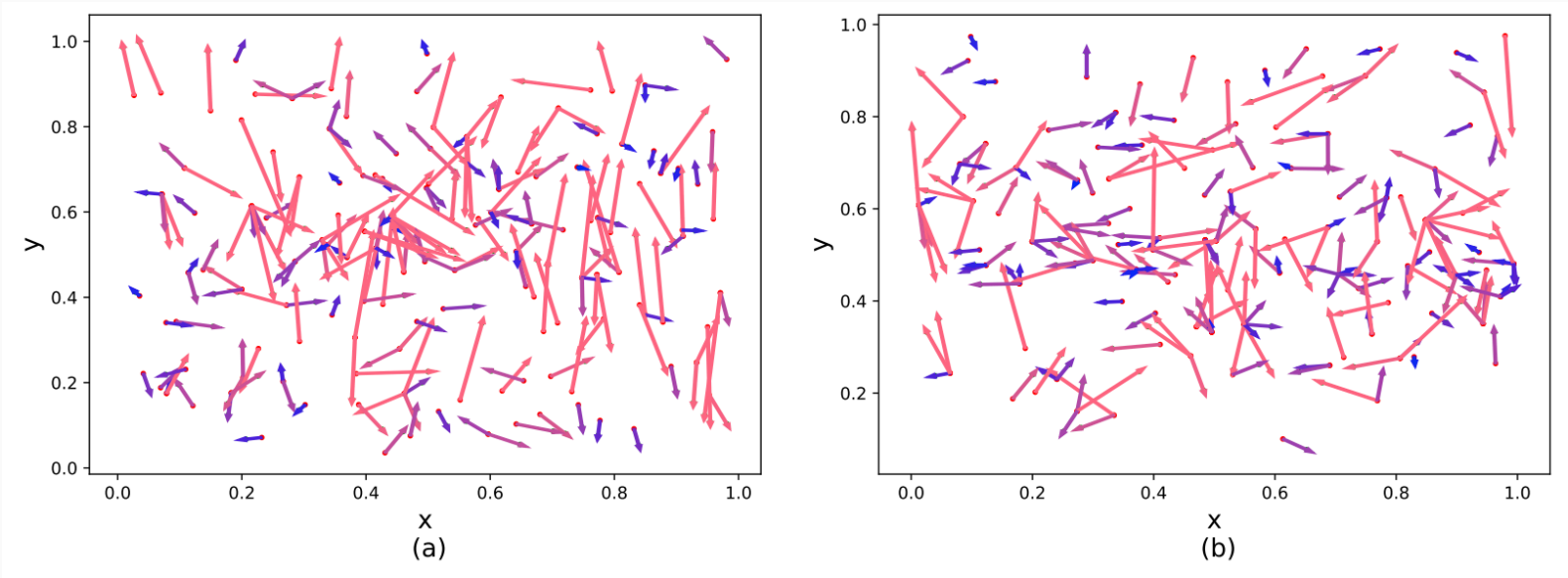
DETR и аналоги

- Другая проблема происходит от оптимального паросочетания: небольшое изменение весов может привести к совершенно другому результату, предсказания нестабильны, особенно в начале обучения
- DN-DETR (Li et al., 2022): давайте добавим другую цель обучения – реконструкцию зашумленных истинных прямоугольников, которые подаются с дополнительными запросами



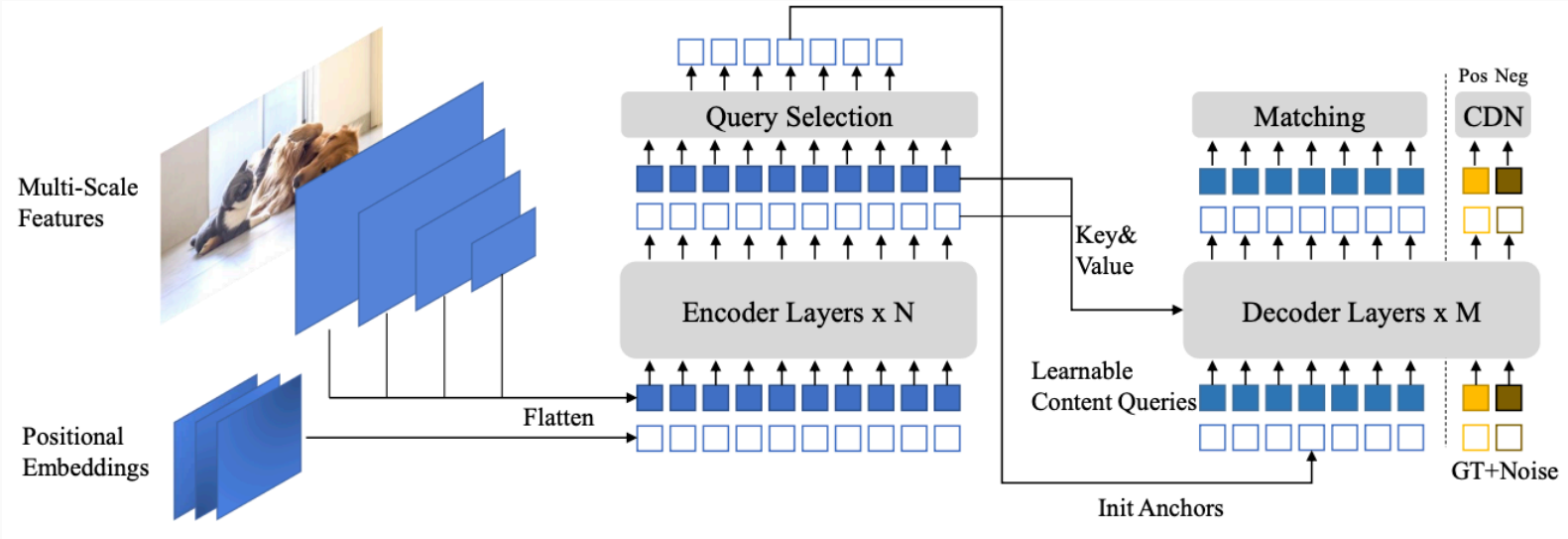
DETR и аналоги

- Это улучшает предсказание смещений: каждый запрос должен предсказывать очень разные смещения из-за нестабильности сопоставления, а запросы шумоподавления служат “хорошими якорями” для предсказания смещений
- В результате якоря располагаются ближе к своим целям (DAB-DETR слева, DN-DETR справа):



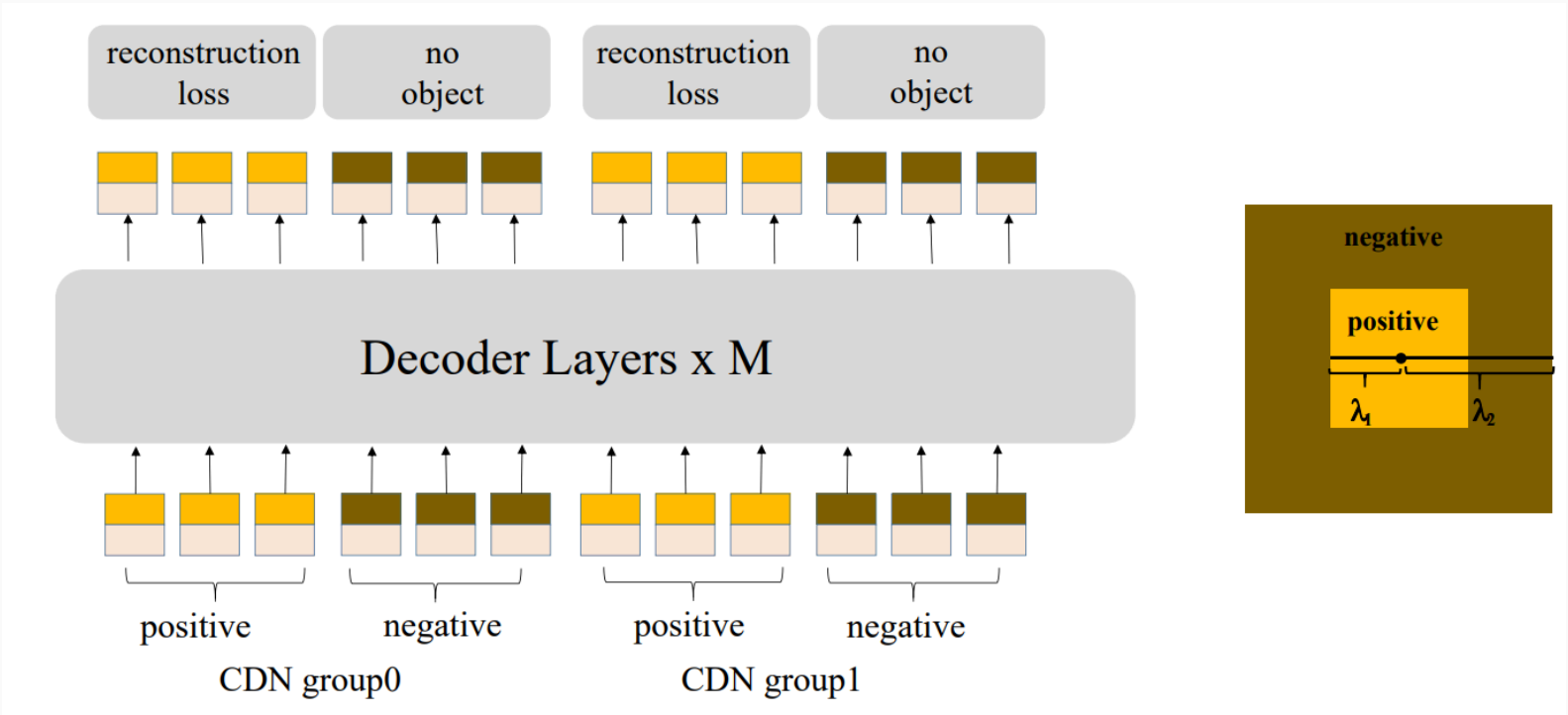
DETR и аналоги

- DINO (DETR with Improved DeNoising Anchor Boxes; Zhang et al., 2022): модификация DETR, основанная на идее DN-DETR
- Базовая структура более или менее та же:



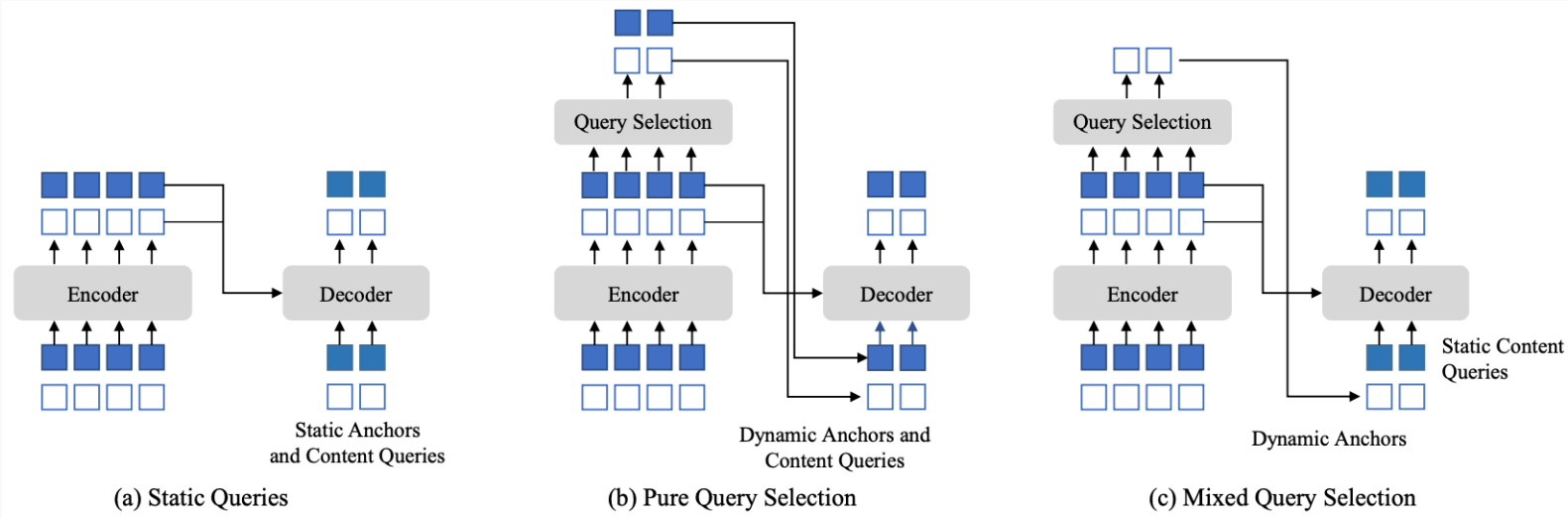
DETR и аналоги

- DN-DETR нужны хорошие якоря для улучшения обучения смещений; но что насчет плохих якорей, у которых нет близких истинных объектов?
- DINO добавляет отрицательные примеры для смягчения этого; часть входов – это якоря с большим шумом, которые должны быть отклонены



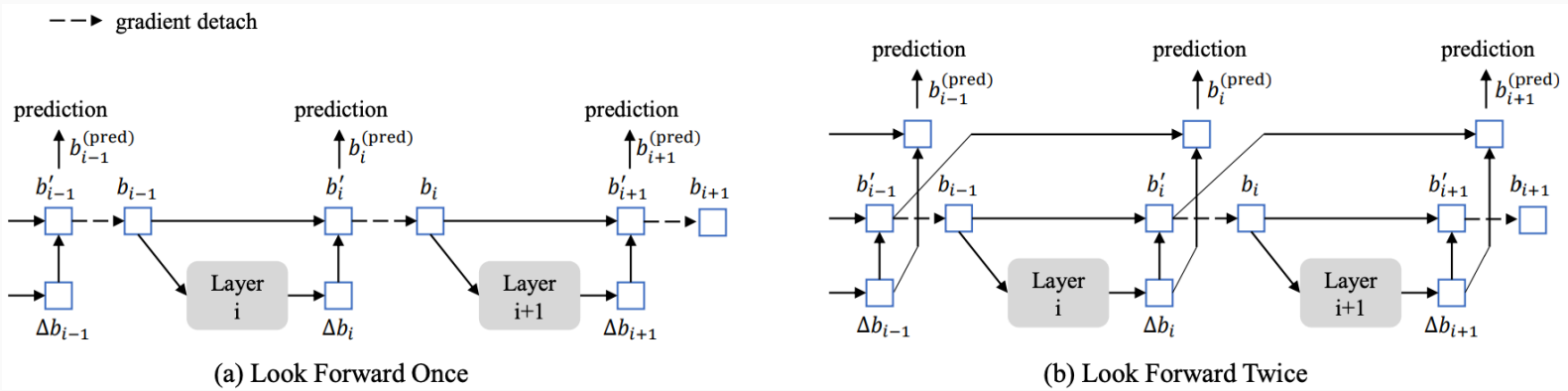
DETR и аналоги

- Смешанный выбор запросов: позиционные запросы инициализируются из предложений кодировщика, а содержательные запросы остаются независимыми



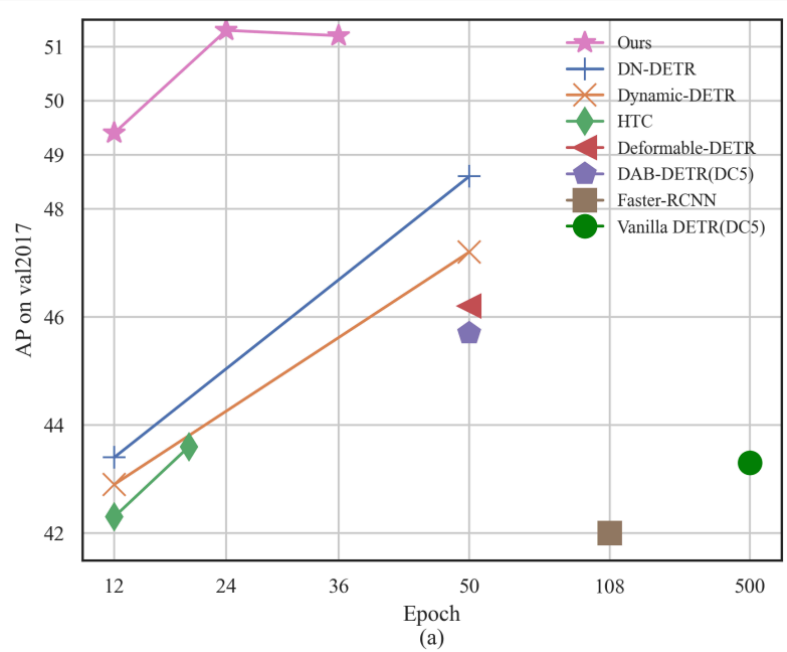
DETR и аналоги

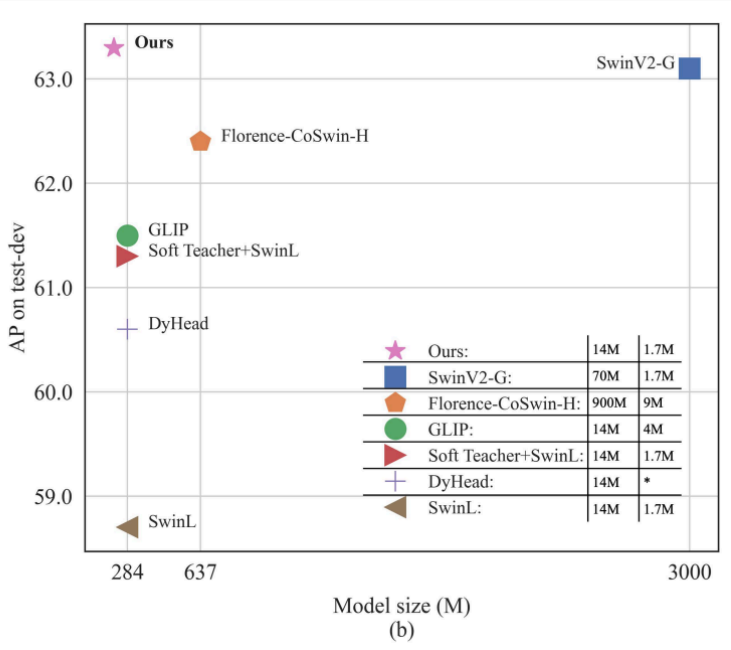
- Двойной lookahead: качество ограничивающего прямоугольника зависит от текущего якоря (позиционного запроса) и текущего предсказания смещения
- Deformable DETR блокировал обратное распространение градиента для стабилизации обучения: параметры слоя i обновляются на основе вспомогательной функции потерь от прямоугольников b_i
- DINO блокирует градиенты еще на один шаг дальше, поэтому параметры слоя i обновляются на основе вспомогательной функции потерь от прямоугольников b_i и b_{i+1}



DETR и аналоги

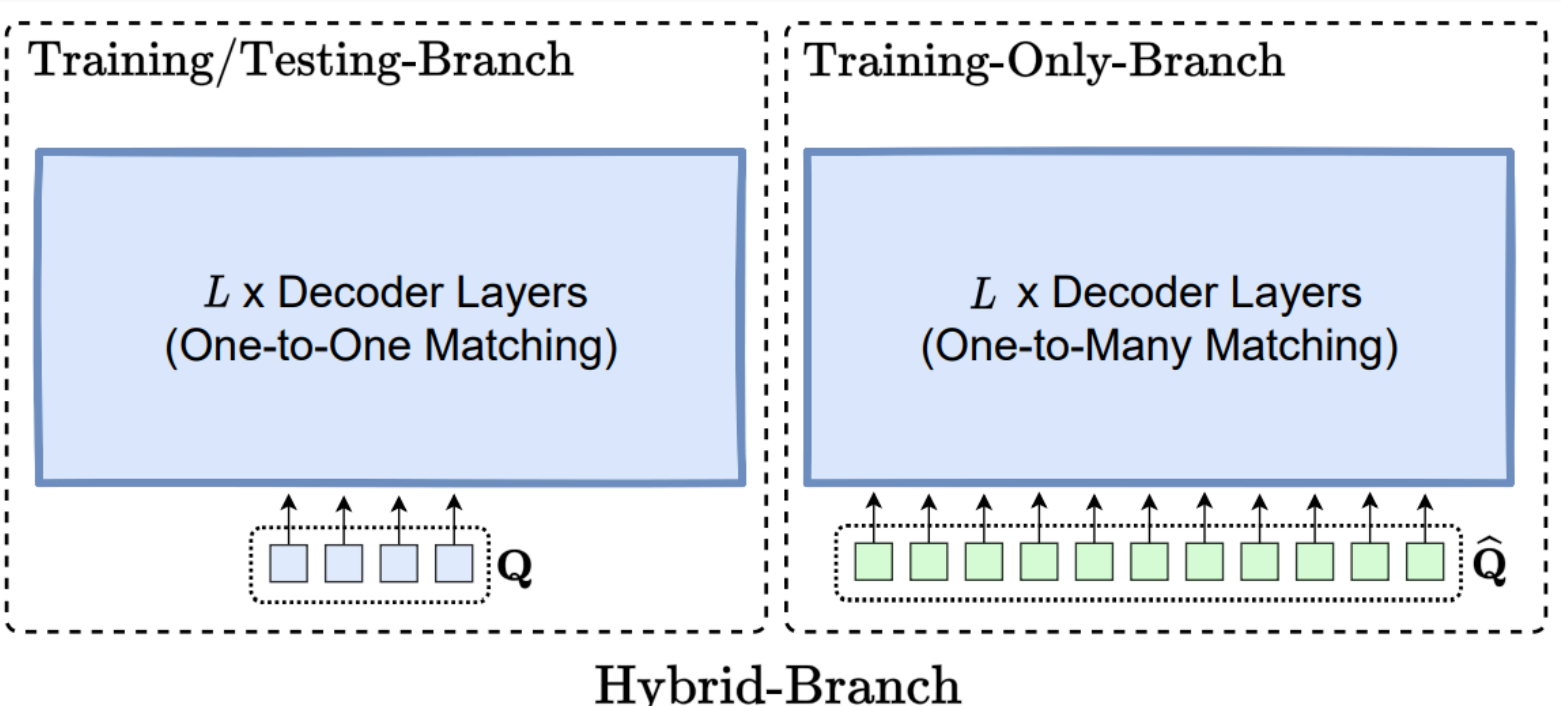
- Очень хорошие результаты, быстрое обучение (слева с ResNet-50 backbone)





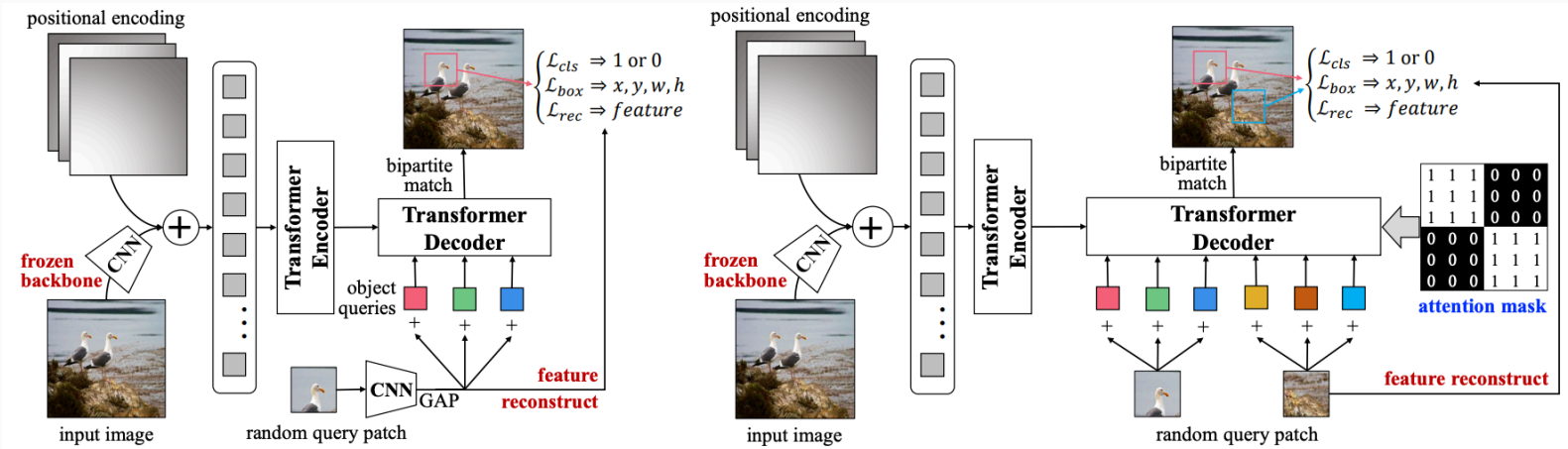
DETR и аналоги

- H-Deformable-DETR (Jia et al., 2023) – очень простая и интересная идея
- У нас очень мало положительных примеров, так что давайте их копировать!
- При обучении мы добавляем дополнительные запросы, которые должны предсказывать те же истинные прямоугольники, с масками внимания, которые разделяют группы запросов



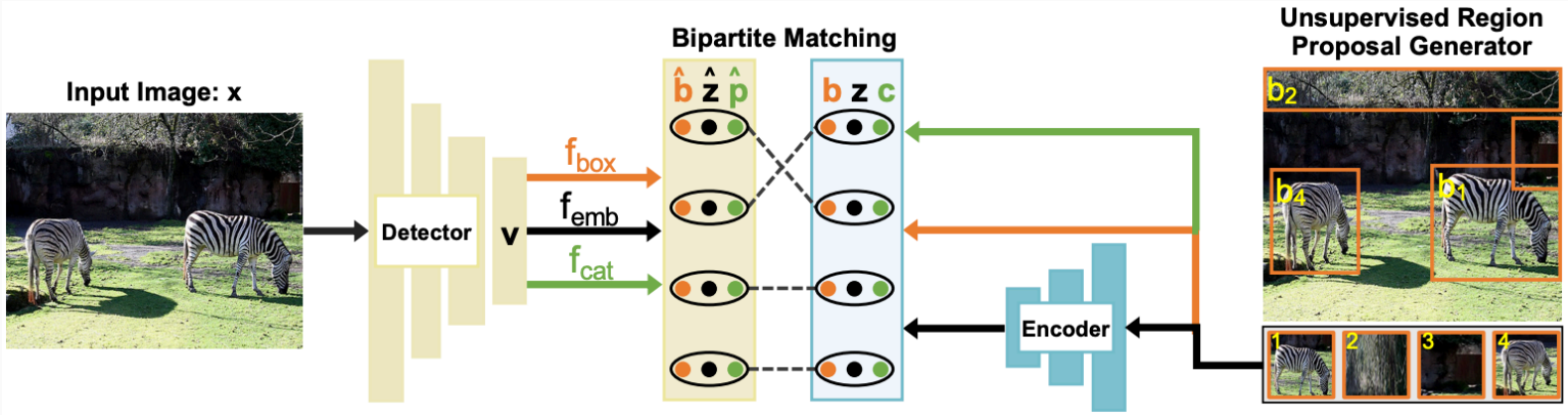
DETR и аналоги

- Как еще мы можем исправить долгое время обучения DETR?
- UP-DETR (Dai et al., 2021) – давайте добавим предобучение без обучения, self-supervised; мы берем случайный кусок и просим DETR предсказать его положение и восстановить его признаки (это нужно, потому что у нас нет меток классификации, но нужно сохранить семантику):



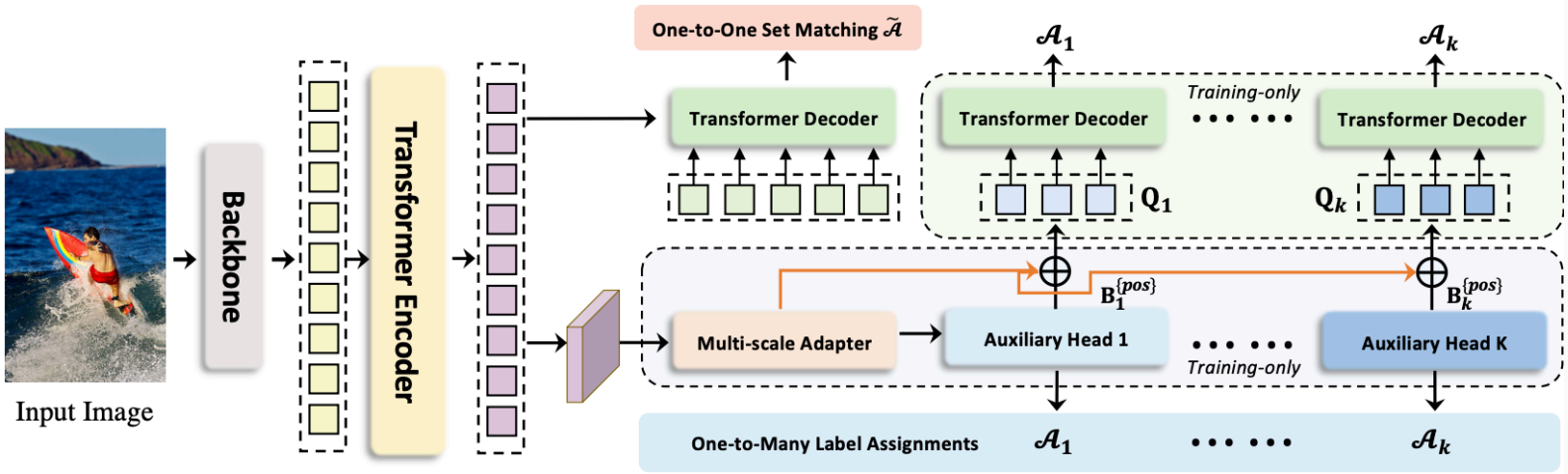
DETR и аналоги

- DETReg (Bar et al., 2022) – вместо случайных кропов давайте используем объекты или области, найденные другими алгоритмами, возможно, даже selective search
- Мы также можем предсказывать вложения для этих псевдо-объектов



DETR и аналоги

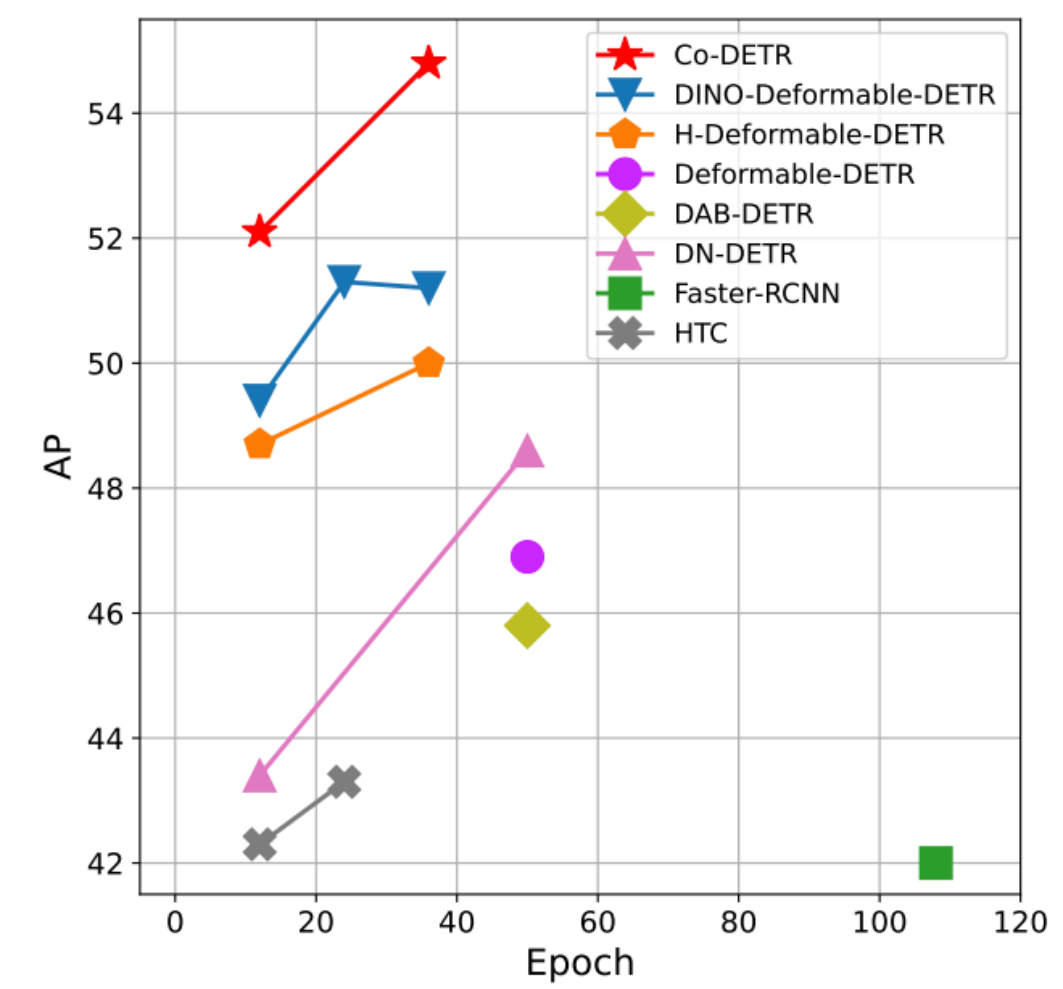
- Co-DETR (Zong et al., 2023) добавляет несколько вспомогательных голов из других object detection моделей; новые положительные примеры через паросочетания



Head i	Loss $\mathcal{L}_i$	Assignment $\mathcal{A}_i$		
		$\{pos\}$, $\{neg\}$ Generation	$\mathbf{P}_i$ Generation	$\mathbf{B}_i^{\{pos\}}$ Generation
Faster-RCNN [27]	cls: CE loss, reg: GIoU loss	$\{pos\}$: IoU(proposal, gt)>0.5 $\{neg\}$: IoU(proposal, gt)<0.5	$\{pos\}$: gt labels, offset(proposal, gt) $\{neg\}$: gt labels	positive proposals (x_1, y_1, x_2, y_2)
ATSS [40]	cls: Focal loss reg: GIoU, BCE loss	$\{pos\}$:IoU(anchor, gt)>(mean+std) $\{neg\}$: IoU(anchor, gt)<(mean+std)	$\{pos\}$: gt labels, offset(anchor, gt), centerness $\{neg\}$: gt labels	positive anchors (x_1, y_1, x_2, y_2)
RetinaNet [21]	cls: Focal loss reg: GIoU Loss	$\{pos\}$: IoU(anchor, gt)>0.5 $\{neg\}$: IoU(anchor, gt)<0.4	$\{pos\}$: gt labels, offset(anchor, gt) $\{neg\}$: gt labels	positive anchors (x_1, y_1, x_2, y_2)
FCOS [31]	cls: Focal Loss reg: GIoU, BCE loss	$\{pos\}$: points inside gt center area $\{neg\}$: points outside gt center area	$\{pos\}$: gt labels, ltrb distance, centerness $\{neg\}$: gt labels	FCOS point (cx, cy) $w = h = 8 \times 2^{2+j}$

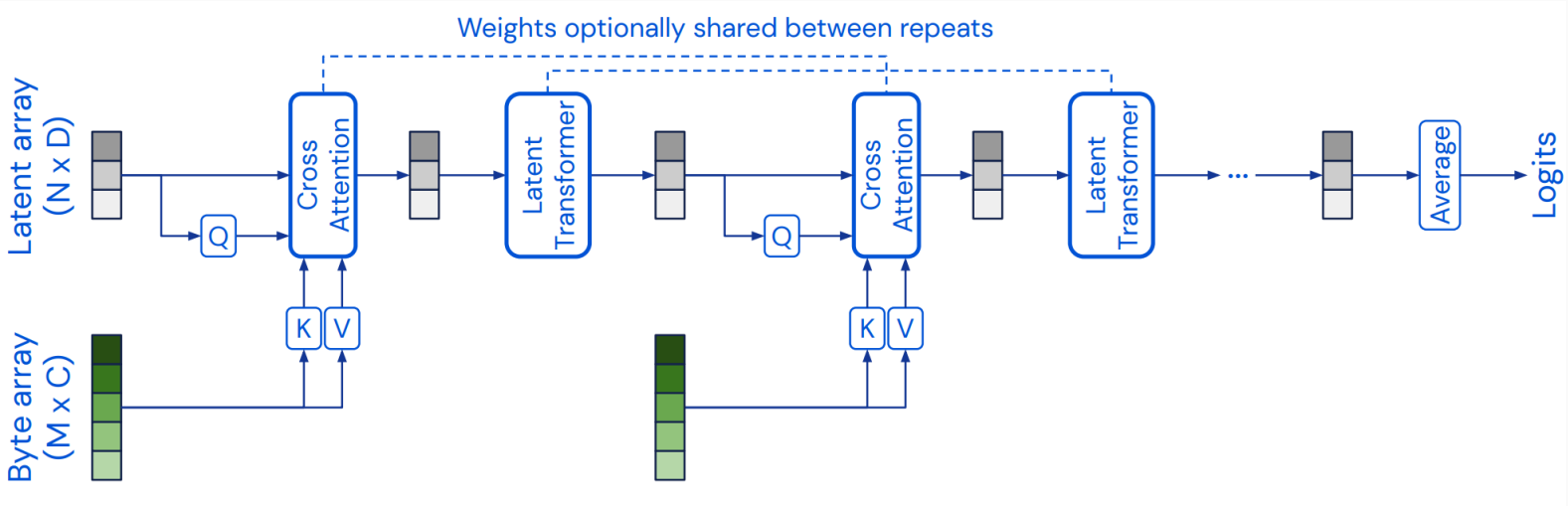
DETR и аналоги

- График для ResNet-50 backbone:



Perceiver

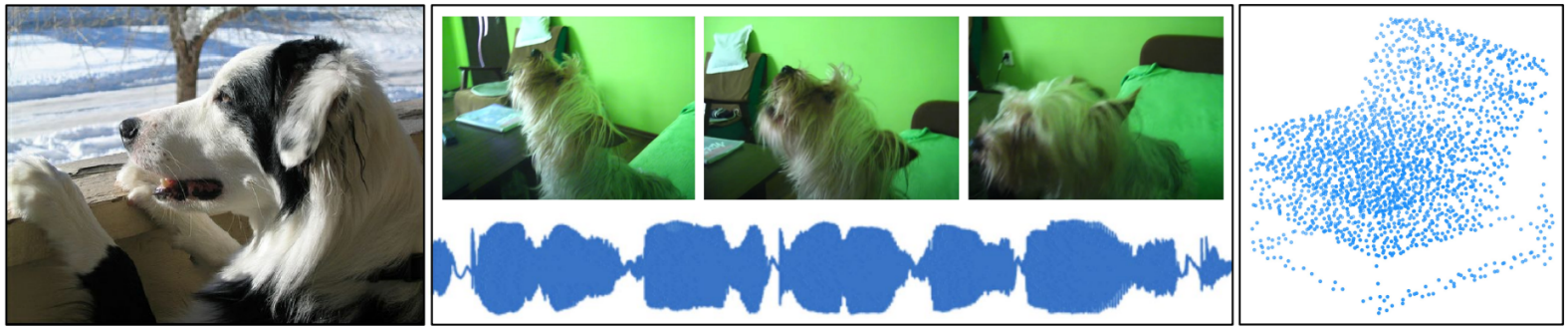
- *Perceiver* от *DeepMind* (Jaegle et al., 2021a) – это архитектура общего назначения, которая может обрабатывать множество различных модальностей



- Основная идея – избежать квадратичной сложности, используя латентные единицы меньшей размерности
- Это квадратично по числу запросов, поэтому мы используем небольшой вектор латентных переменных для запросов и добавляем большие массивы для K и V

Perceiver

- Обучен на изображениях, облаках точек, аудио и видео, без специальных кодировщиков и без изменения архитектуры

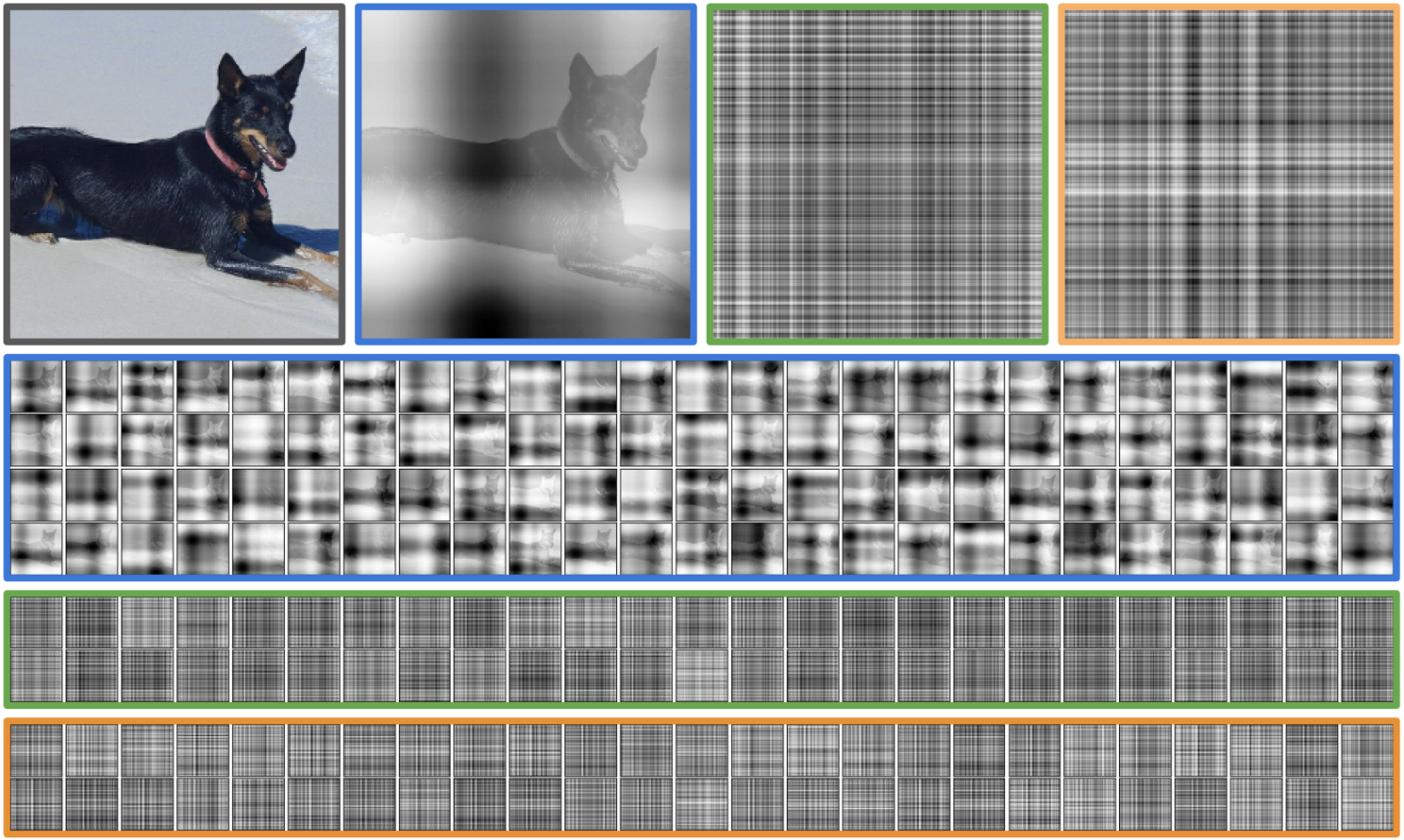


ResNet-50 (He et al., 2016)	77.6
ViT-B-16 (Dosovitskiy et al., 2021)	77.9
ResNet-50 (FF)	73.5
ViT-B-16 (FF)	76.7
Transformer (64x64, FF)	57.0
Perceiver (FF)	78.0

	Raw	Perm.	Input RF
ResNet-50 (FF)	73.5	39.4	49
ViT-B-16 (FF)	76.7	61.7	256
Transformer (64x64) (FF)	57.0	57.0	4,096
Perceiver:			
(FF)	78.0	78.0	50,176
(Learned pos.)	70.9	70.9	50,176

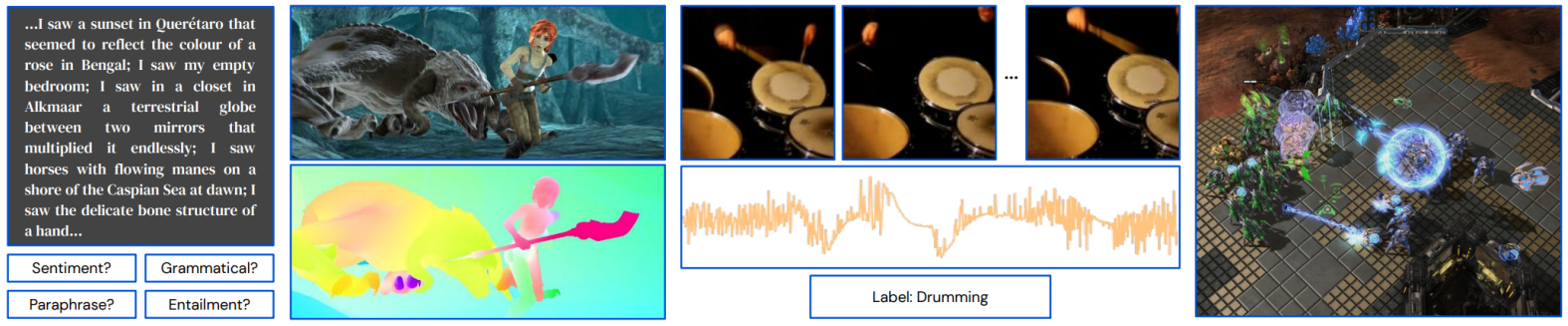
Perceiver

- Карты внимания опускаются до отдельных пикселей:



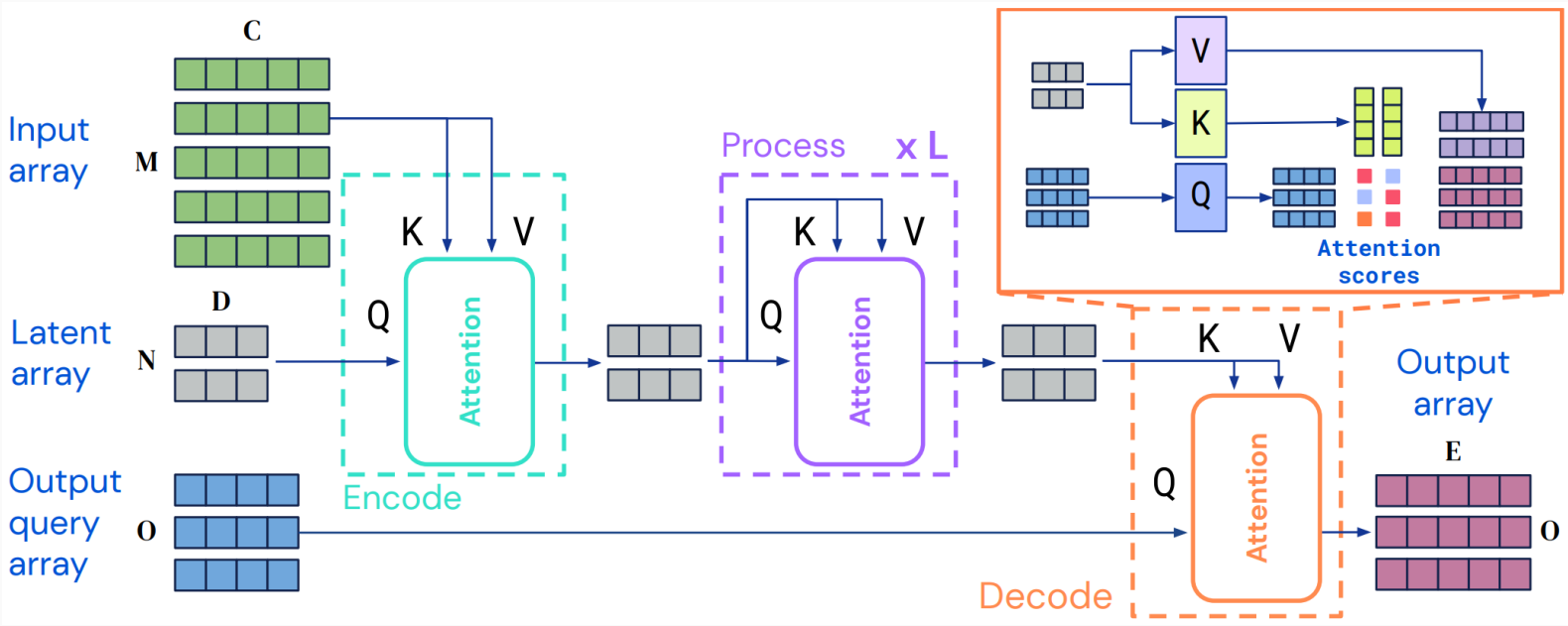
Perceiver

- *Perceiver IO* (Jaegle et al., 2021b), следующая версия, которая также позволяет строить большие *выходы*
- Может обрабатывать еще более разнообразные структурированные данные: многозадачное понимание языка, плотные визуальные задачи (например, optical flow), гибридные плотные/разреженные мультимодальные задачи (например, автокодировщик для видео+аудио), задачи с символьными выходами (например, StarCraft II) и так далее



Perceiver

- Архитектура Perceiver IO:

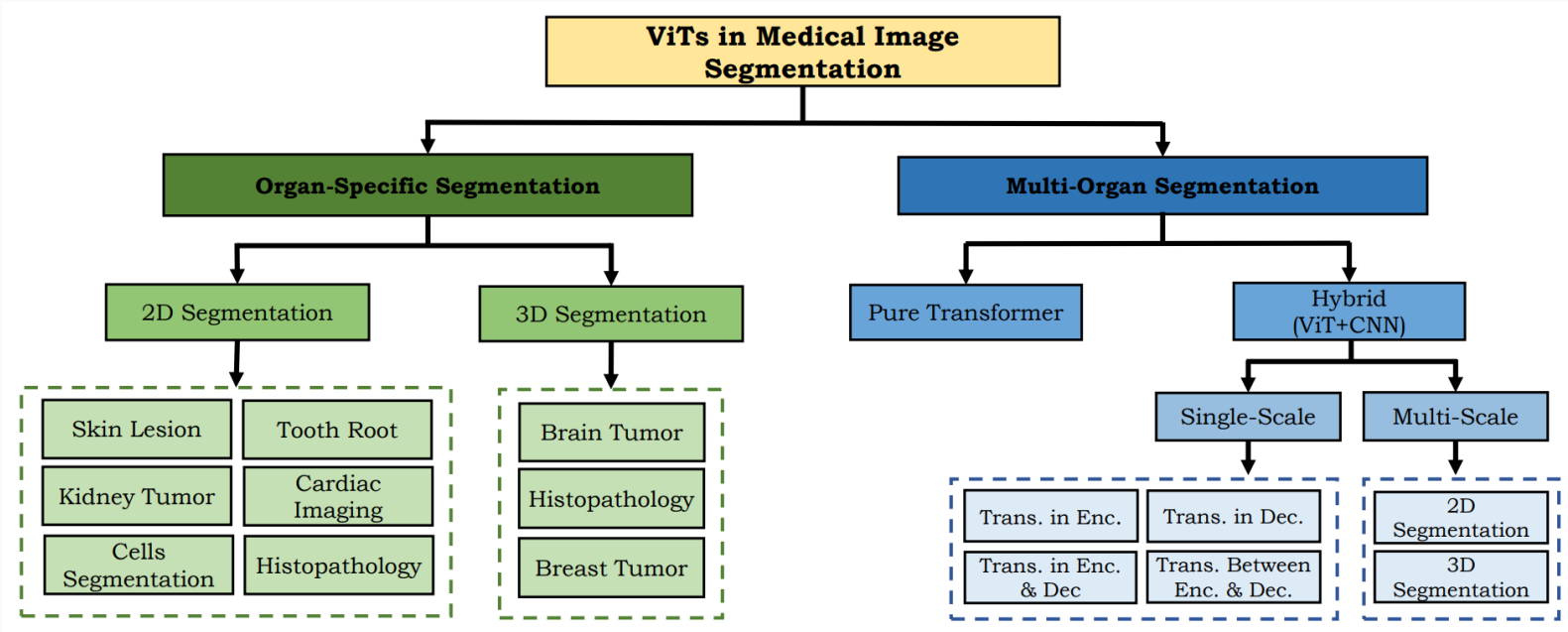


- Фактические выходные запросы конструируются автоматически путем комбинирования набора векторов, которые описывают свойства текущего выхода

Case study: сегментация медицинских изображений

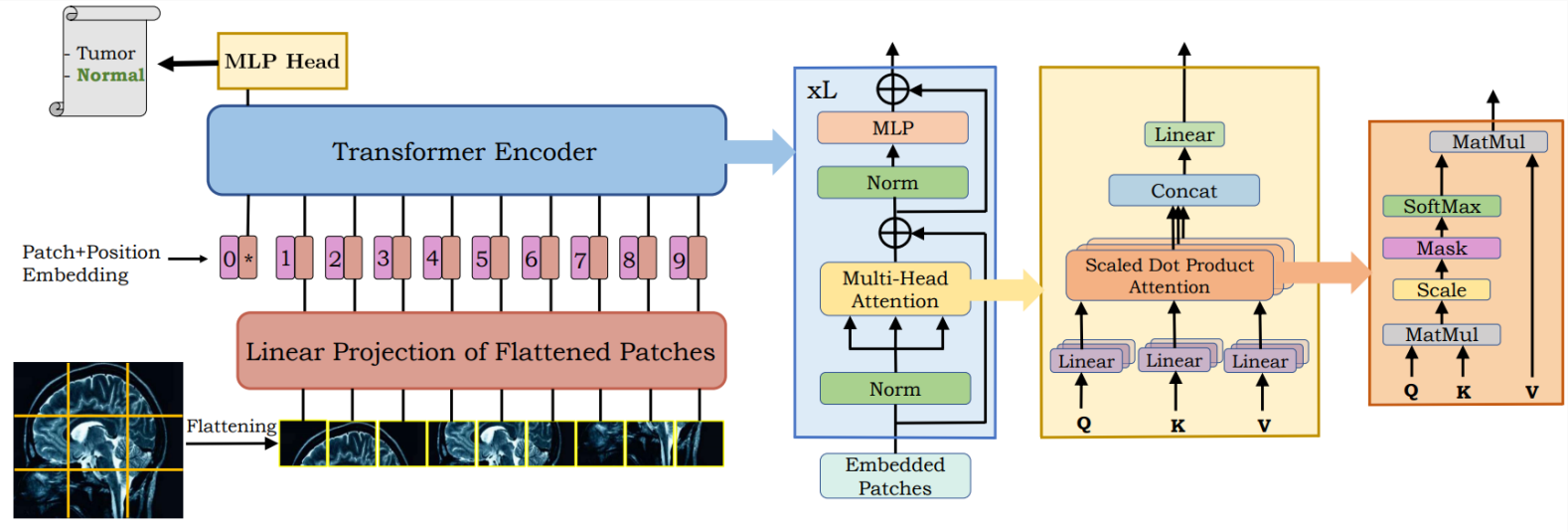
Сегментация медицинских изображений

- Сегментация звучит как более сложная задача для трансформеров
- Но вот таксономия только для ViT, только в сегментации медицинских снимков, причём три года назад (Shamshad et al., 2022):



Сегментация медицинских изображений

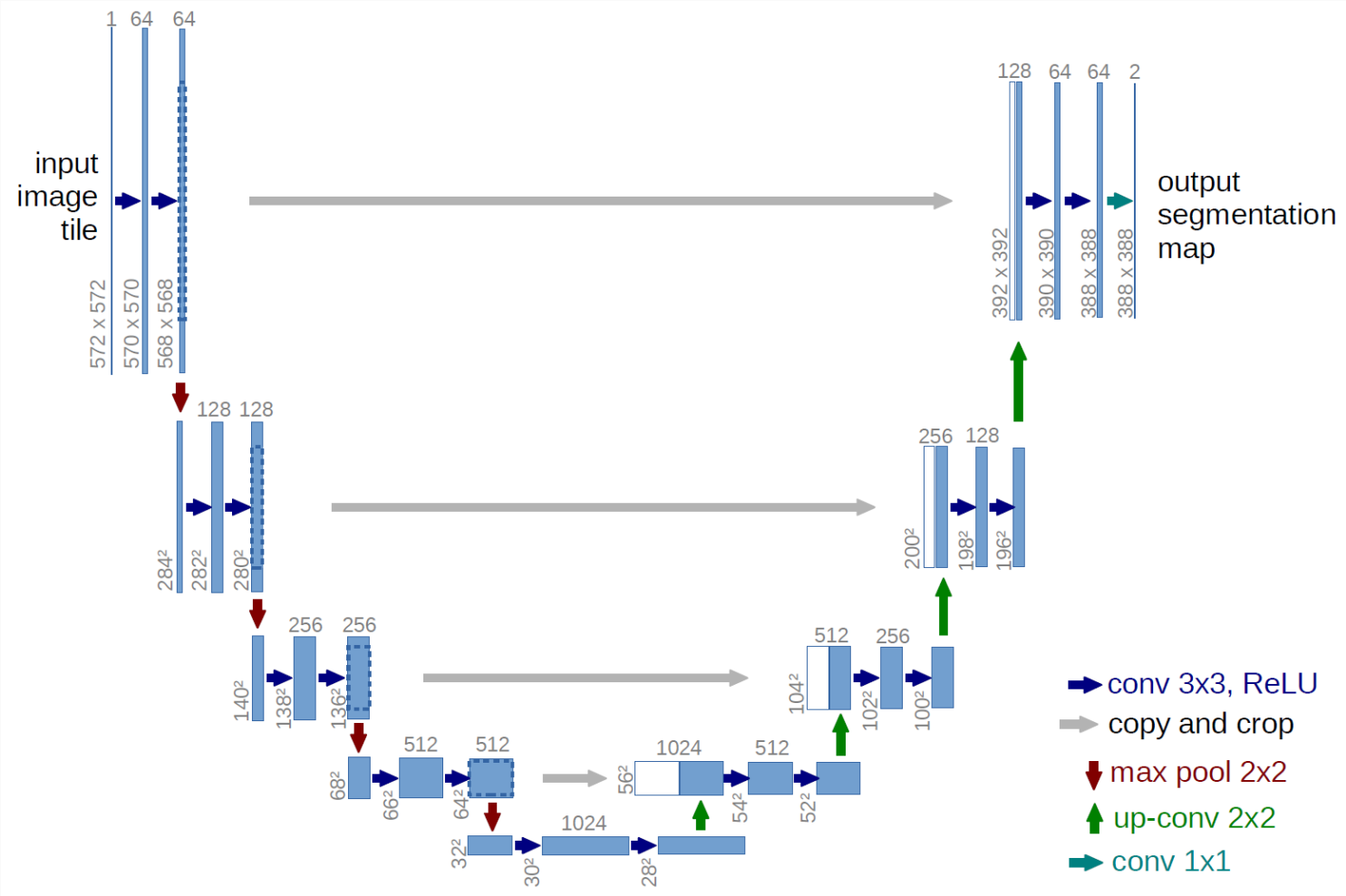
- В медицинских задачах *маленькие датасеты*, поэтому трудно ожидать ViT, обученный с нуля на медицинском датасете



- Иногда это возможно, но часто приходится модифицировать архитектуру, чтобы добавить априорную информацию о медицинских особенностях

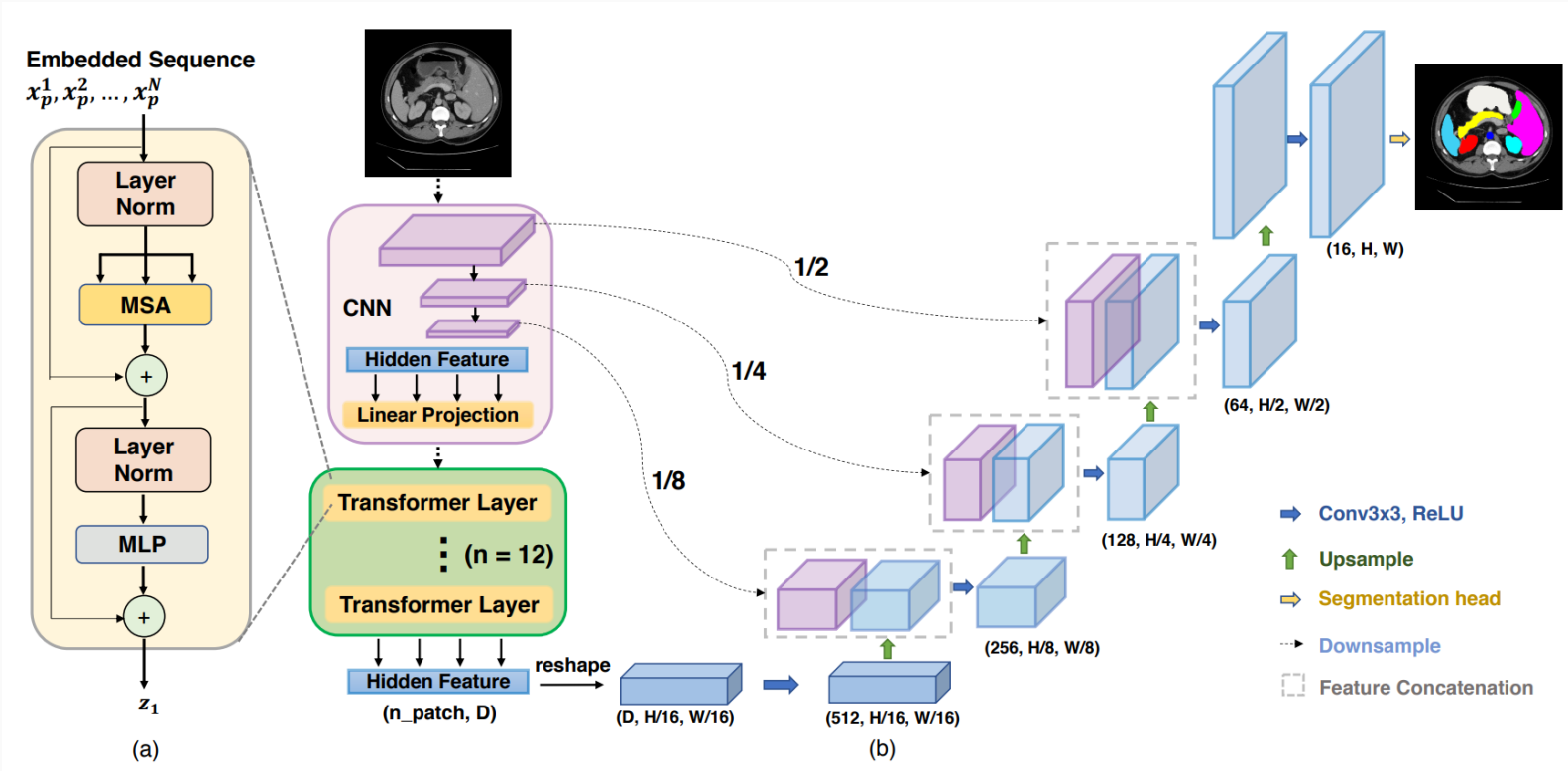
Сегментация медицинских изображений

- В сегментации U-Net долгое время была самой важной архитектурой для medical imaging:



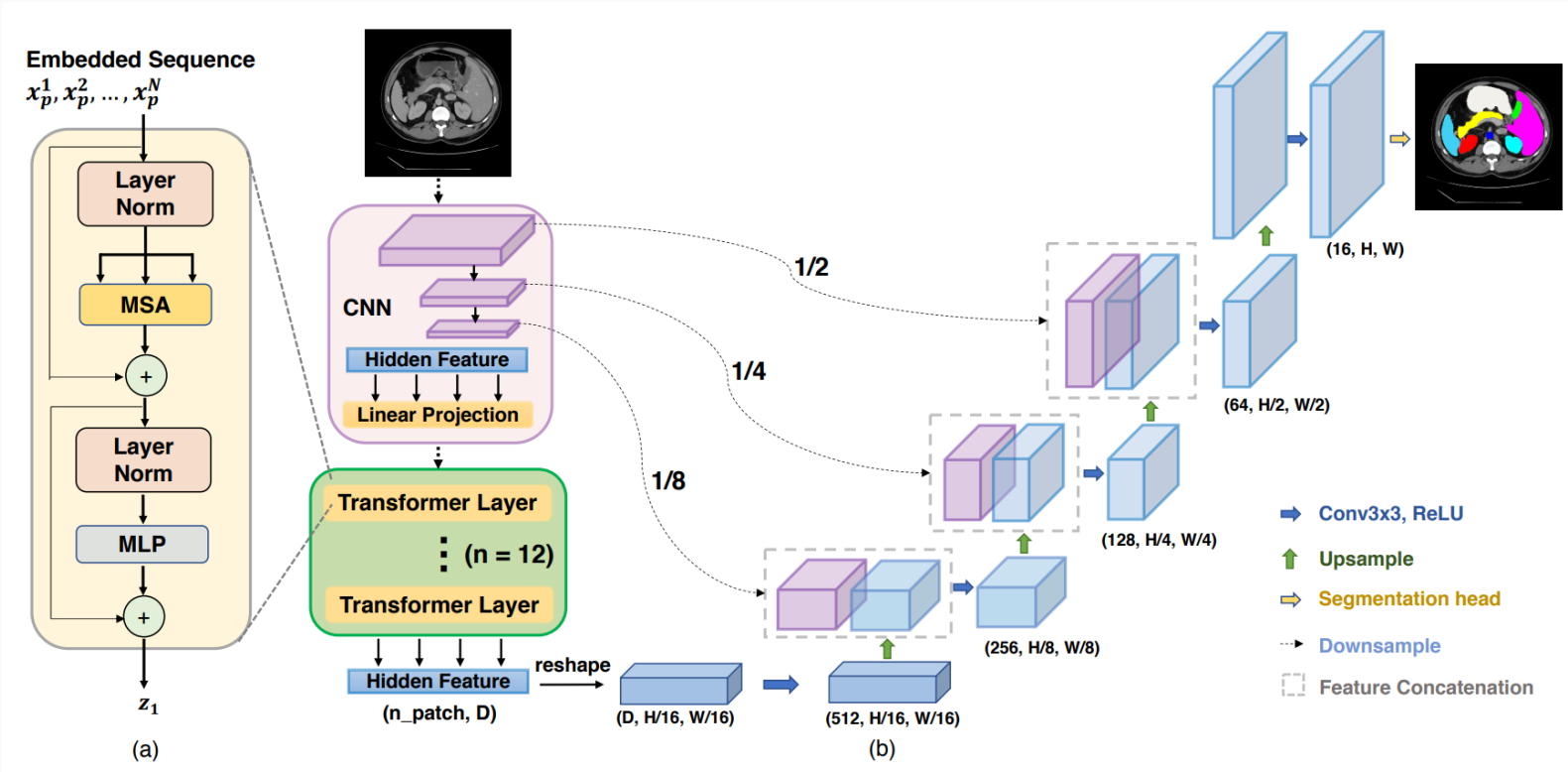
Сегментация медицинских изображений

- TransUNet (Chen et al., 2021): давайте используем трансформеры в кодировщике U-Net



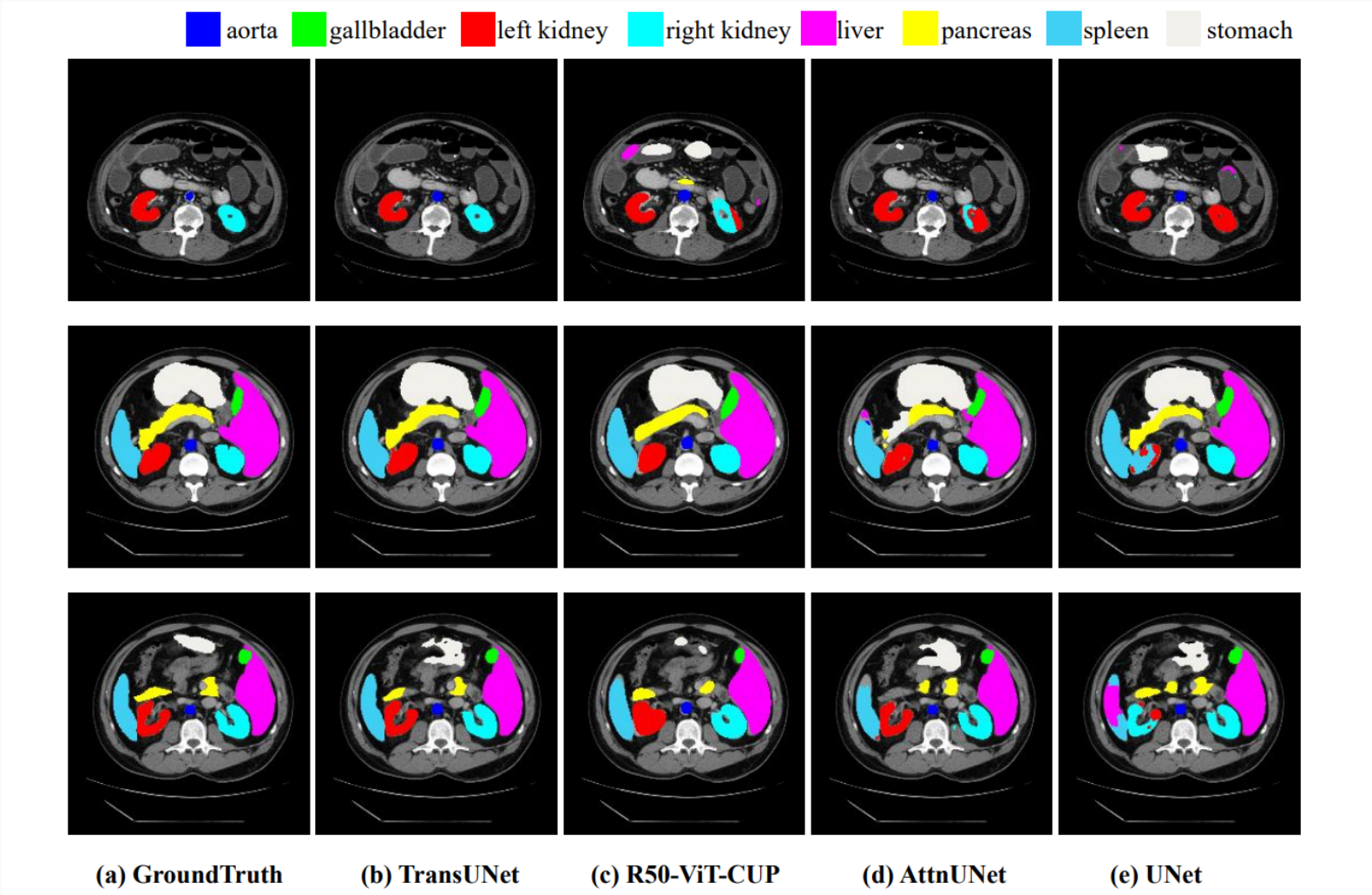
Сегментация медицинских изображений

- ViT получает вложения патчей из разных слоев CNN, а повышение разрешения аналогично U-Net:



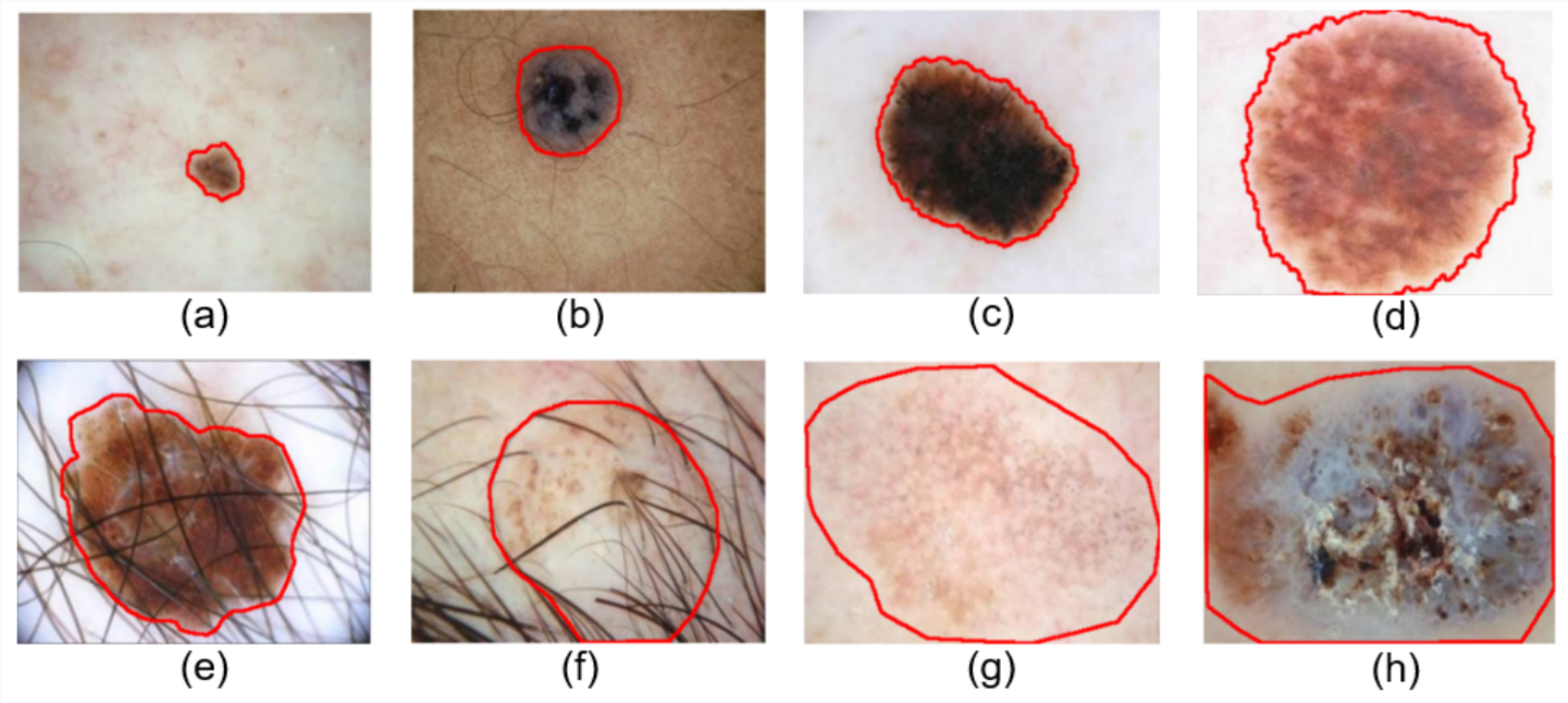
Сегментация медицинских изображений

- TransUNet сохраняет больше деталей:



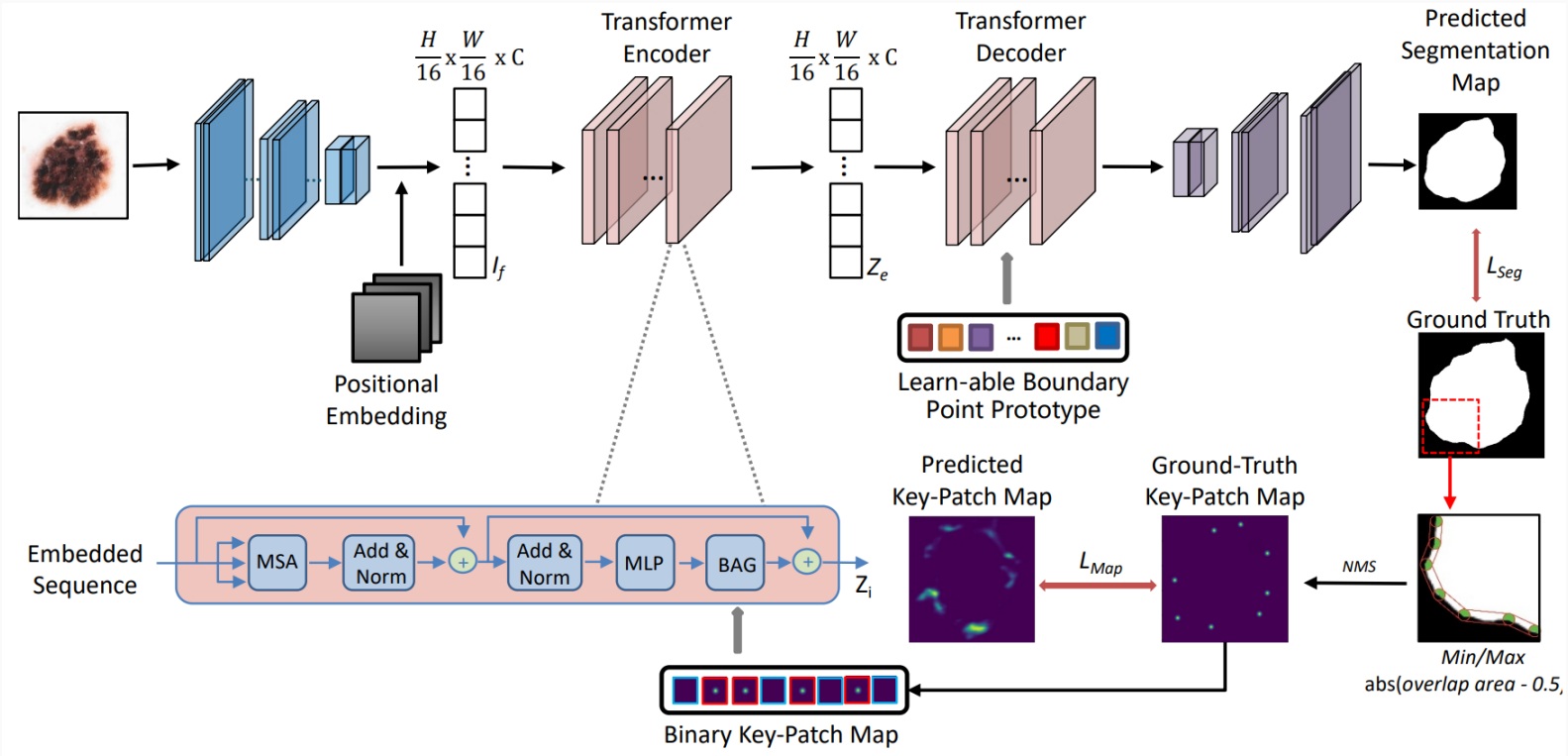
Сегментация медицинских изображений

- Boundary-aware Transformers (BAT; Wang et al., 2021) для сегментации поражений кожи, типичная задача визуализации, где нужно добавить какие-то дополнительные априорные распределения



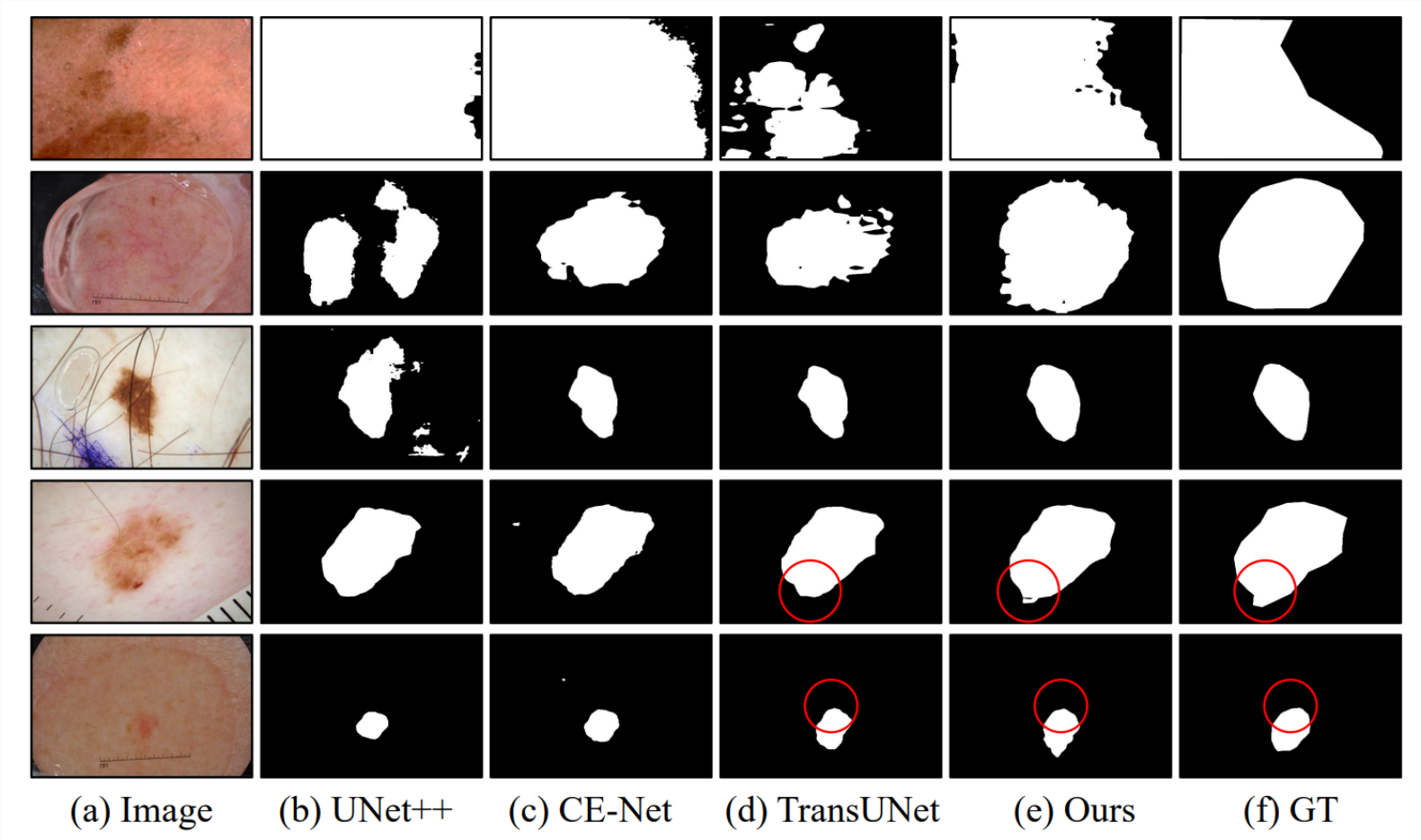
Сегментация медицинских изображений

- BAT похож на ViT, но добавляет гейт граничного внимания (BAG), который принимает признаки на вход и выдает бинарную карту внимания на уровне патчей, где 1 означает, что патч лежит на неоднозначной границе:



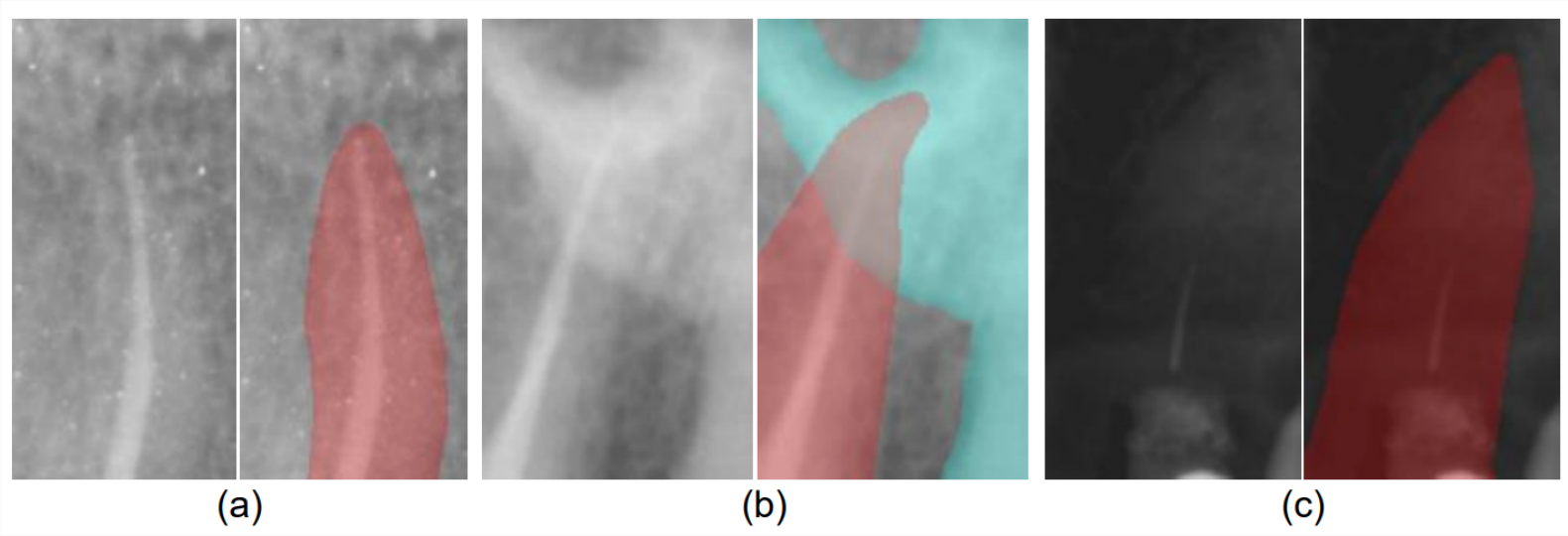
Сегментация медицинских изображений

- BAT дает результаты лучше, чем, например, TransUNet:



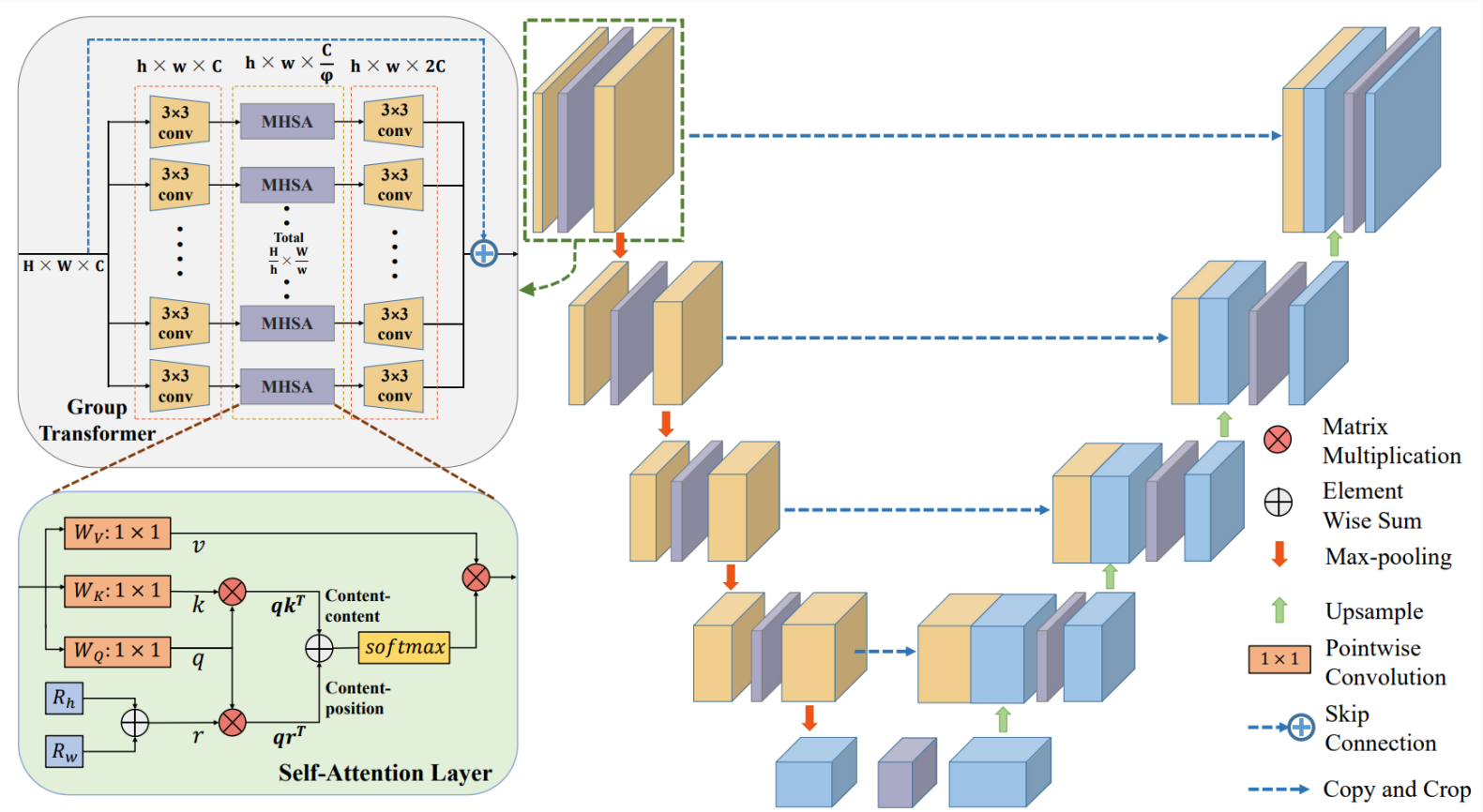
Сегментация медицинских изображений

- Group Transformer Network (GT U-Net; Li et al., 2021): сегментация рентгеновских снимков зубов для эндодонтического лечения
- Рентгеновские снимки часто плохого качества (недодержанные или передержанные), границы размыты, есть наложения...



Сегментация медицинских изображений

- GT U-Net сохраняет структуру U-Net, но добавляет Group Transformer:



Сегментация медицинских изображений

- Интересная часть – *shape-sensitive Fourier descriptor*: если (x_m, y_m) — координата на границе корня зуба с N пикселями, мы можем сформировать форму границы как комплексное число $z(m) = x_m + jy_m$ и определить дескриптор Фурье как

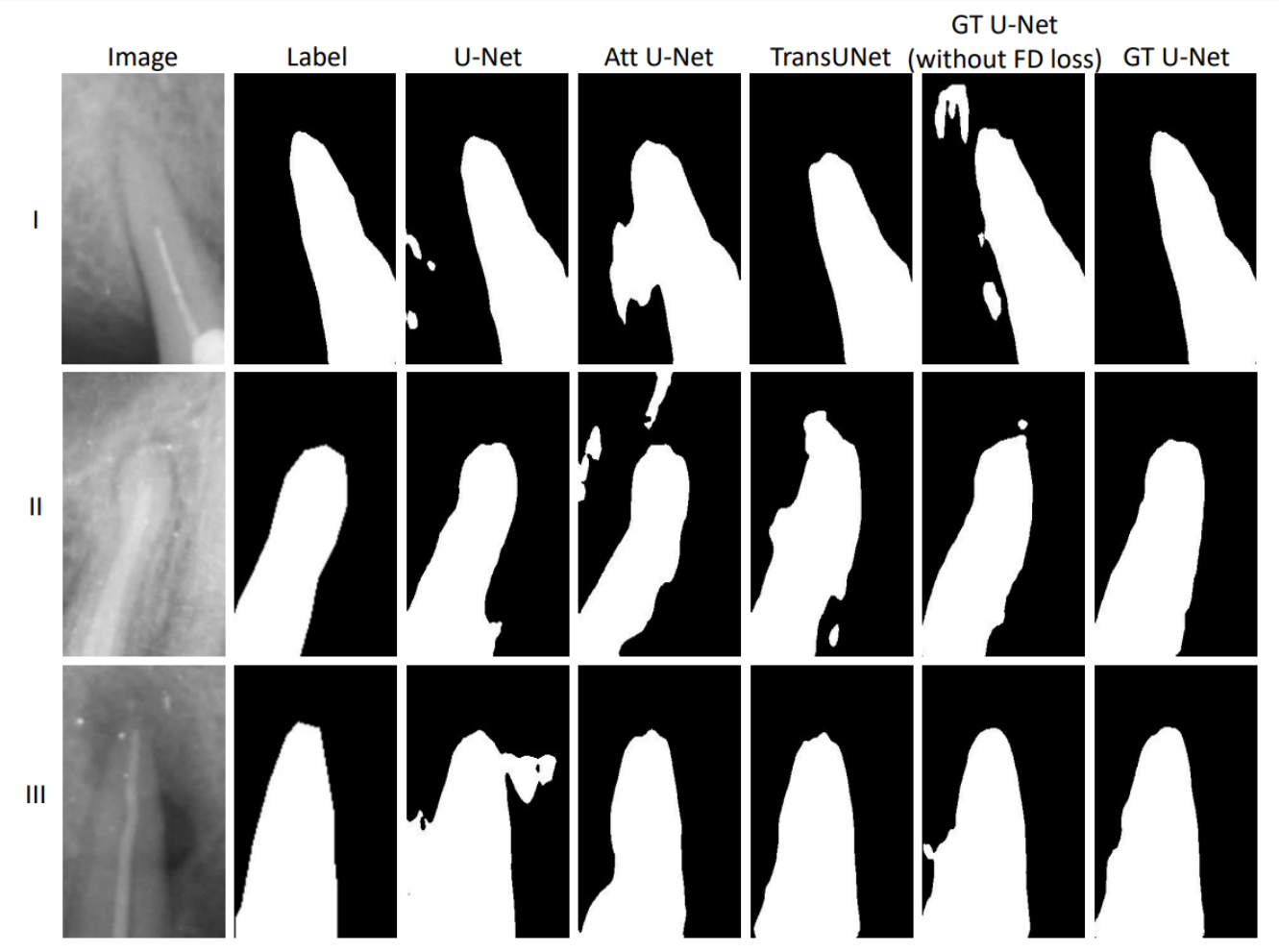
$$Z(k) = \mathrm{DFT}(z(m)) = \frac{1}{N} \sum_{m=0}^{N-1} z(m) e^{-j\frac{2\pi mk}{N}}.$$

- Это количественное представление замкнутых контуров, независимое от их начальной точки, масштаба, местоположения и вращения
- Поэтому в дополнение к бинарной кросс-энтропии мы можем вычислить разность между дескрипторами Фурье $\Delta Z(k) = \left| Z(k) - \hat{Z}(k) \right|$ и определить функцию потерь FD как

$$L_{\mathrm{FD}} = \mathrm{BCE}(\mathbf{x}, \hat{\mathbf{x}}) \cdot \frac{1}{1 + e^{-\beta \cdot \Delta Z(k)}}.$$

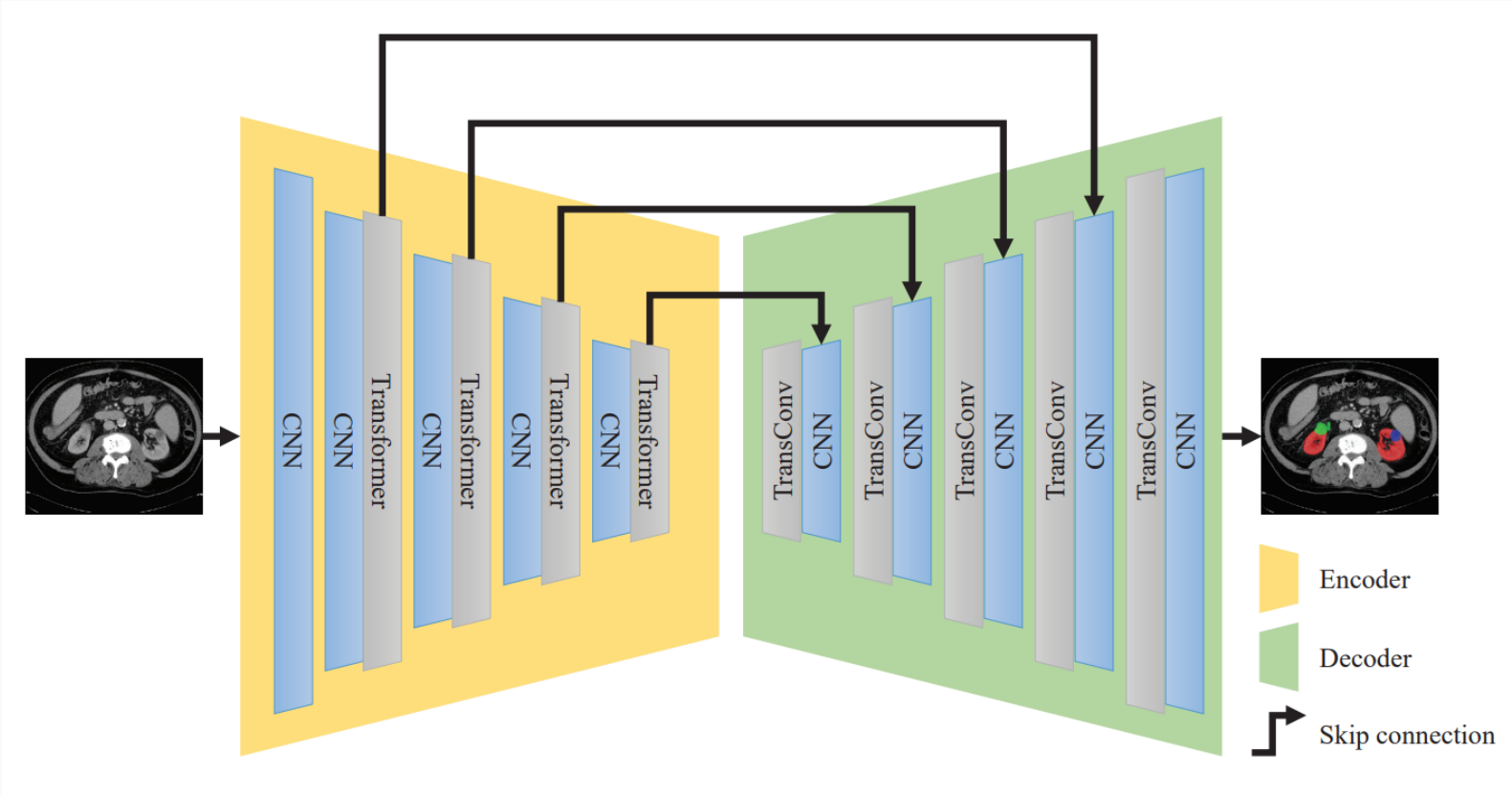
Сегментация медицинских изображений

- В результате GT U-Net улучшает результаты сегментации, особенно с функцией потерь FD:



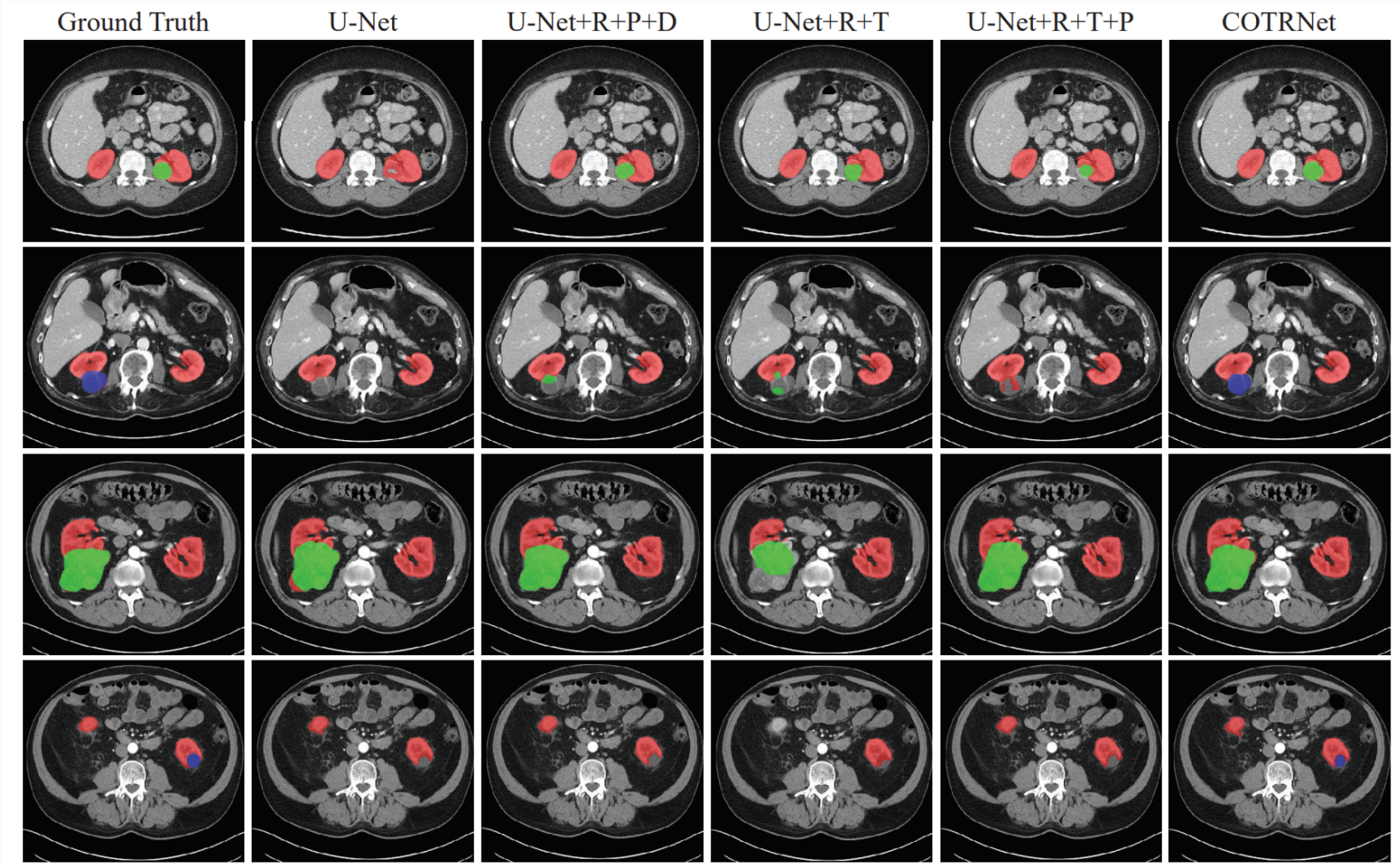
Сегментация медицинских изображений

- Сегментация опухолей почек с COTR-Net (Shen et al., 2021): добавим трансформеры внутрь U-Net



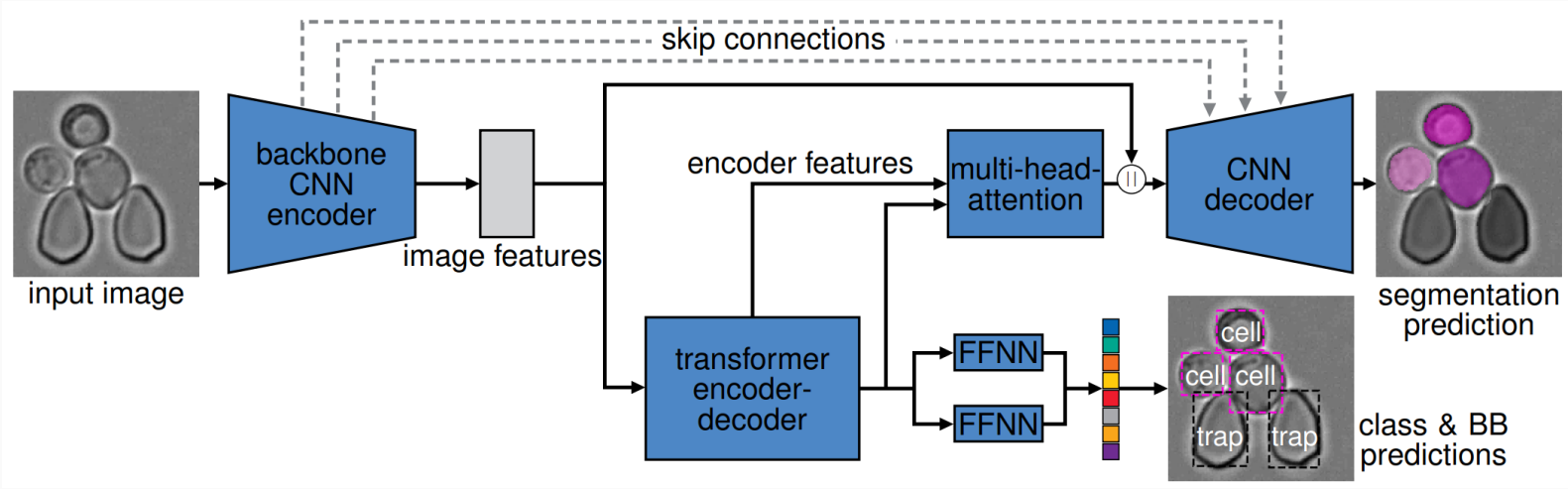
Сегментация медицинских изображений

- COTR-Net улучшает сегментацию:



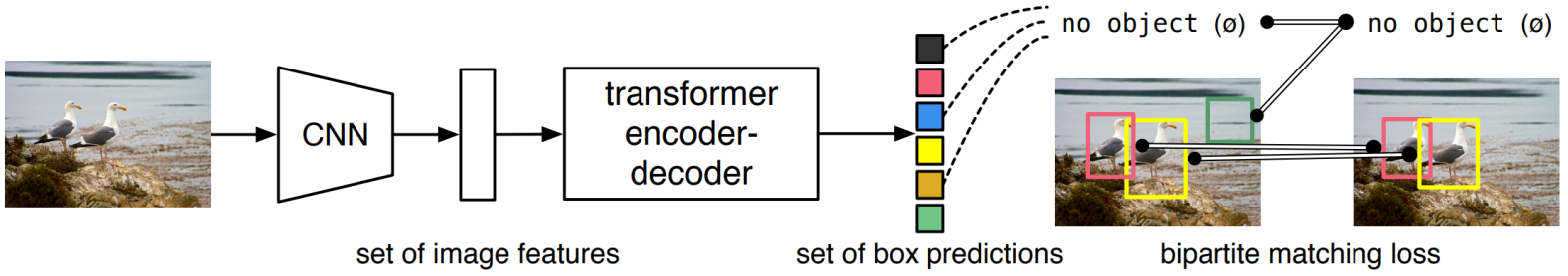
Сегментация медицинских изображений

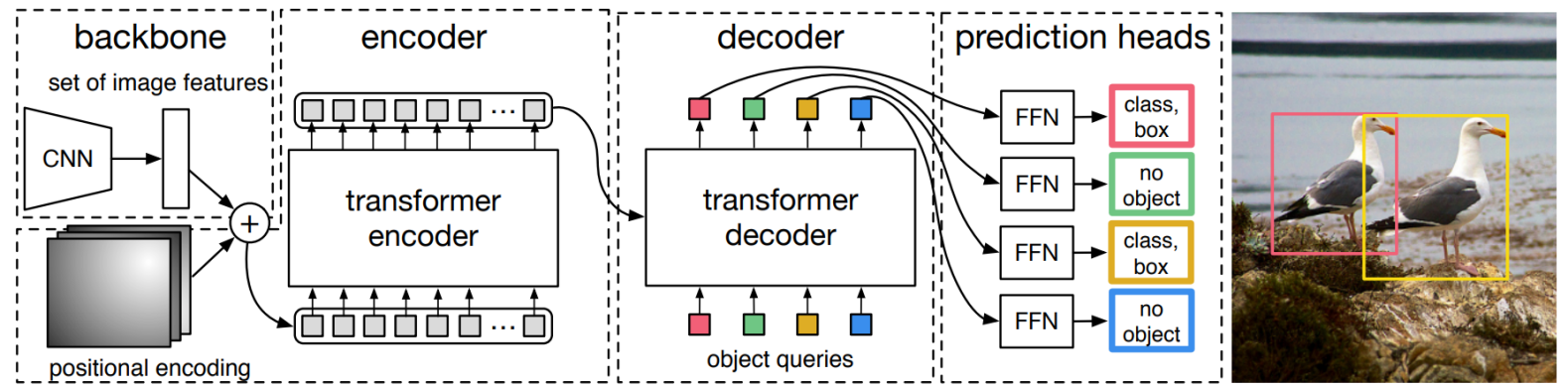
- Сегментация клеток с Cell-DETR (Prangemeier et al., 2020) на основе Detection Transformer (DETR):



Сегментация медицинских изображений

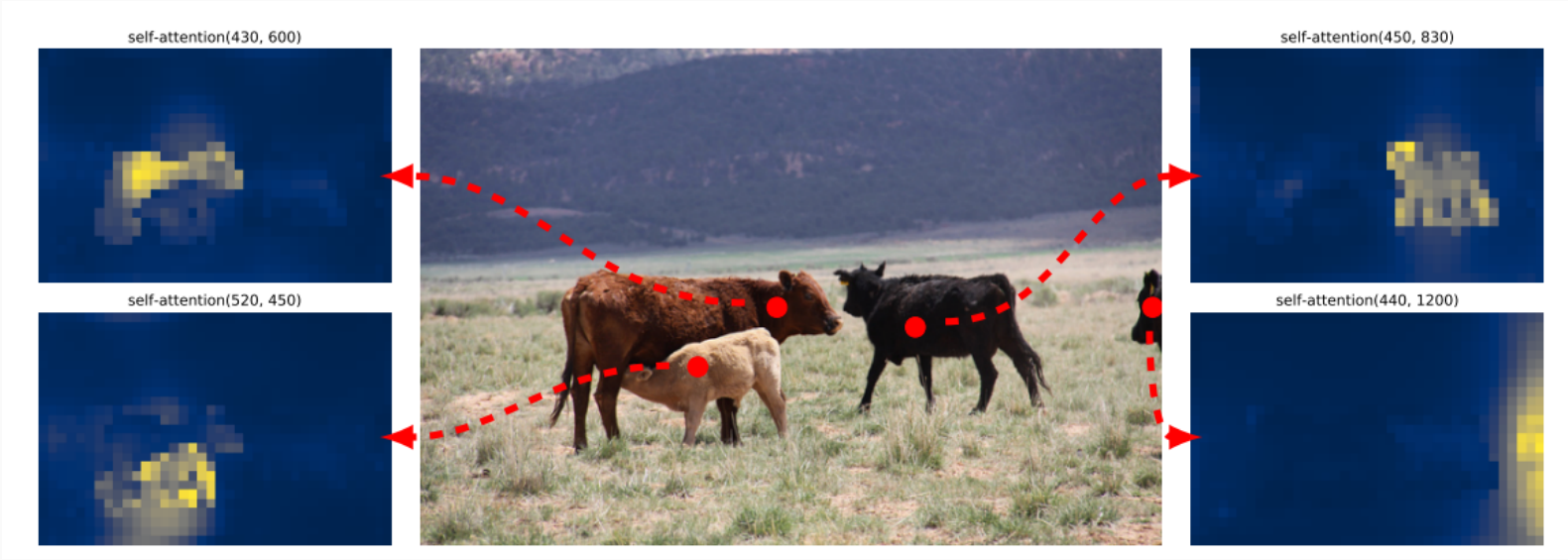
- Сам DETR (Carion et al., 2020) мы уже обсуждали:





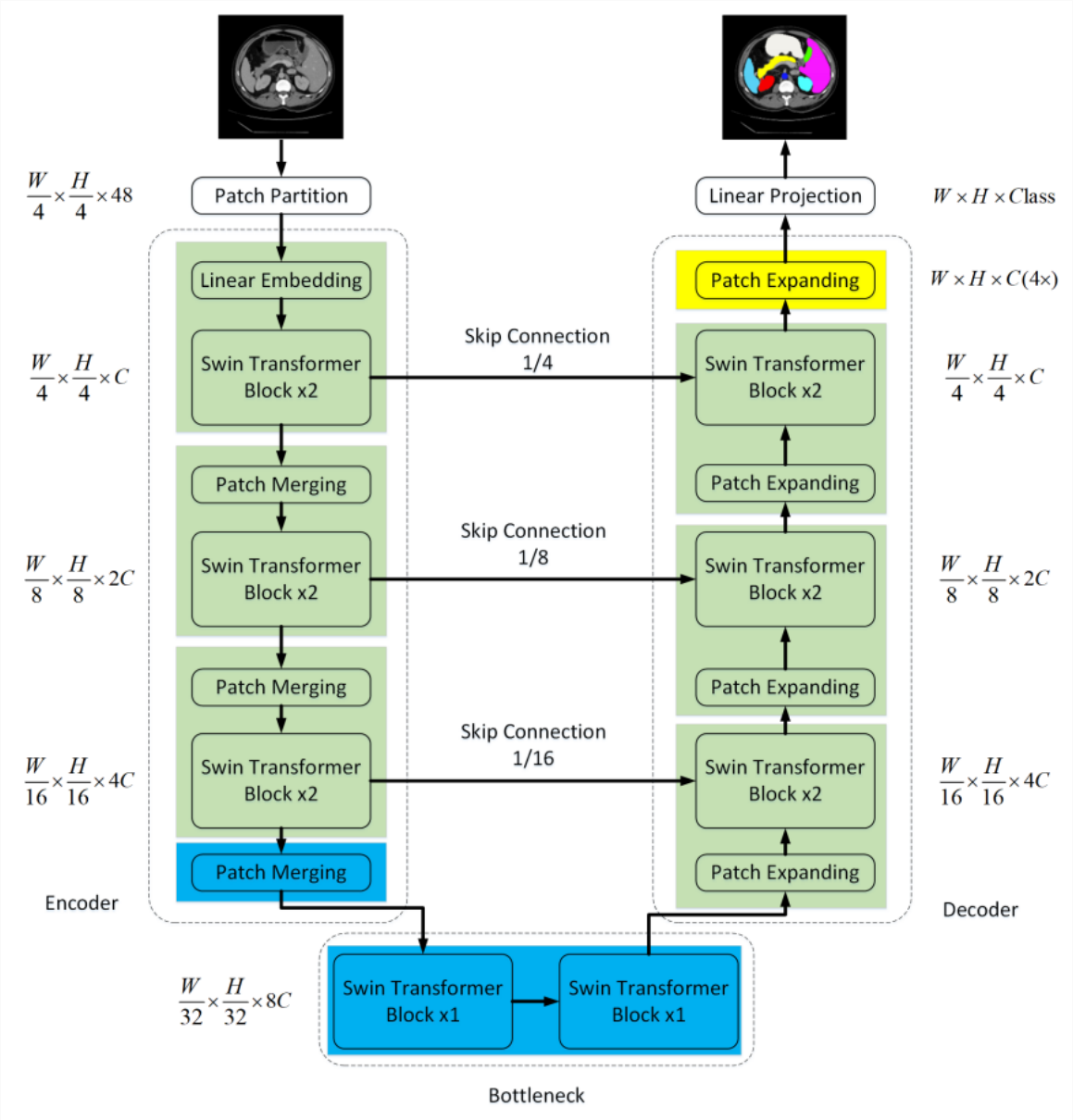
Сегментация медицинских изображений

- Карты внимания DETR хорошо разделяют экземпляры:



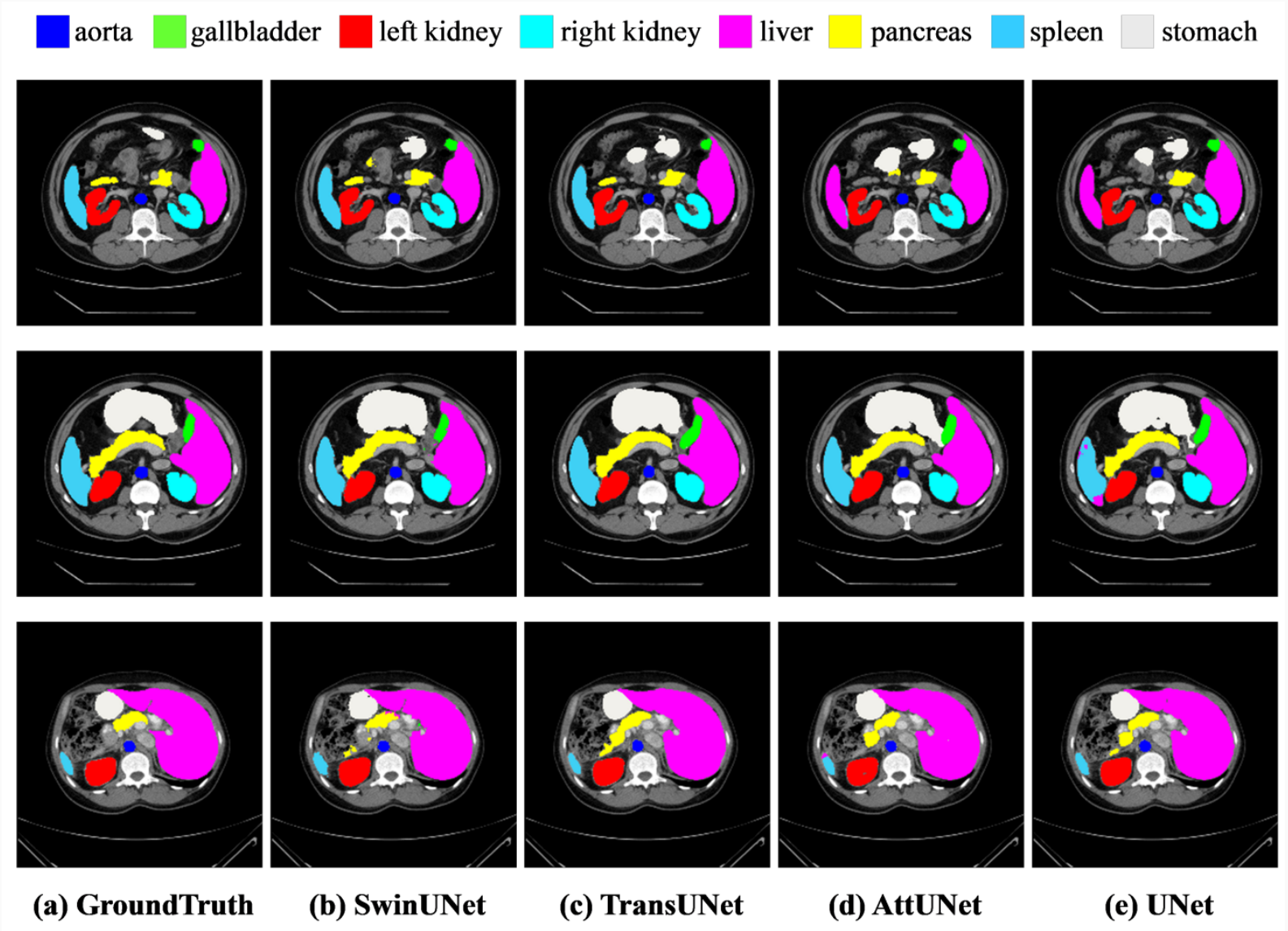
Сегментация медицинских изображений

- Swin-UNet (Cao et al., 2021):



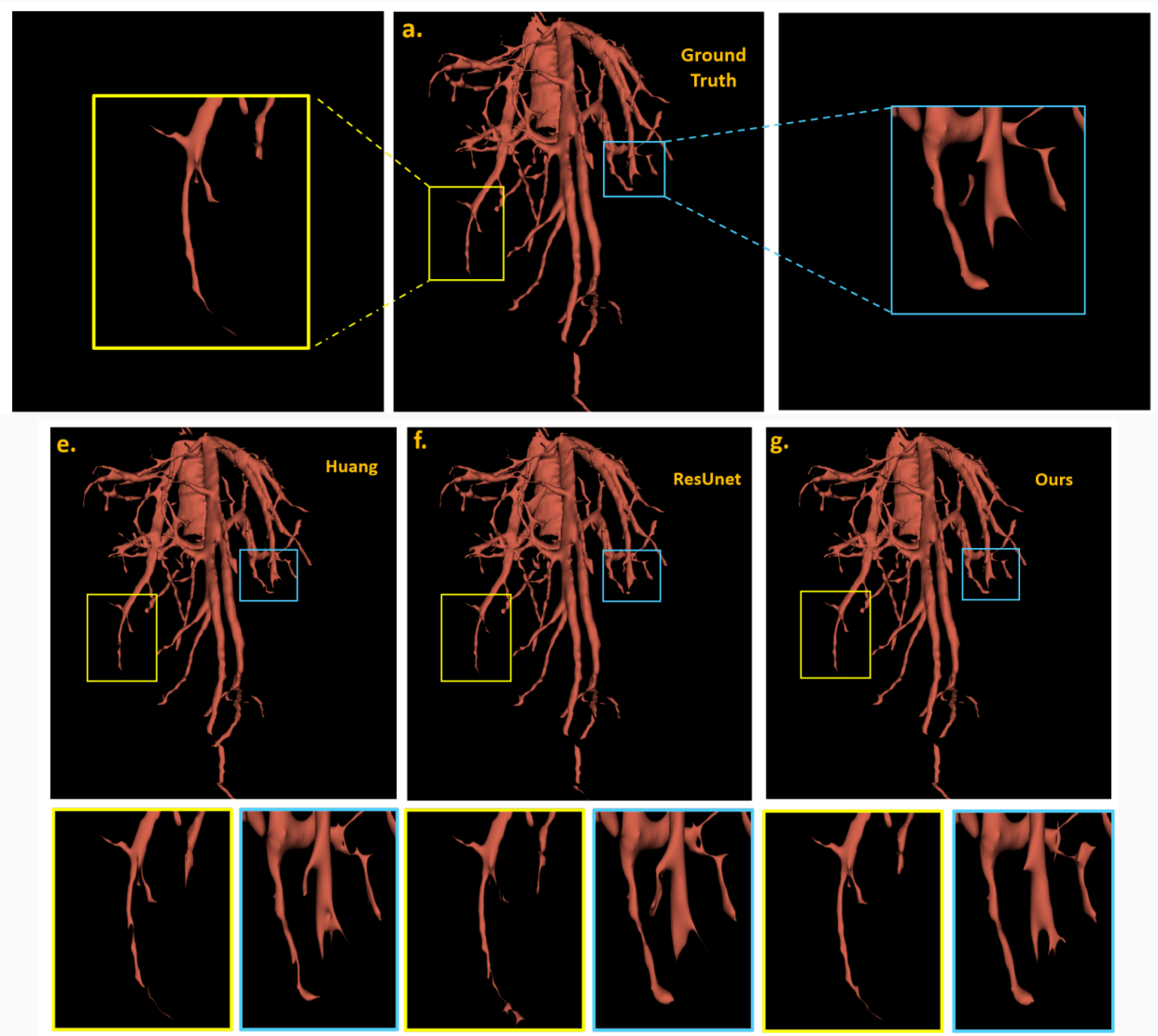
Сегментация медицинских изображений

- Дополнительно улучшает сегментацию компьютерных томограмм:



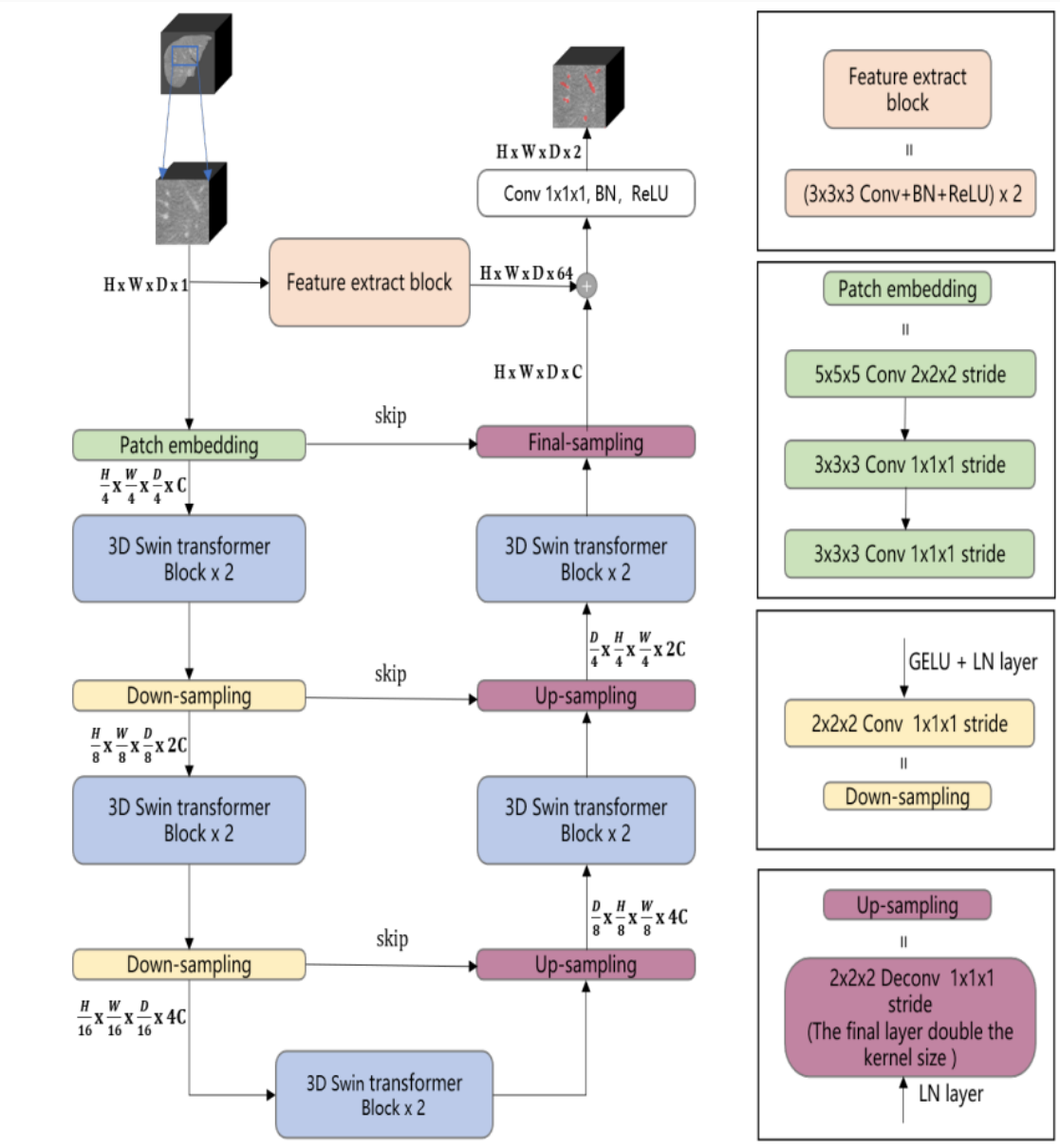
Сегментация медицинских изображений

- В 3D: сегментация печеночных сосудов на компьютерных томограммах (Wu et al., 2021)



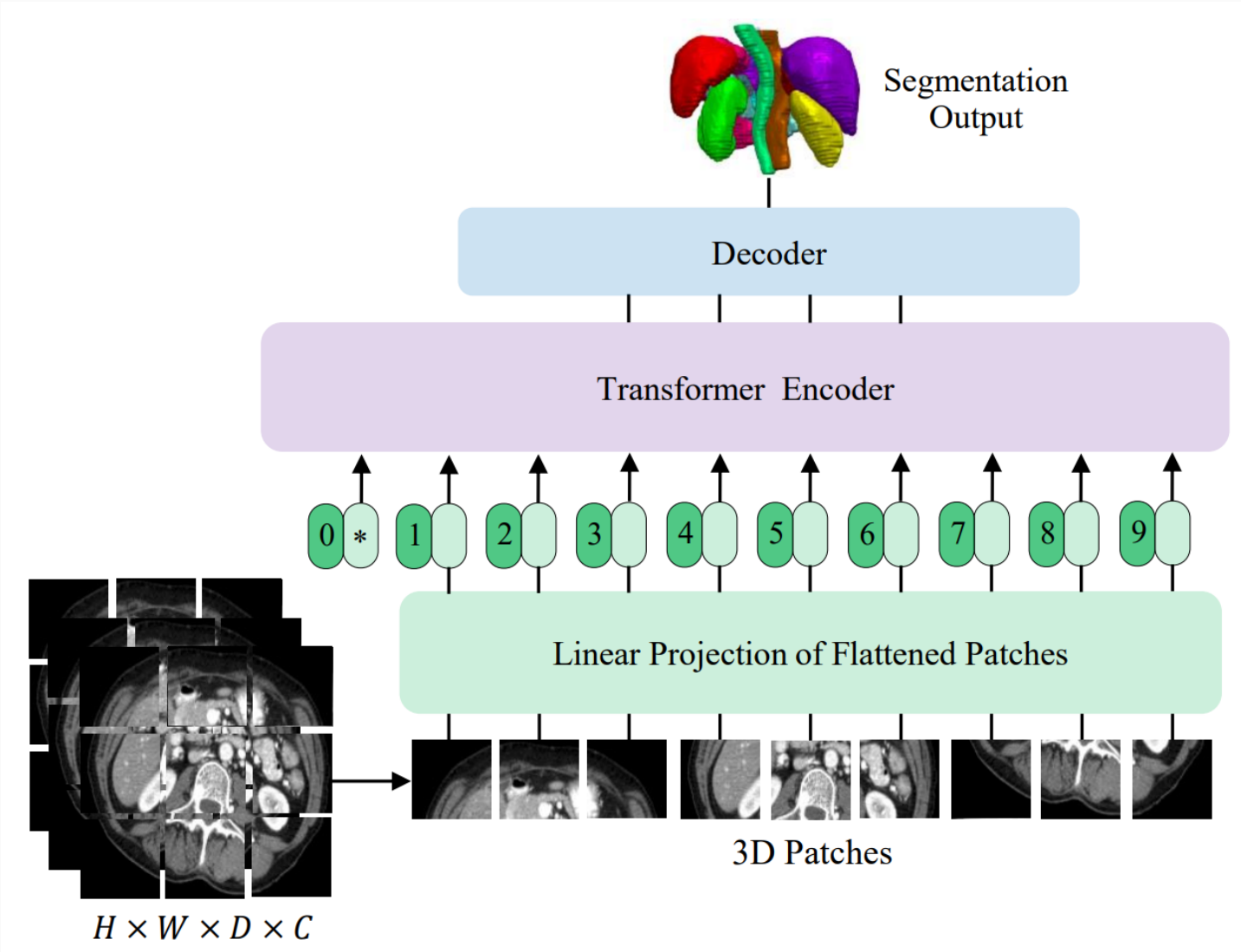
Сегментация медицинских изображений

- Архитектура основана на 3D Swin Transformer:



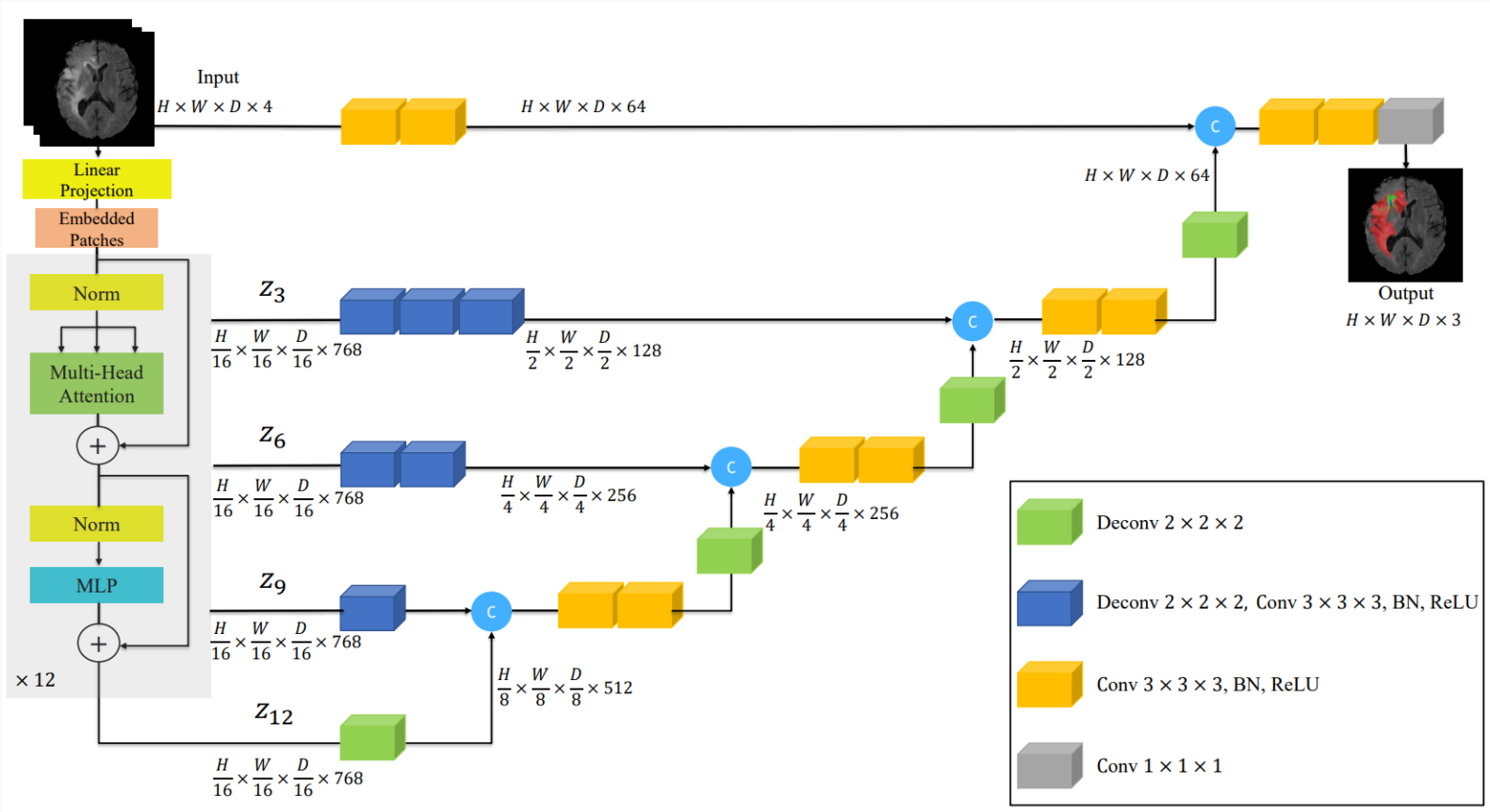
Сегментация медицинских изображений

- UNETR (Hatamizadeh et al., 2021): еще одна структура ViT+UNet для 3D сегментации



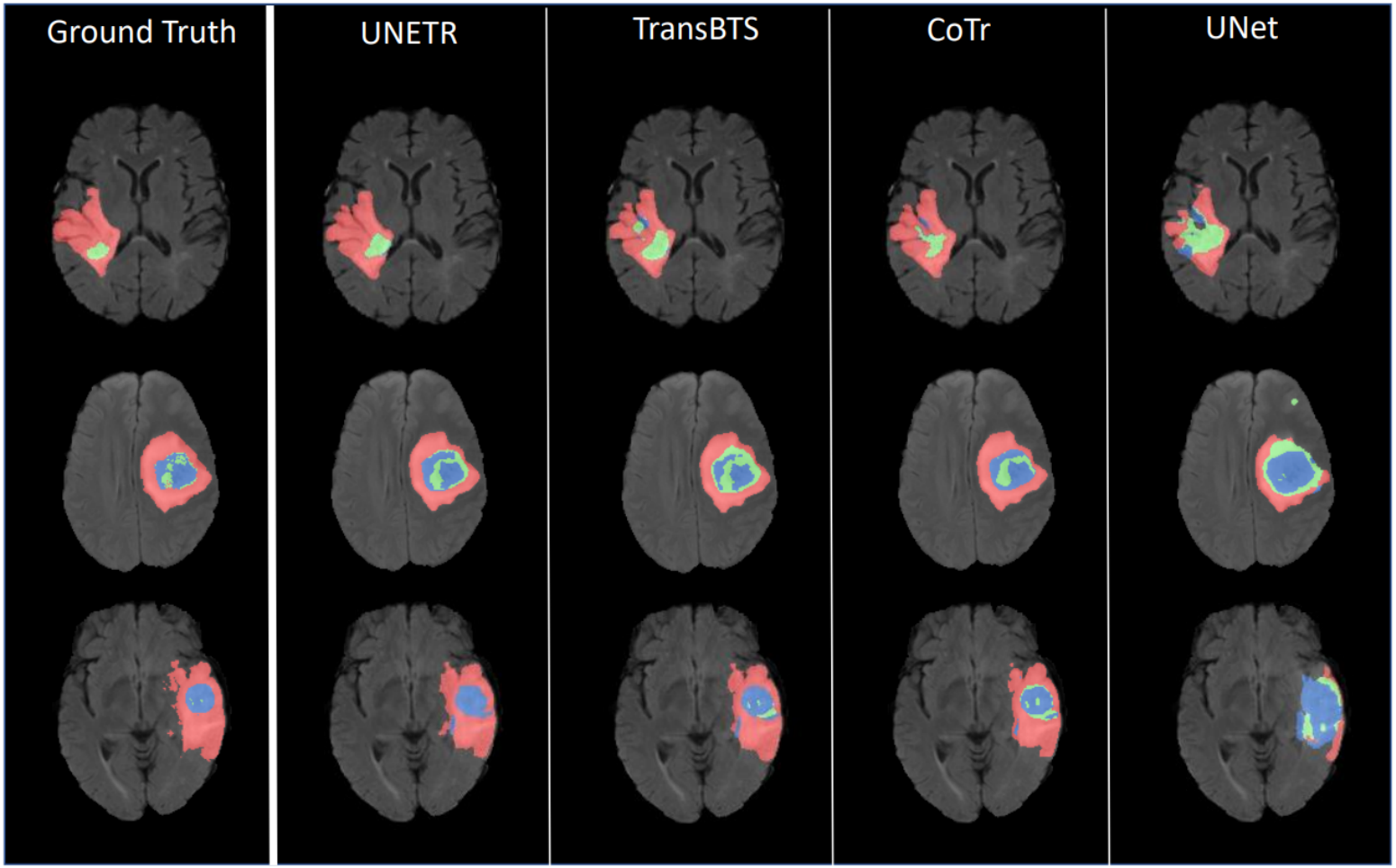
Сегментация медицинских изображений

- Представления, произведенные трансформером, объединяются в декодере для обеспечения мелких деталей, в стиле U-Net:



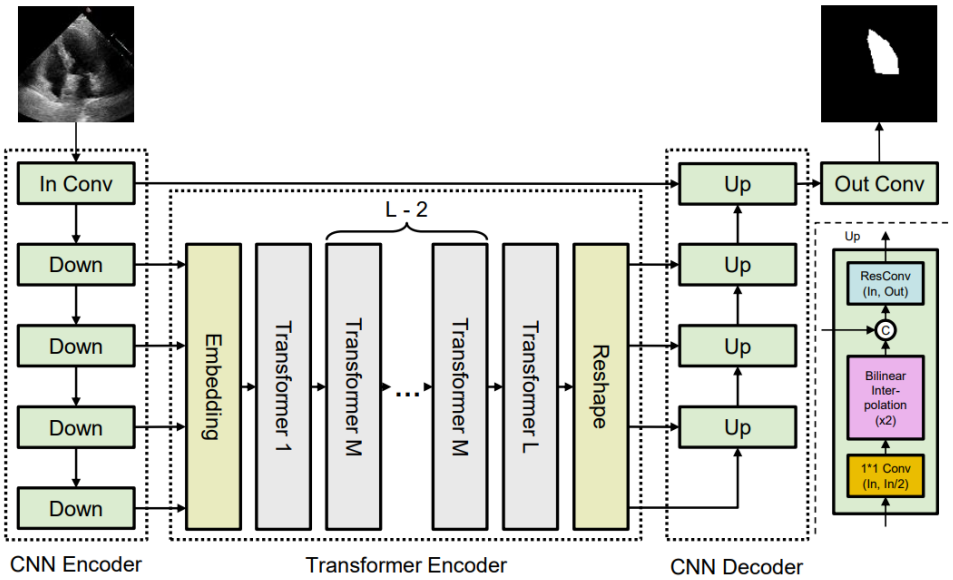
Сегментация медицинских изображений

- Результаты улучшаются:



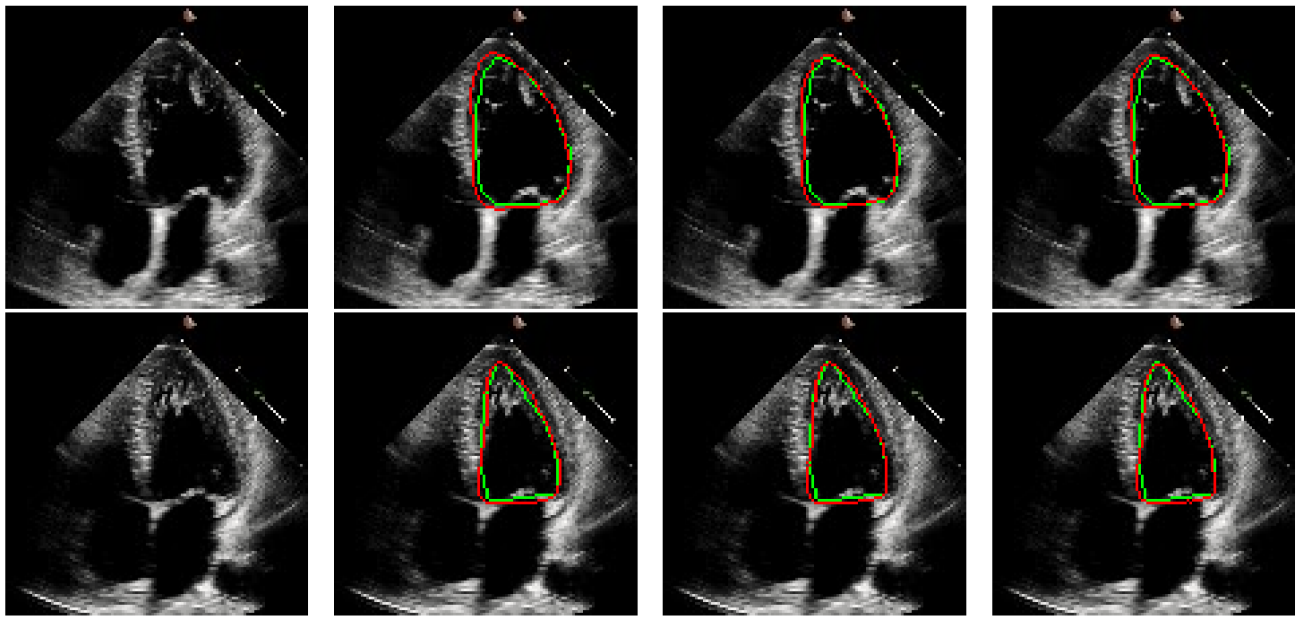
Сегментация медицинских изображений

- Transbridge (Deng et al., 2021): сегментация в электрокардиографии
- Трансформер работает как мост между свёрточным кодировщиком и свёрточным декодером



Сегментация медицинских изображений

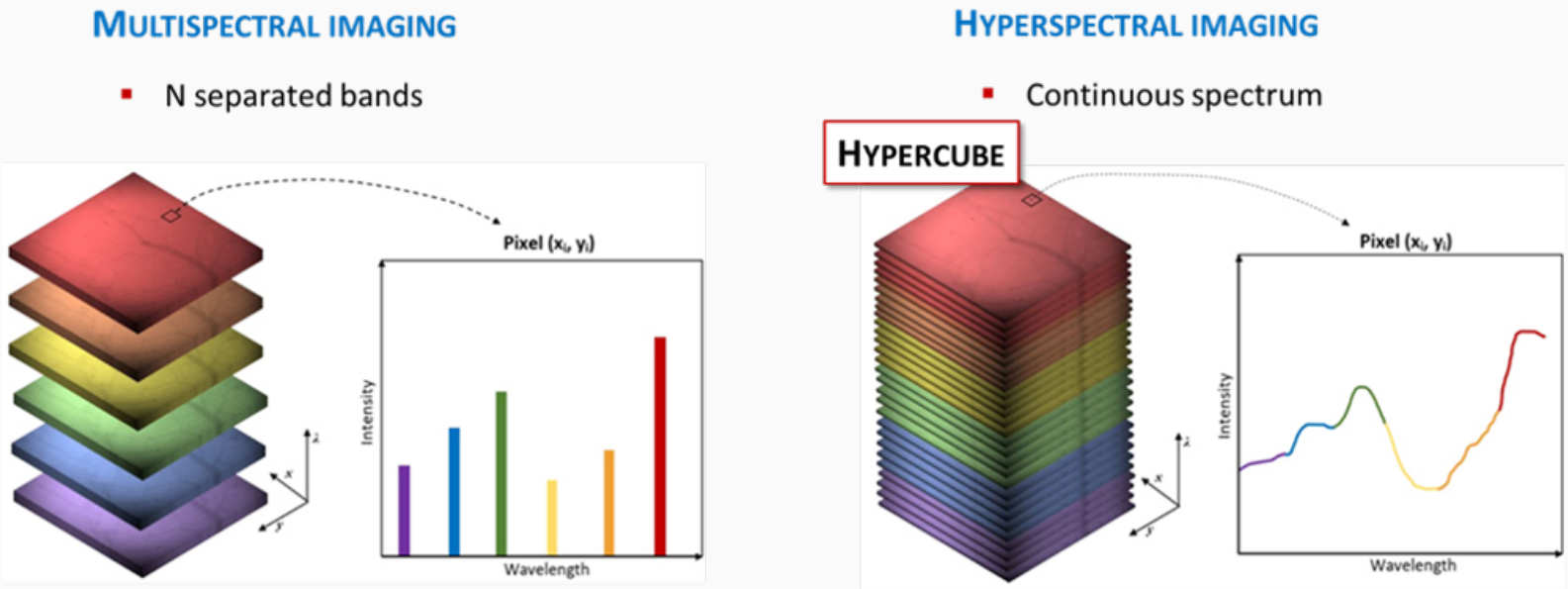
- Улучшает границы сегментации:



(a) Original image (b) CoTr (c) TransBridge-B (d) TransBridge-L

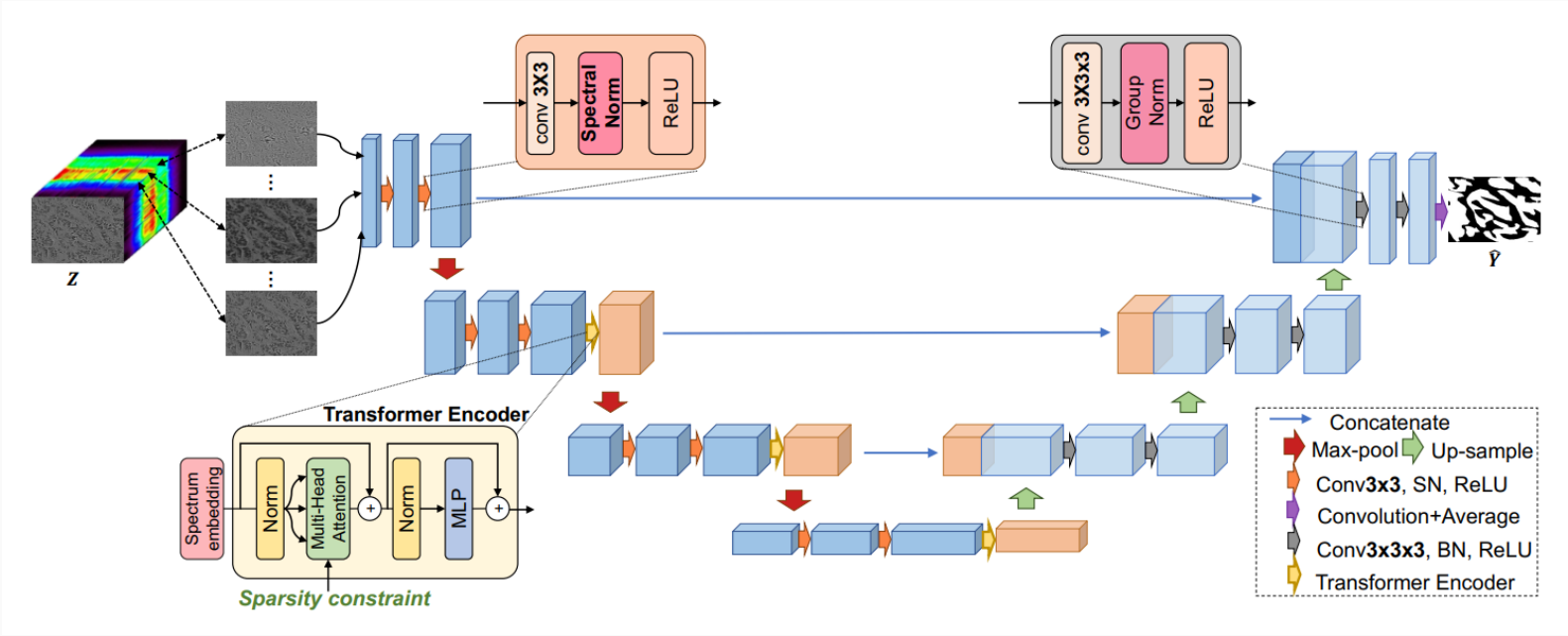
Сегментация медицинских изображений

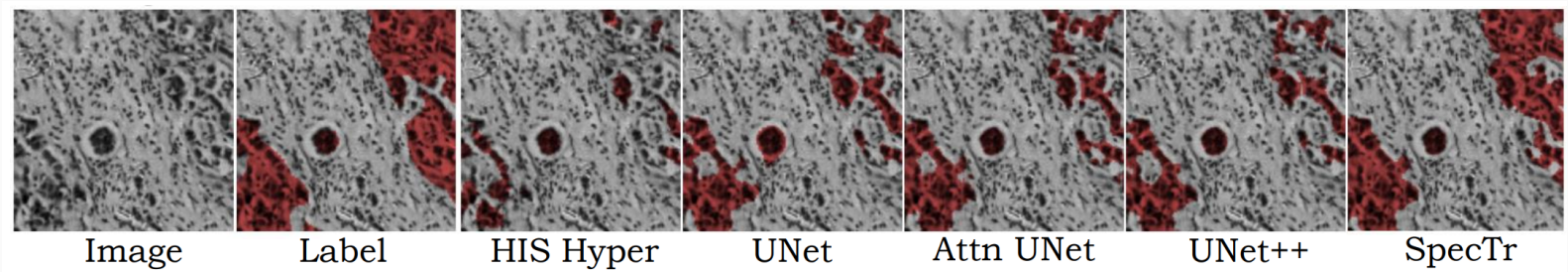
- Более того, можно не ограничиваться видимыми изображениями
- Гиперспектральная съёмка (hyperspectral imaging, HSI) изучает, как широкий спектр света взаимодействует с наблюдаемыми материалами; спектральная информация представлена сотнями узких смежных спектральных полос, гораздо больше каналов, чем просто три основных цвета
- Вход теперь – гиперкуб со множеством спектральных полос:



Сегментация медицинских изображений

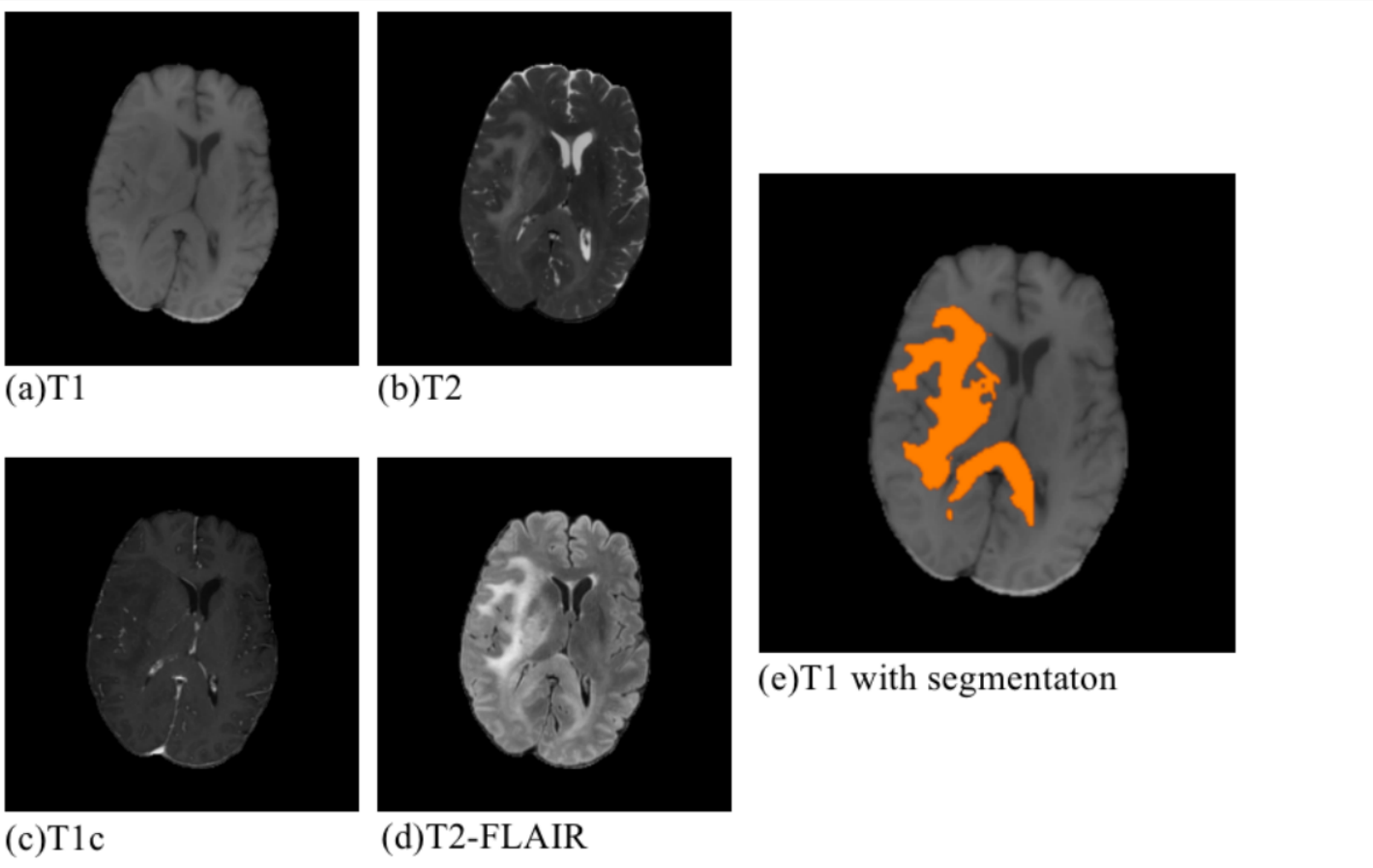
- SpecTr (Yun et al., 2021): U-Net + Transformer для гиперспектральных снимков





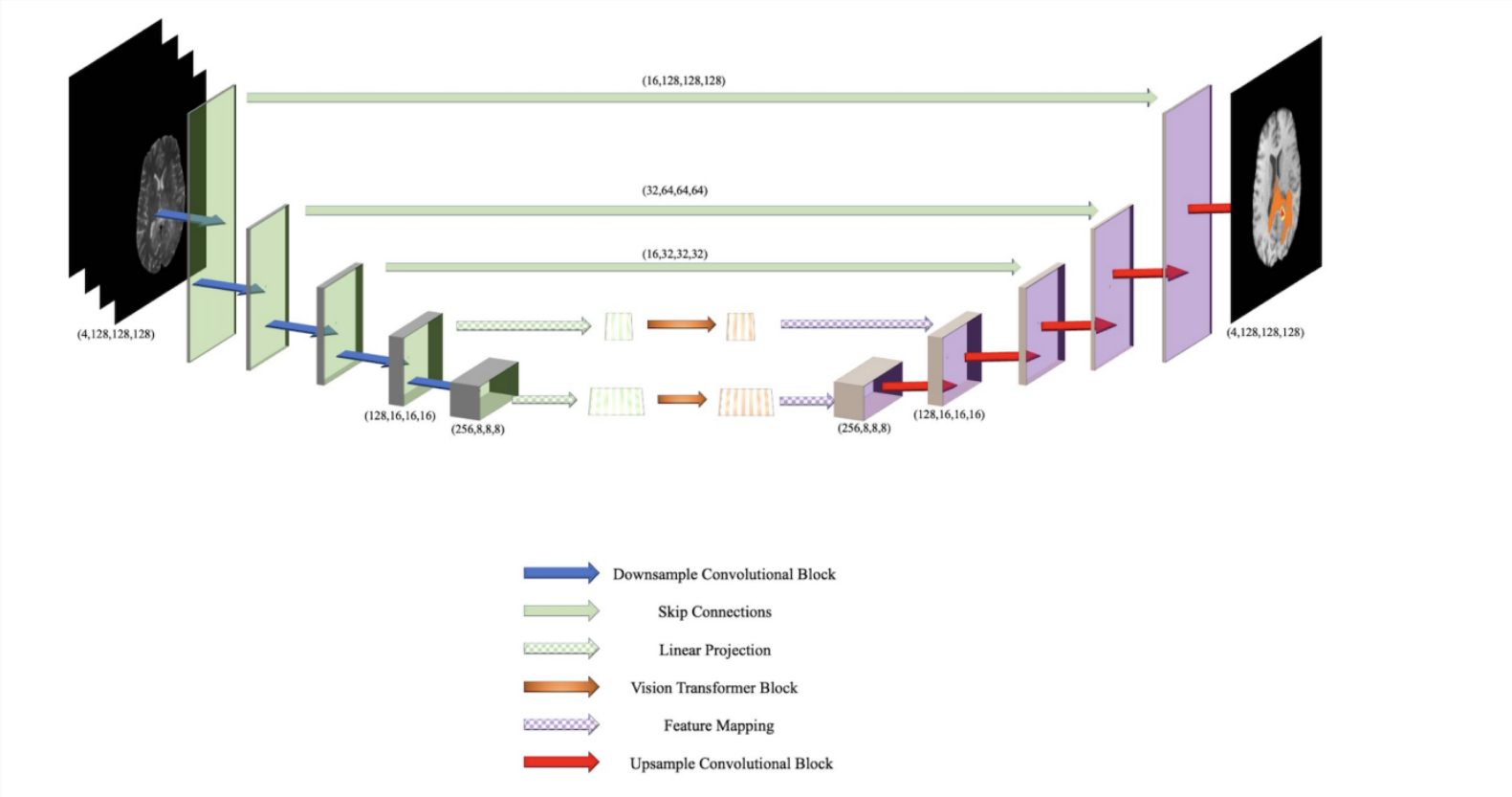
Сегментация медицинских изображений

- Другое направление – 3D сегментация
- BiTr-Unet (Jia, Shu, 2021): CNN + Transformer для сегментации опухолей мозга на МРТ



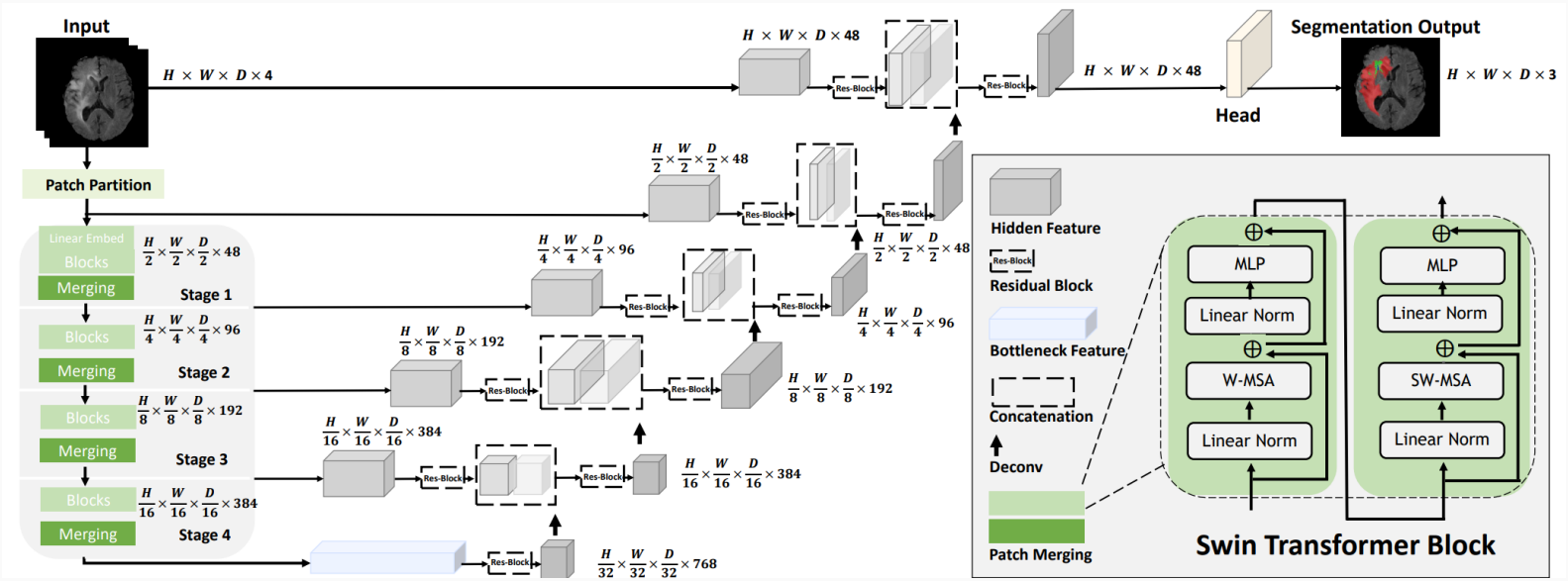
Сегментация медицинских изображений

- Снова U-Net-подобная архитектура с блоками ViT:



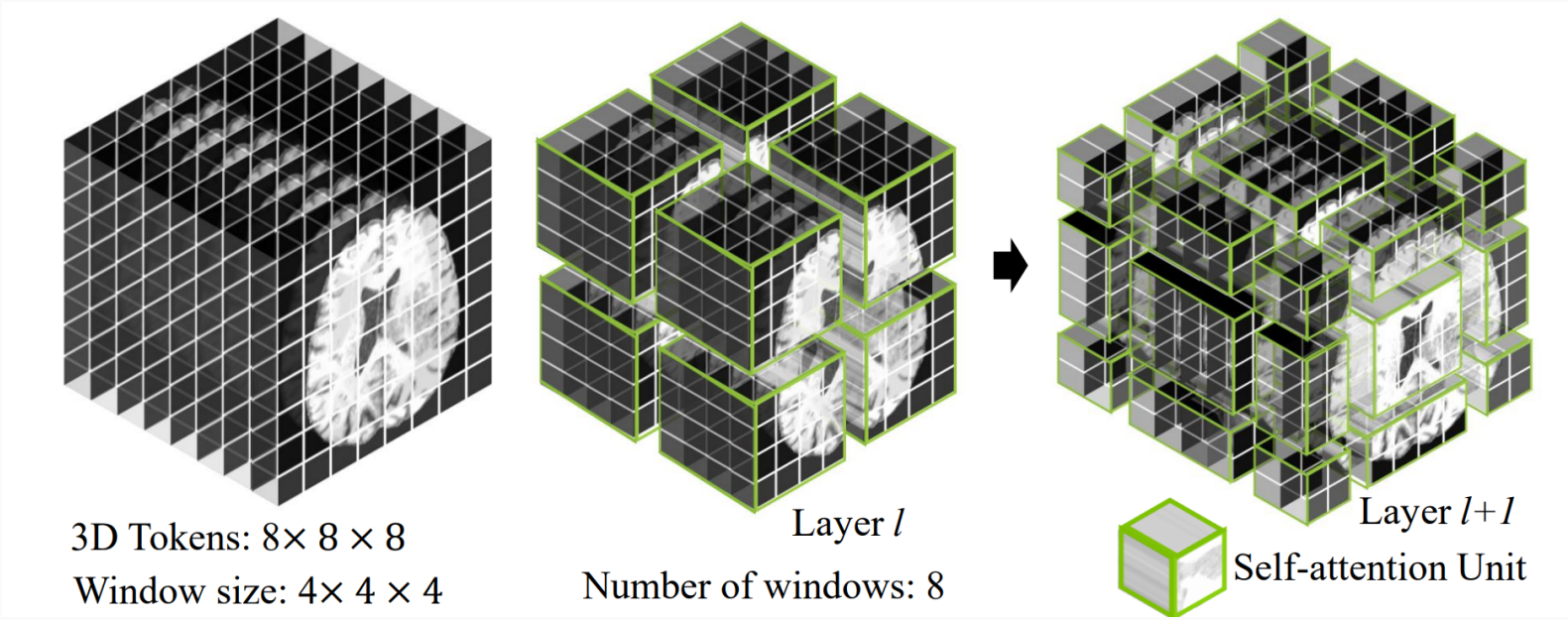
Сегментация медицинских изображений

- Swin UNETR: Swin Transformers для семантической сегментации опухолей мозга на МРТ-изображениях



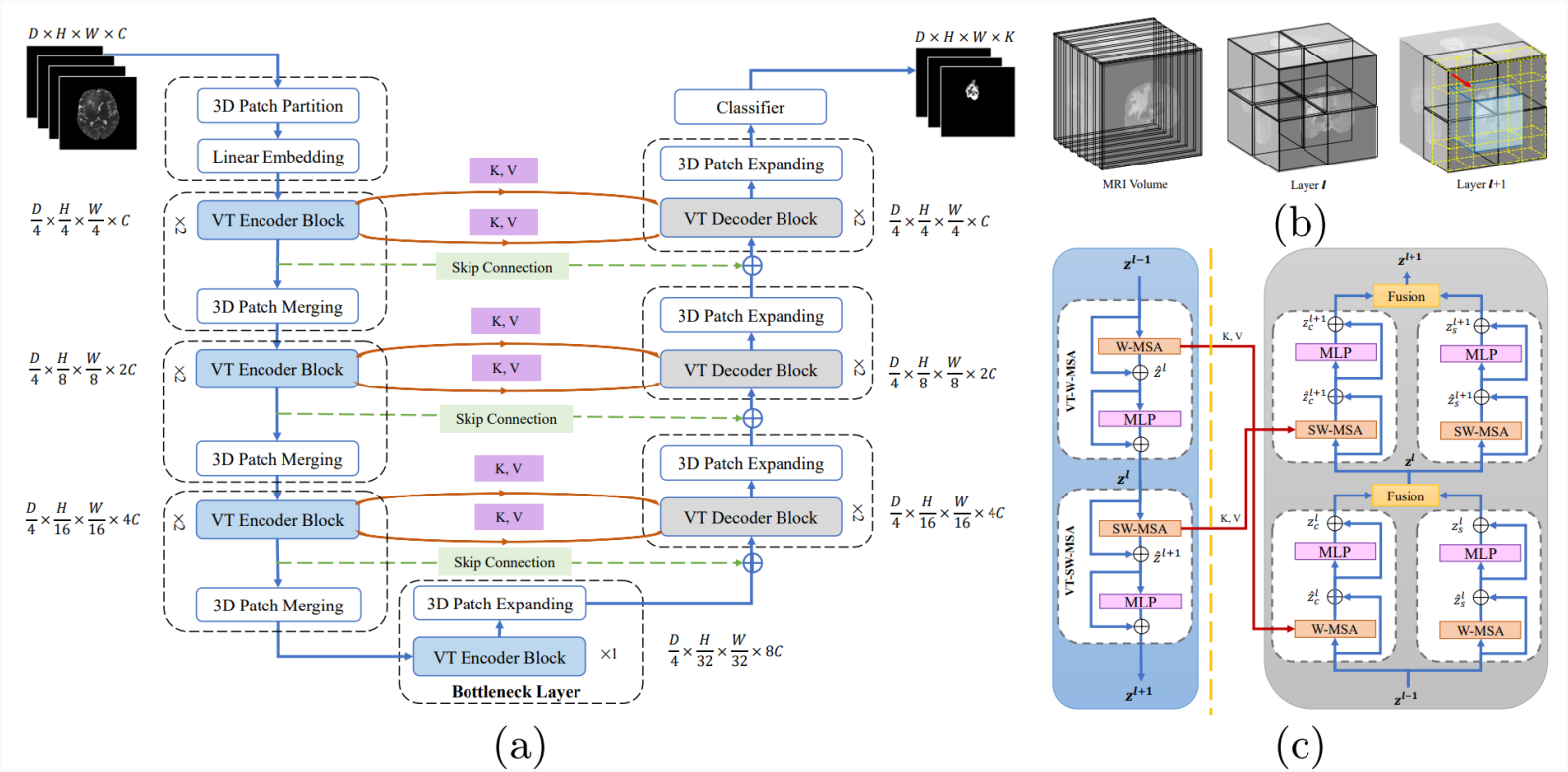
Сегментация медицинских изображений

- Swin UNETR адаптирует метод сдвинутых окон к 3D объёмным окнам:



Сегментация медицинских изображений

- VT-UNET: Transformer для объёмной сегментации (Peiris et al., 2021)



Сегментация медицинских изображений

- VT-UNET: Transformer для объёмной сегментации (Peiris et al., 2021)

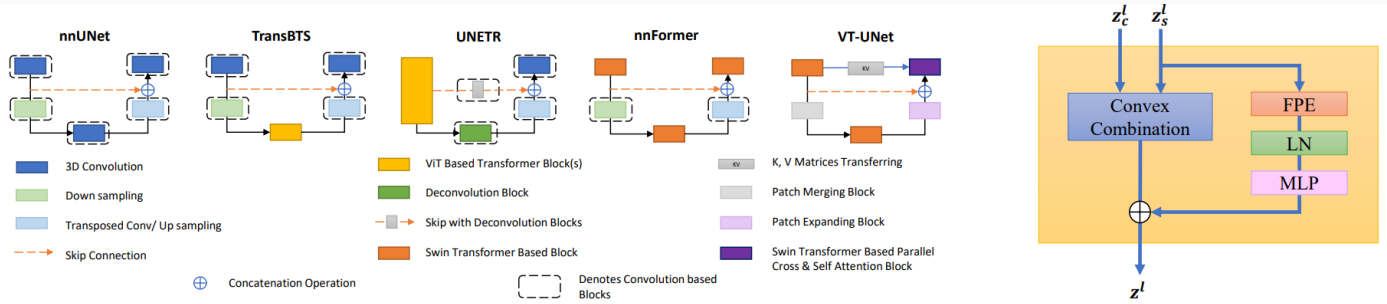
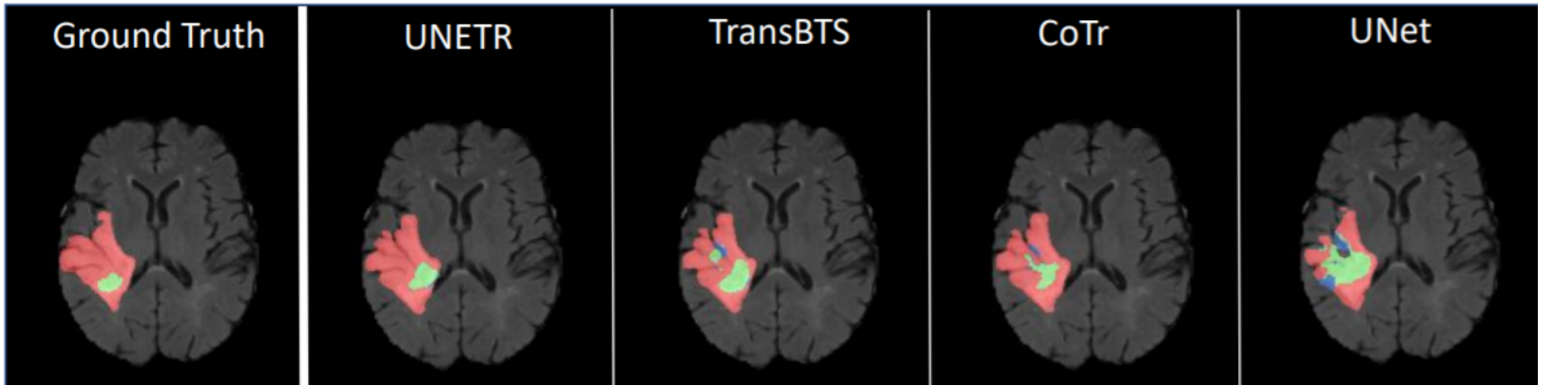


Fig. 2: (a) Illustrates VT-UNet Architecture. Here, k denotes the number of classes. (b) shows visualization of Volumetric Shifted Windows. Consider an MRI volume of size $D \times H \times W$ with $D = H = W = 8$ for the sake of illustration. Further, let the window size for partitioning the volume be $P \times M \times M$ with $P = M = 4$. Here, layer l adopts the regular window partition in the first step of Volumetric Transformer(VT) block which results in $2 \times 2 \times 2 = 8$ windows. Inside layer $l + 1$, volumetric windows are shifted by $(\frac{P}{2}, \frac{M}{2}, \frac{M}{2})$ =(2, 2, 2) tokens. This results in $3 \times 3 \times 3 = 27$ windows. (c) shows VT Encoder-Decoder Structure. (d)Encoder-Decoder structural comparison with other SOTA methods. The proposed VT-UNet architecture has no convolution modules and is purely based on Transformer blocks. (e) Illustrates the structure of the Fusion Module.

Сегментация медицинских изображений

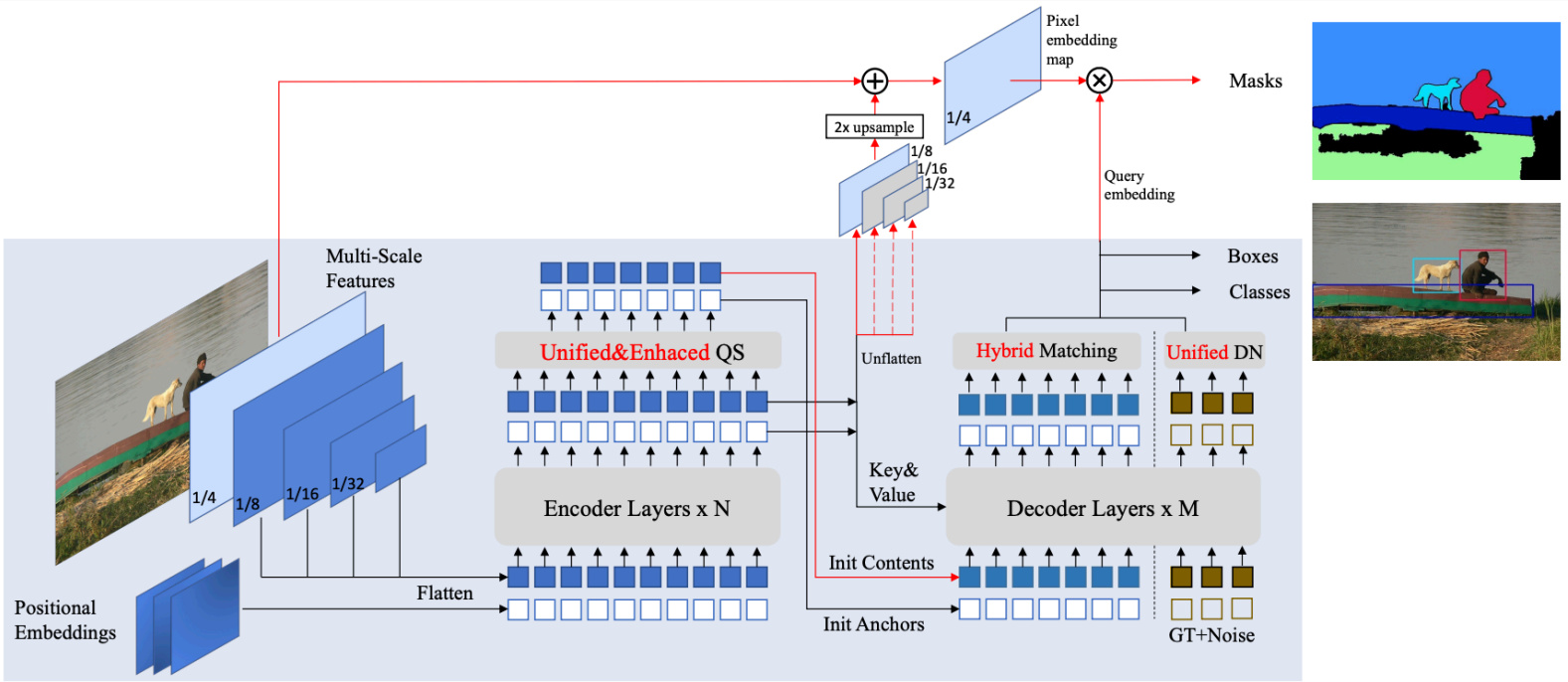
- Есть общие датасеты для сравнения; например, 3D датасет опухолей мозга BraTS 2021 (Baid et al., 2021)

Method	#params	Flops	Dice Score (Avg.)
TransBTS [138]	33 M	333 G	84.99
BiTr-UNet [139]	-	-	86.20
UNETR [35]	102.5 M	193.5 G	84.51
nnFormer [144]	39.7 M	110.7 G	86.56
Swin UNETR [145]	61.98 M	394.84 G	**88.97**
VT-UNET-T [143]	**5.4 M**	**52 G**	86.82
VT-UNET-S [143]	11.8 M	100.8 G	87.00
VT-UNET-B [143]	20.8 M	165 G	88.07



Mask DINO

- Mask DINO (Li et al., 2023): идеи Mask R-CNN можно применить и к детекторам на основе трансформеров



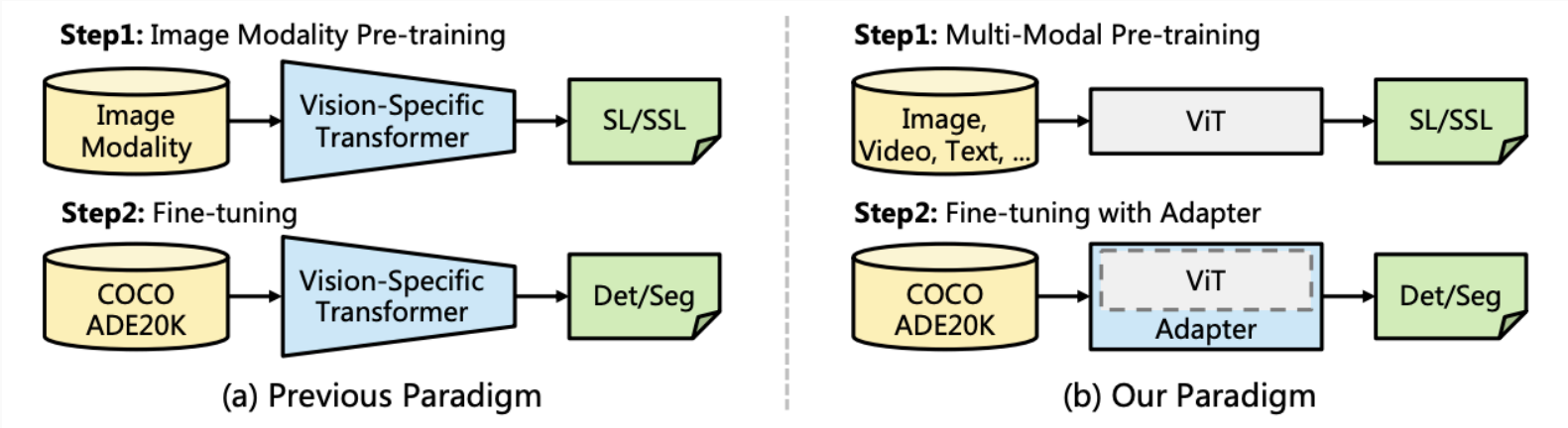
Mask DINO

- Улучшаются результаты на COCO и ADE20K

Method	Params	Backbone	Backbone Pre-training Dataset	Detection Pre-training Dataset	val	
					w/o TTA	w/ TTA
Instance segmentation on COCO					AP	
Mask2Former [3]	216M	SwinL	IN-22K-14M	–	50.1	–
Soft Teacher [36]	284M	SwinL	IN-22K-14M	O365	51.9	52.5
SwinV2-G-HTC++ [23]	3.0B	SwinV2-G	IN-22K-ext-70M [23]	O365	53.4	53.7
MasK DINO(Ours)	**223**M	SwinL	IN-22K-14M	–	**52.6**	–
MasK DINO(Ours)	**223**M	SwinL	IN-22K-14M	O365	**54.5**(+1.1)	–
Panoptic segmentation on COCO					PQ	
Panoptic SegFormer [19]	–M	SwinL	IN-22K-14M	–	55.8	–
Mask2Former [3]	216M	SwinL	IN-22K-14M	–	57.8	–
MasK DINO (ours)	223M	SwinL	IN-22K-14M		**58.4**(+0.6)	–
MasK DINO (ours)	223M	SwinL	IN-22K-14M	O365	**59.4**(+1.6)	–
Semantic segmentation on ADE20K					mIoU	
Mask2Former [3]	215M	SwinL	IN-22K-14M	–	56.1	57.3
SeMask-L MSFaPN-Mask2Former [14]	–M	SwinL-FaPN	IN-22K-14M	–	–	58.2
SwinV2-G-UperNet [23]	3.0B	SwinV2-G	IN-22K-ext-70M [23]	–	59.3	59.9
MasK DINO (ours)	**223M**	SwinL	IN-22K-14M	–	**56.6**	–
MasK DINO (ours)	**223M**	SwinL	IN-22K-14M	O365	**59.5**	**60.8**(+0.9)

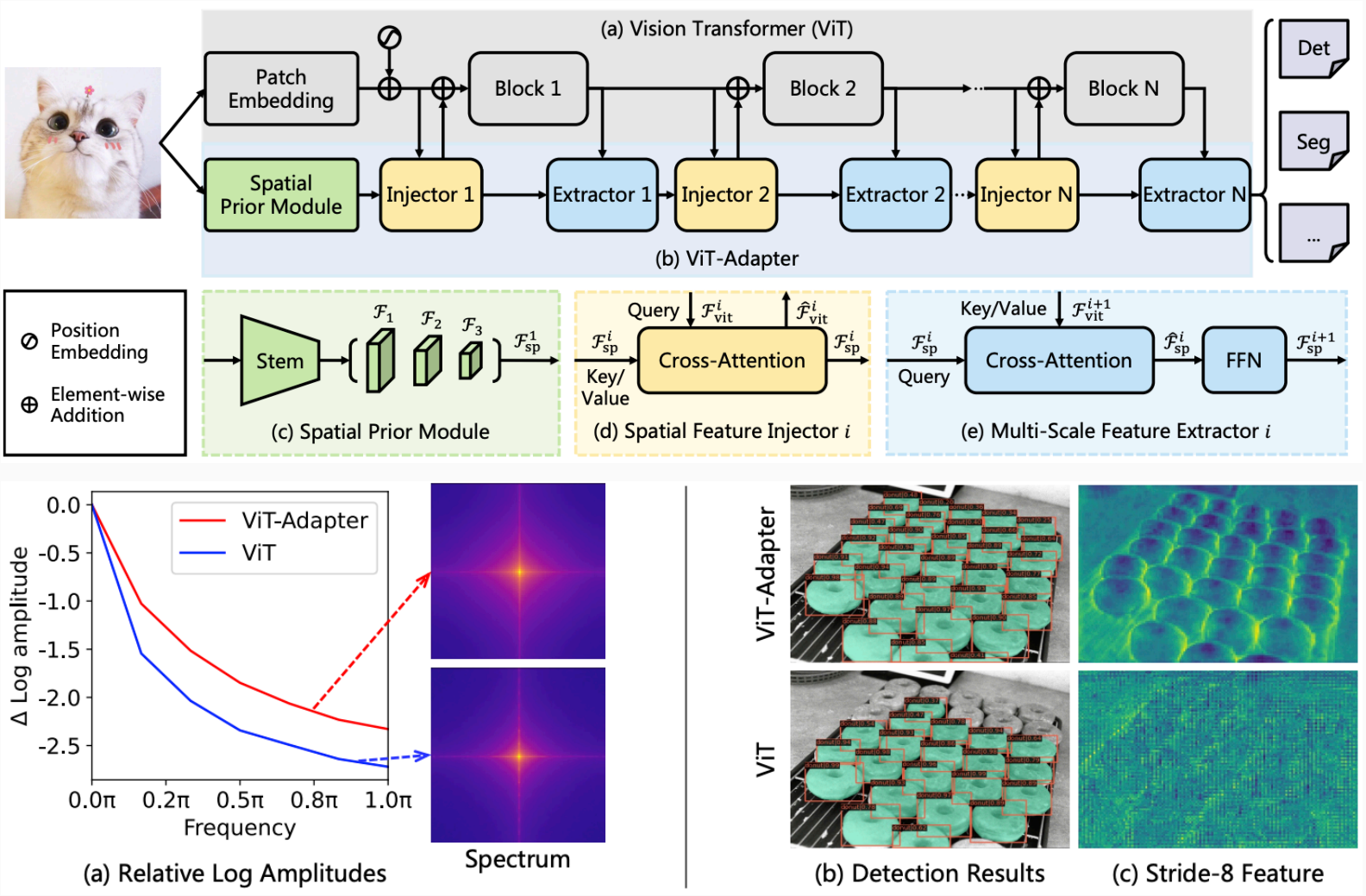
ViT-Adapter

- ViT-Adapter (Chen et al., 2023): адаптер без предобучения, чтобы приблизить ViT к Swin для плотного предсказания



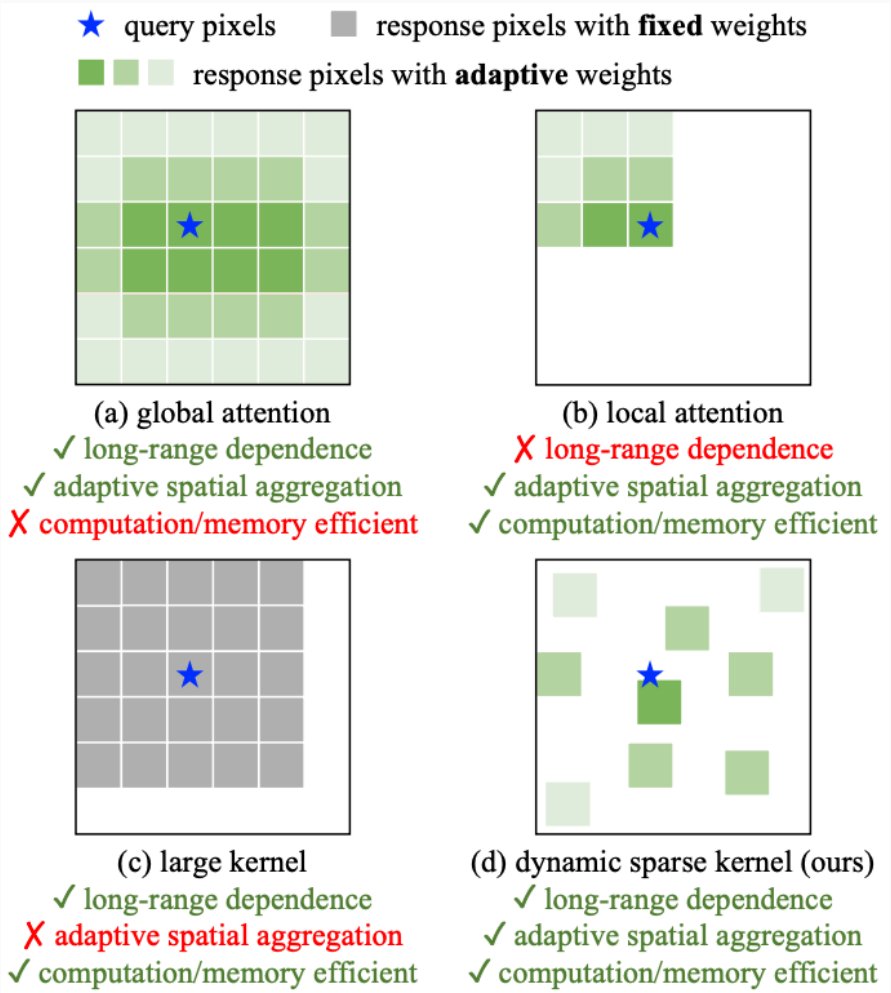
ViT-Adapter

- Адаптер позволяет ViT работать с сегментацией



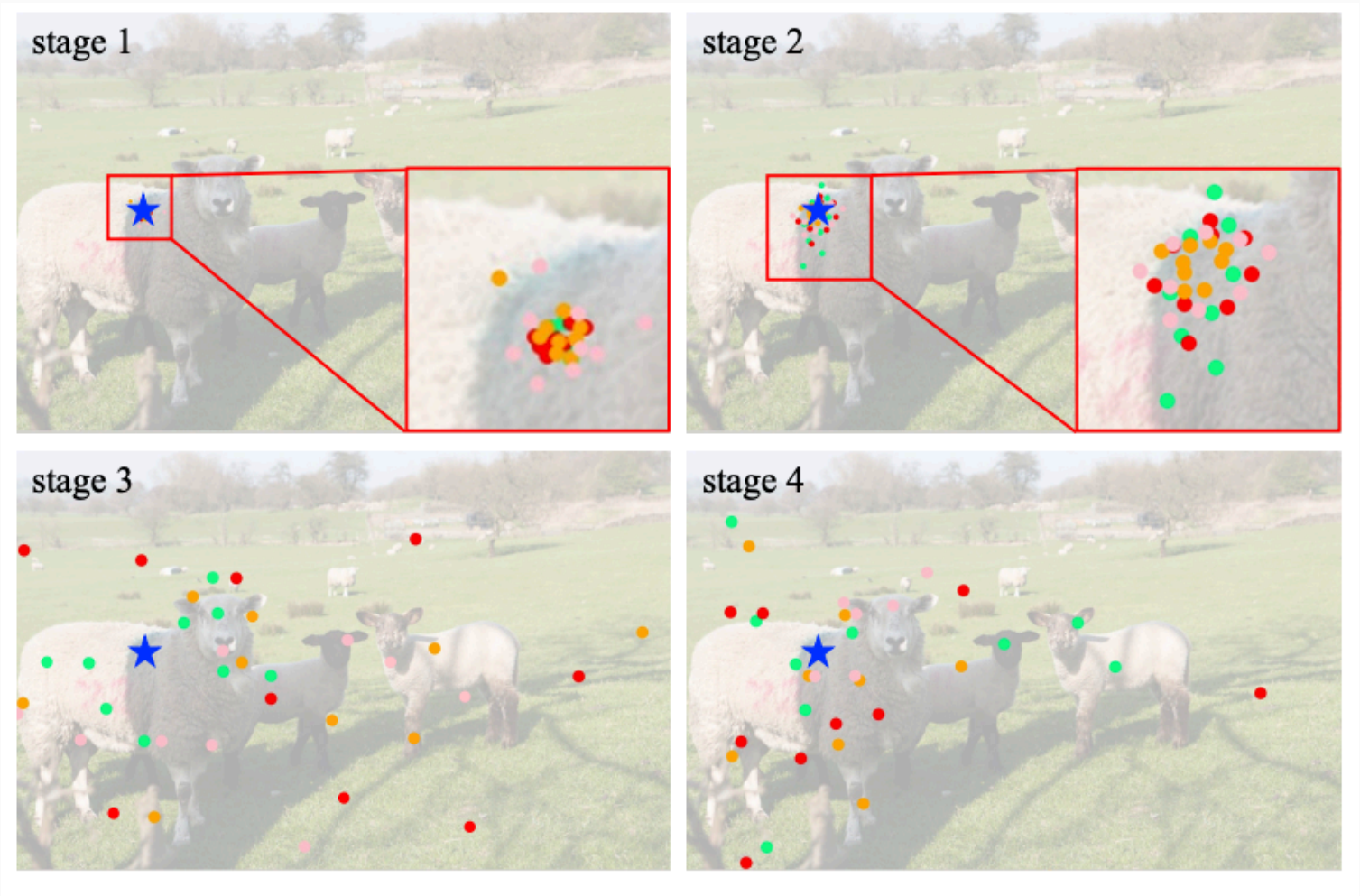
InternImage

- CNN не умерли! InternImage (Wang et al., 2023) – новая базовая модель на основе CNN, использующая деформируемые свертки



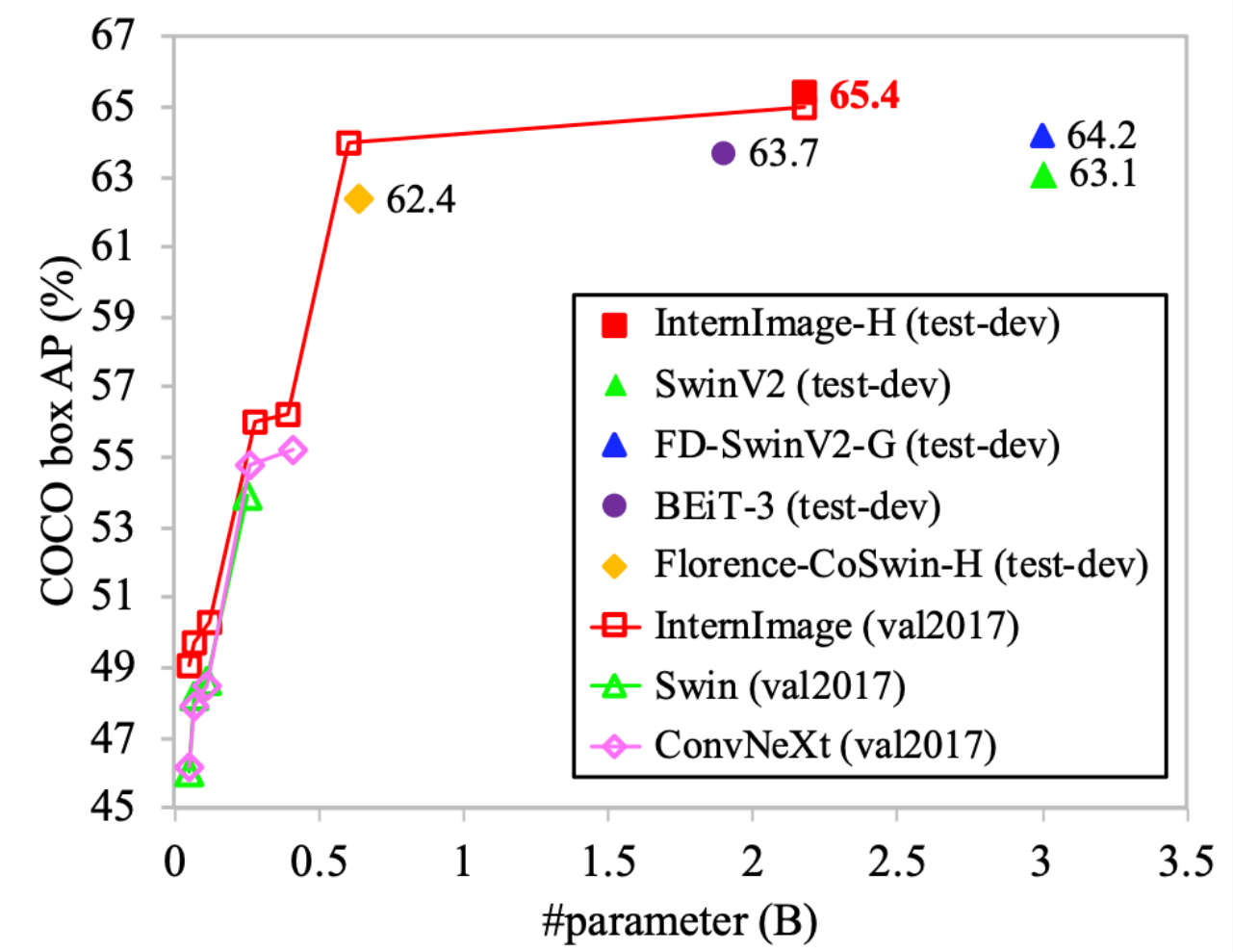
InternImage

- Новая идея – иметь многогрупповую структуру, как разные головы внимания: разные группы имеют разные смещения в свертках и действительно получают разную семантику



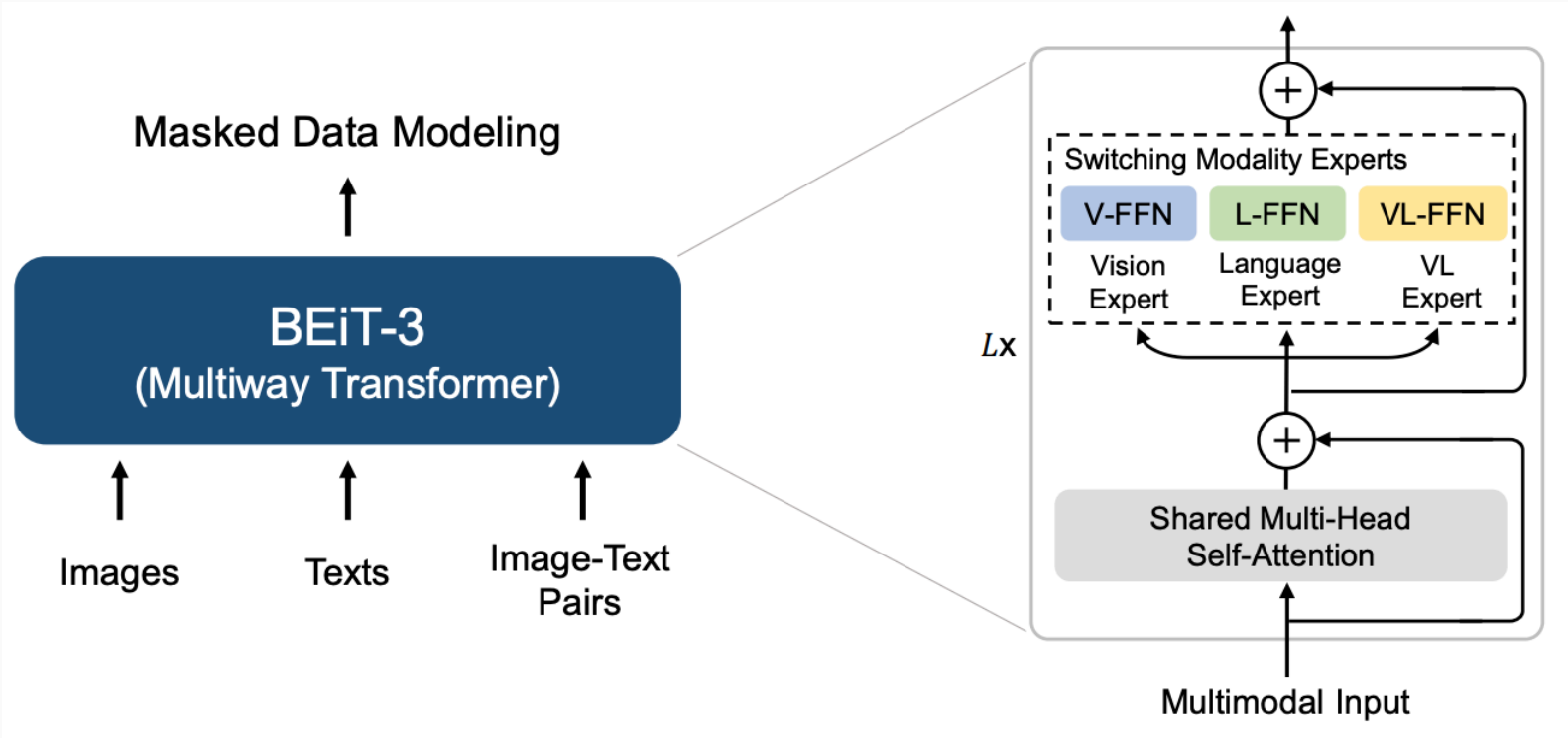
InternImage

- Хорошо и для распознавания объектов:



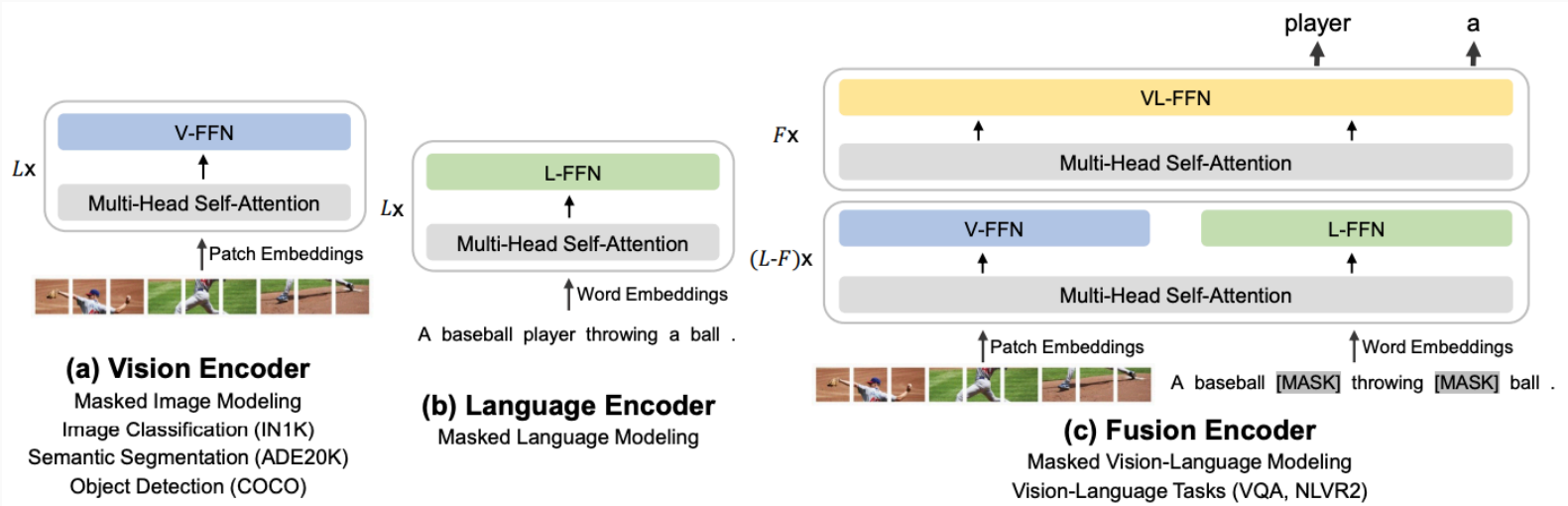
BEiT

- Еще одна большая область – мультимодальные модели; они могут быть очень эффективны для сегментации
- BEiT-3 (Wang et al., 2022): сходятся вместе предобучение языка, визуальные данныы и мультимодальность

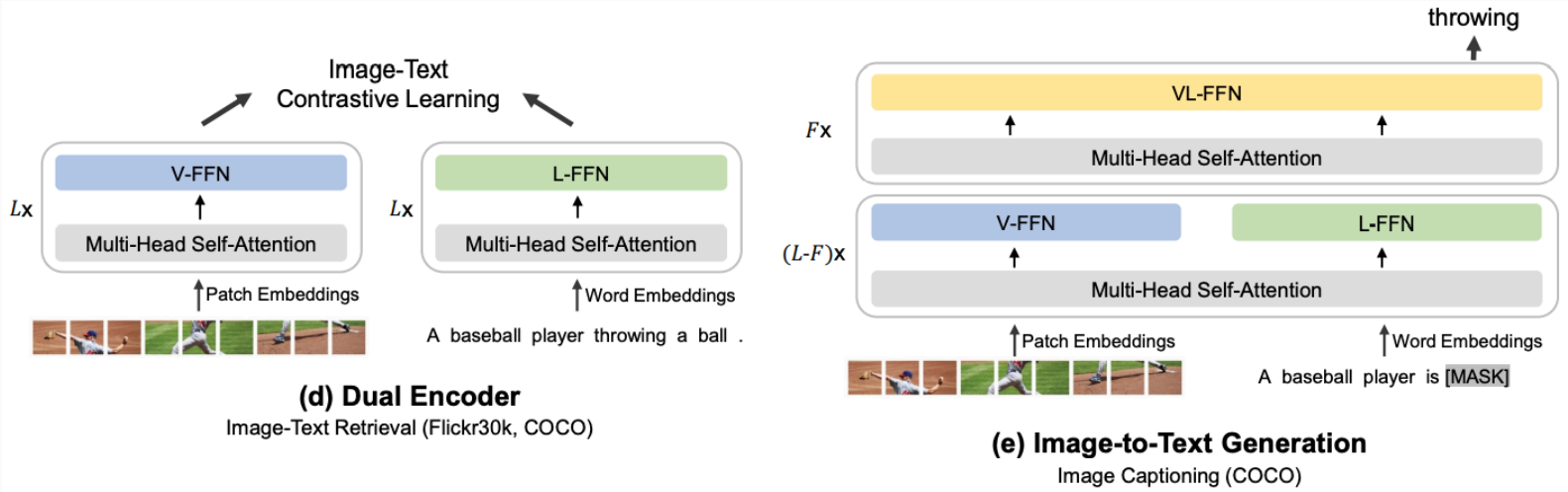


BEiT

- BEiT выполняет маскированное визуально-языковое моделирование во многих модальностях:

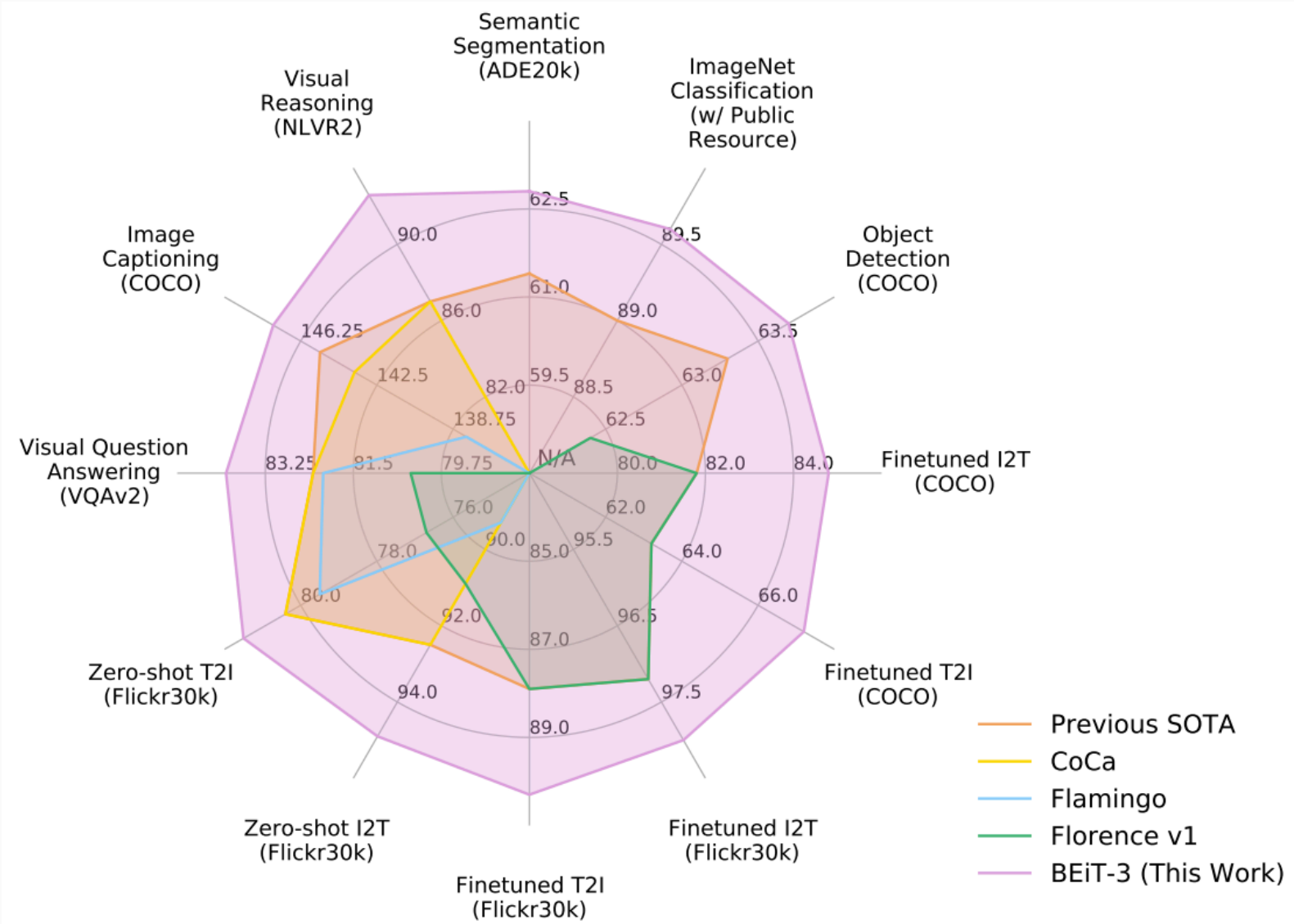


- А также поиск и порождение:



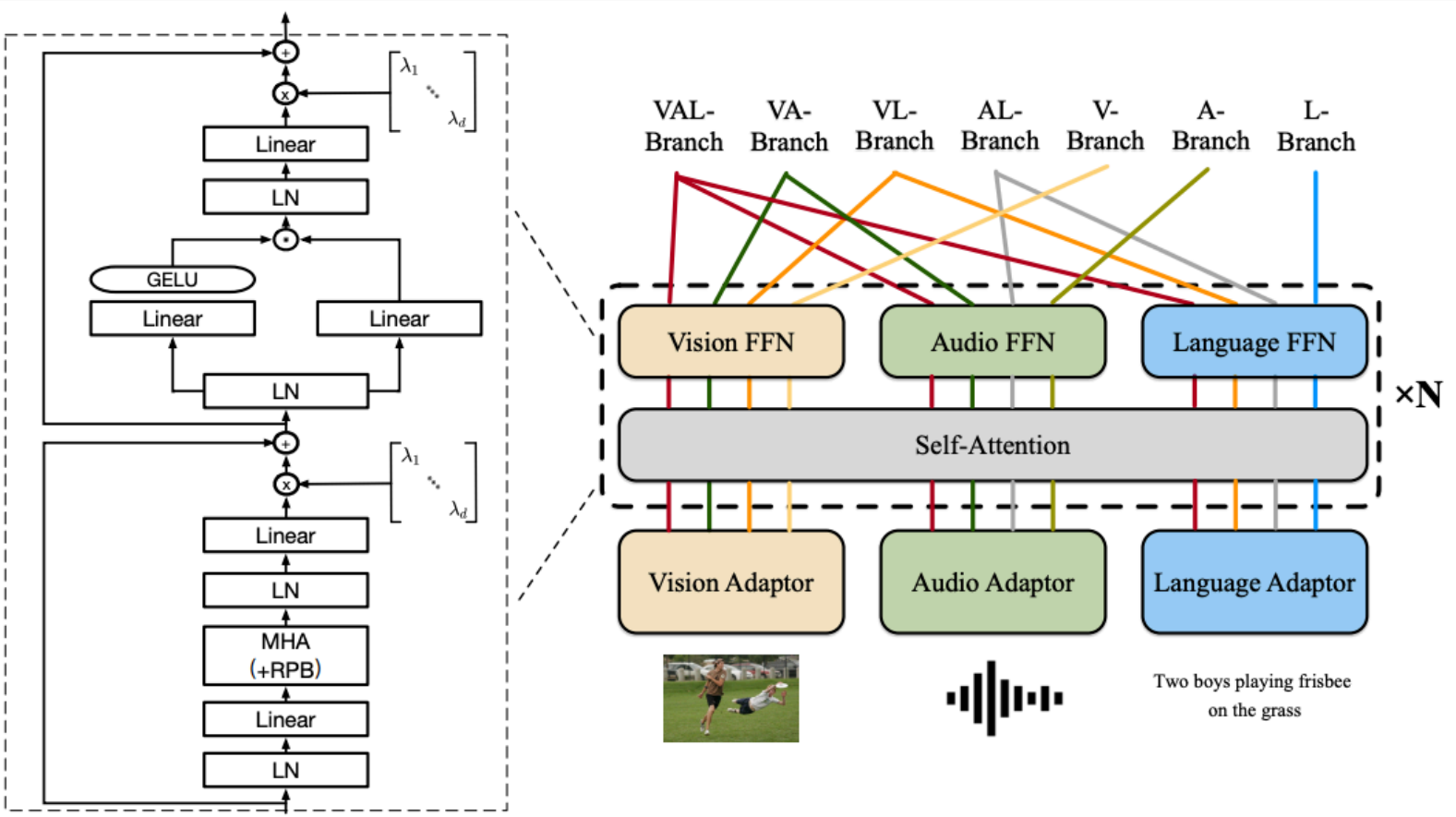
BEiT

- Улучшил многие результаты (в 2022 году):



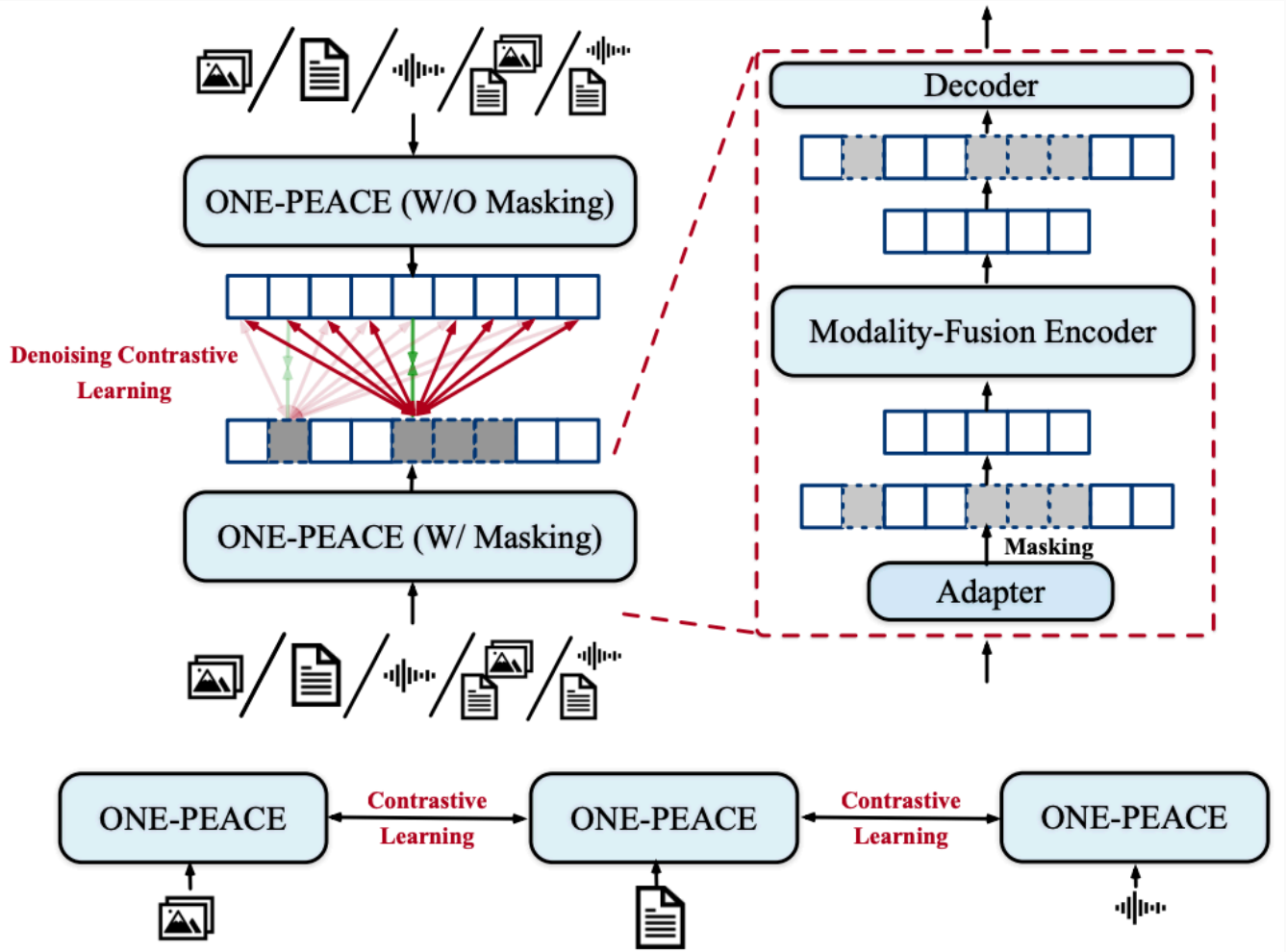
ONE-PEACE

- ONE-PEACE (Wang et al., 2023)



ONE-PEACE

- Задача здесь обычно в том, чтобы изобрести множество задач с self-supervision во всех модальностях; и это хороший момент, чтобы перейти к обсуждению мультимодальных представлений



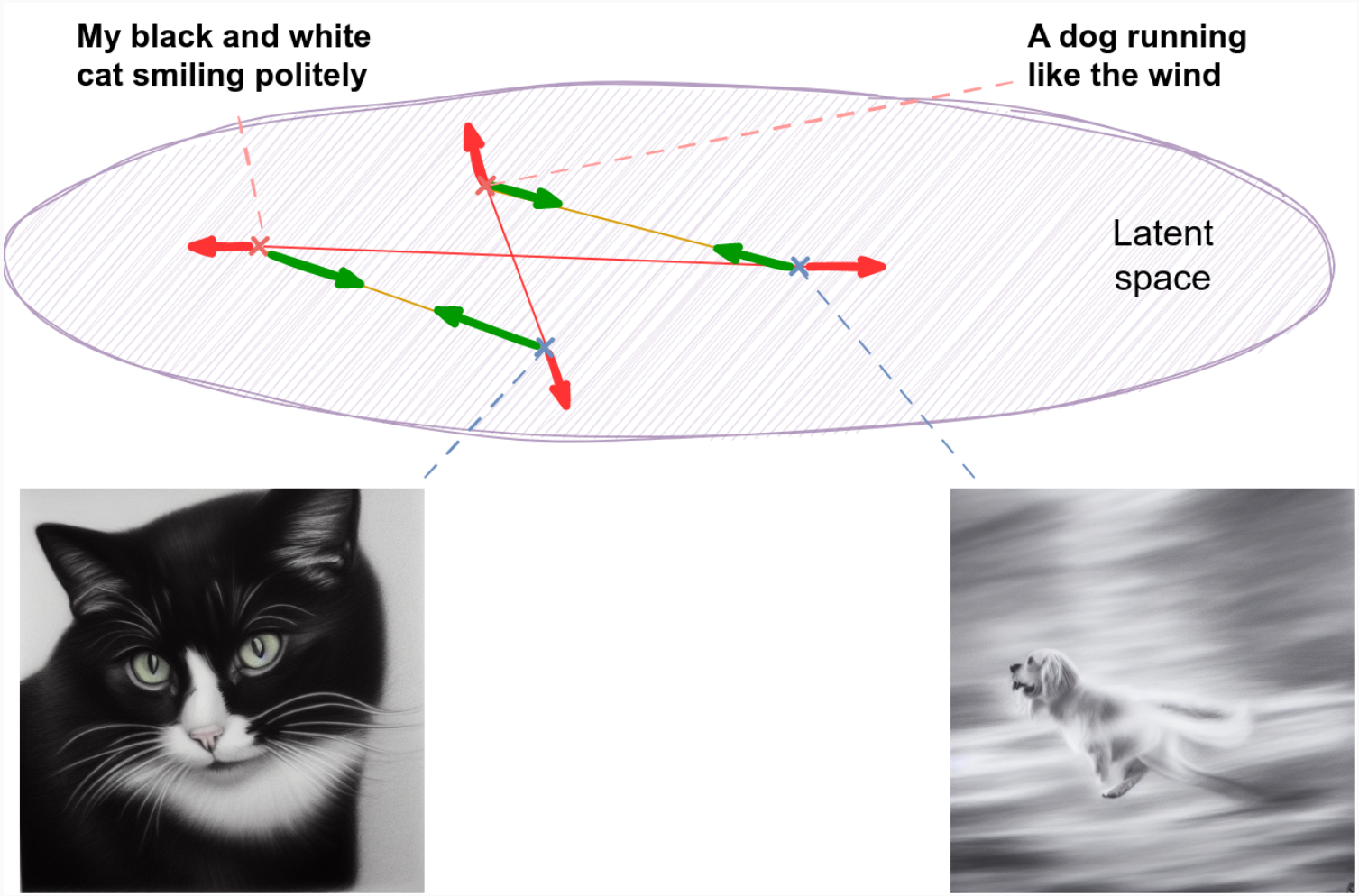
CLIP и BLIP

Мультимодальные латентные пространства

- У нас есть много картинок и текстов, свободно доступных в интернете
- Как использовать текст в паре с изображениями так, чтобы получить хорошее мультимодальное латентное пространство?
- Авторы CLIP пишут, что их первой идеей было обучить свёрточную сеть для изображений и трансформер для текста, чтобы предсказывать подпись к изображению
- Но это не сработало: предсказывать точную подпись очень сложно (практически безнадёжно), и это не совсем то, что нам нужно в этой модели, нам просто нужны хорошие мультимодальные представления

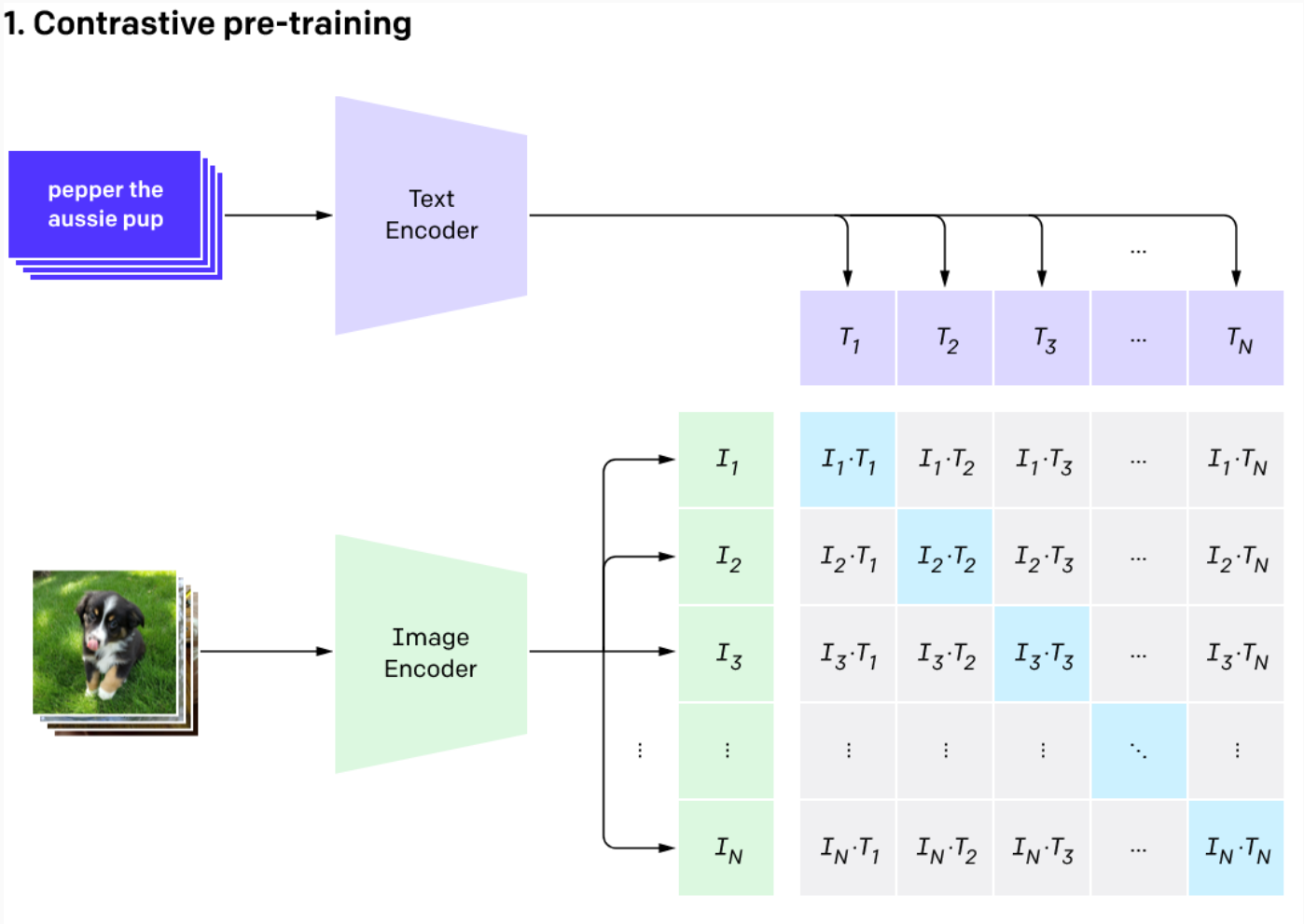
CLIP

- Контрастивное предобучение (contrastive pretraining): моделируем силы притяжения и отталкивания в совместном латентном пространстве



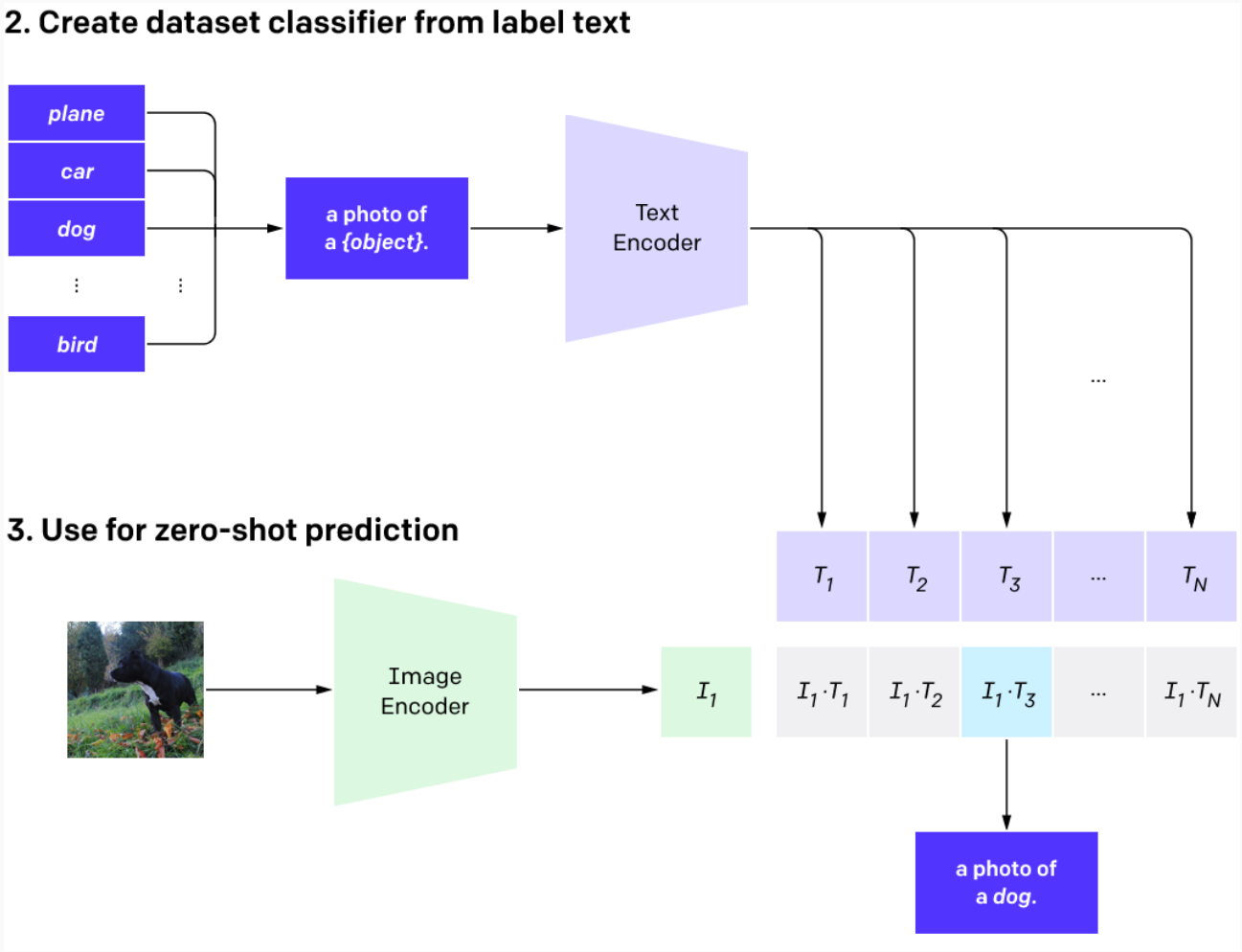
CLIP

- CLIP (Radford et al., 2021; OpenAI), Contrastive Language–Image Pretraining; использовали пары изображение-текст, которые были просто собраны из интернета:



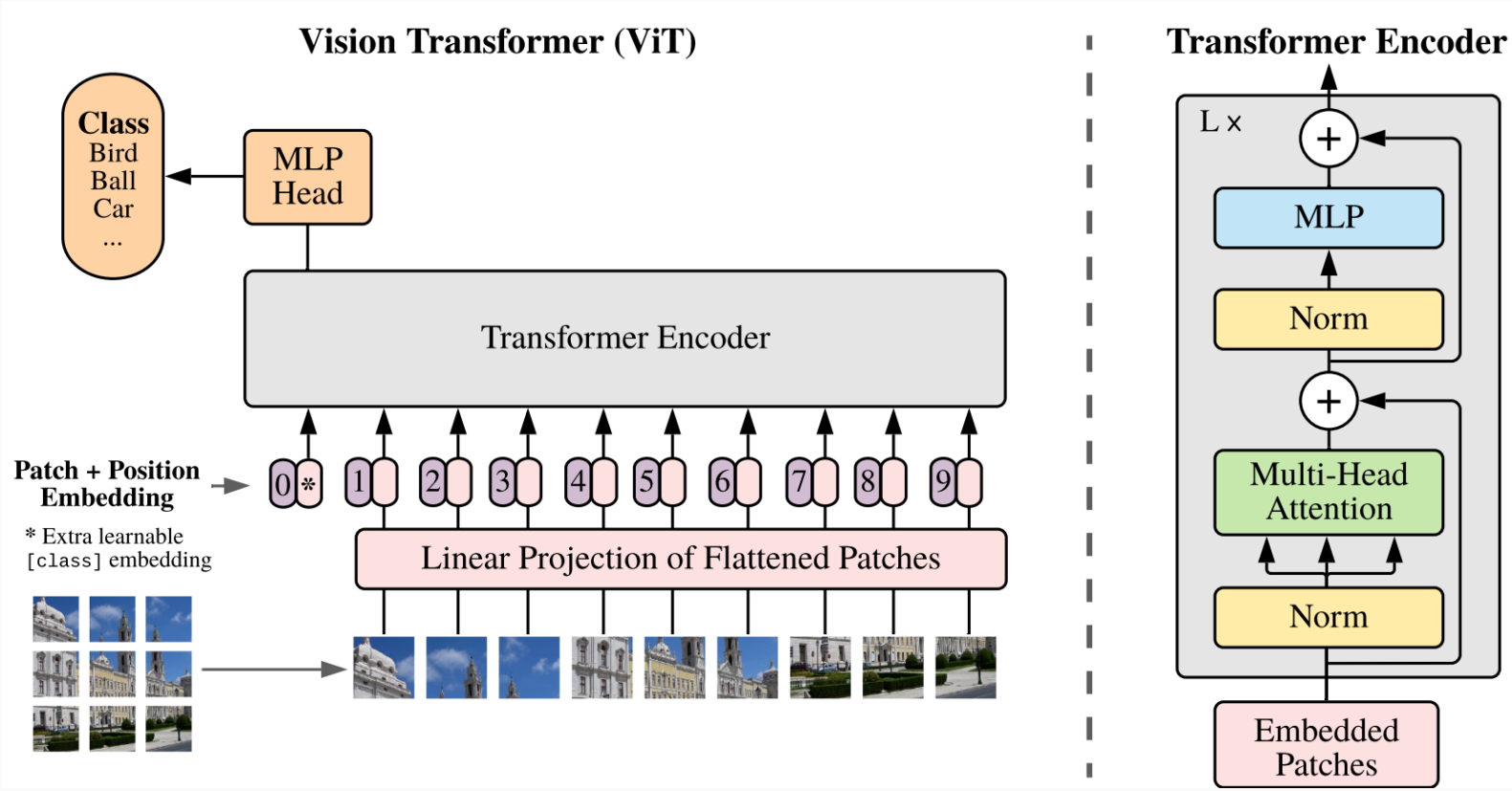
CLIP

- В итоге работала даже zero-shot классификация – преобразуем метки классов в простые запросы:



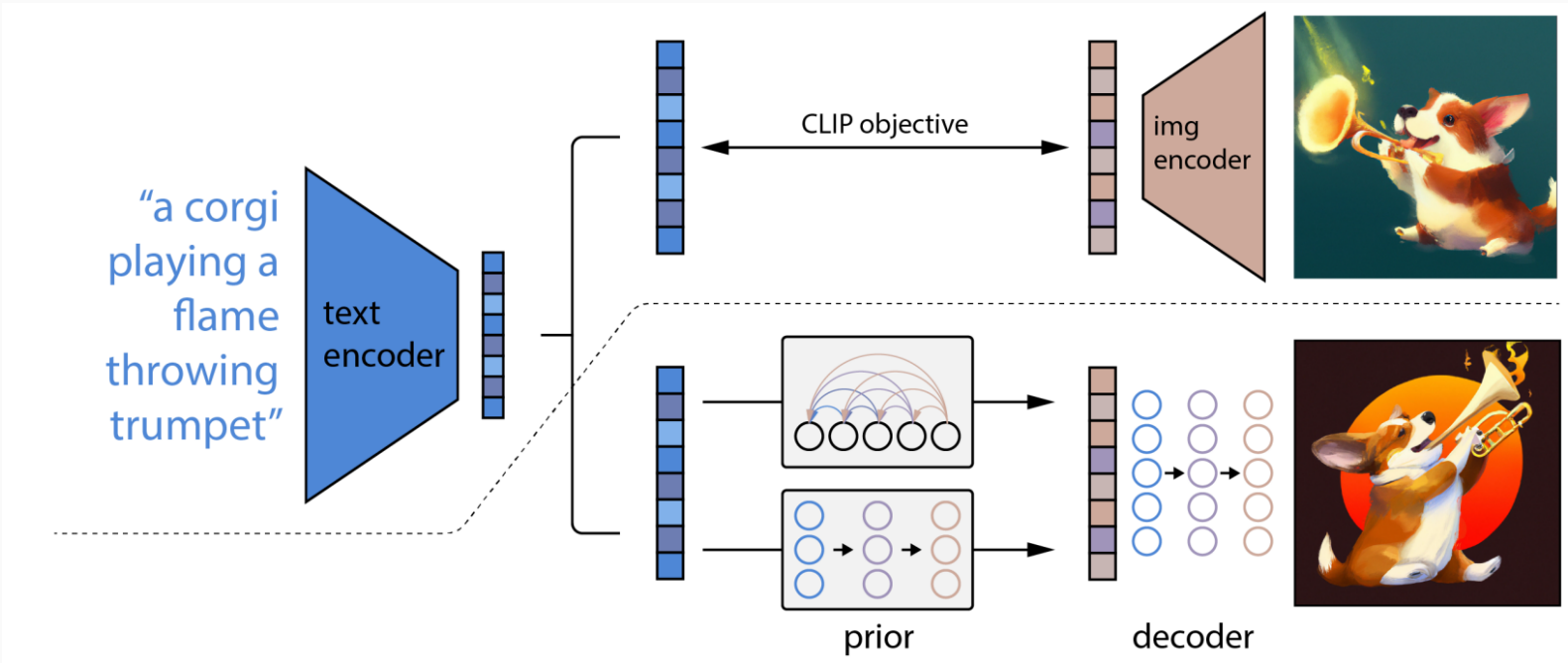
CLIP

- Кодировщик изображений был ViT (Vision Transformer, Dosovitsky et al., 2020)



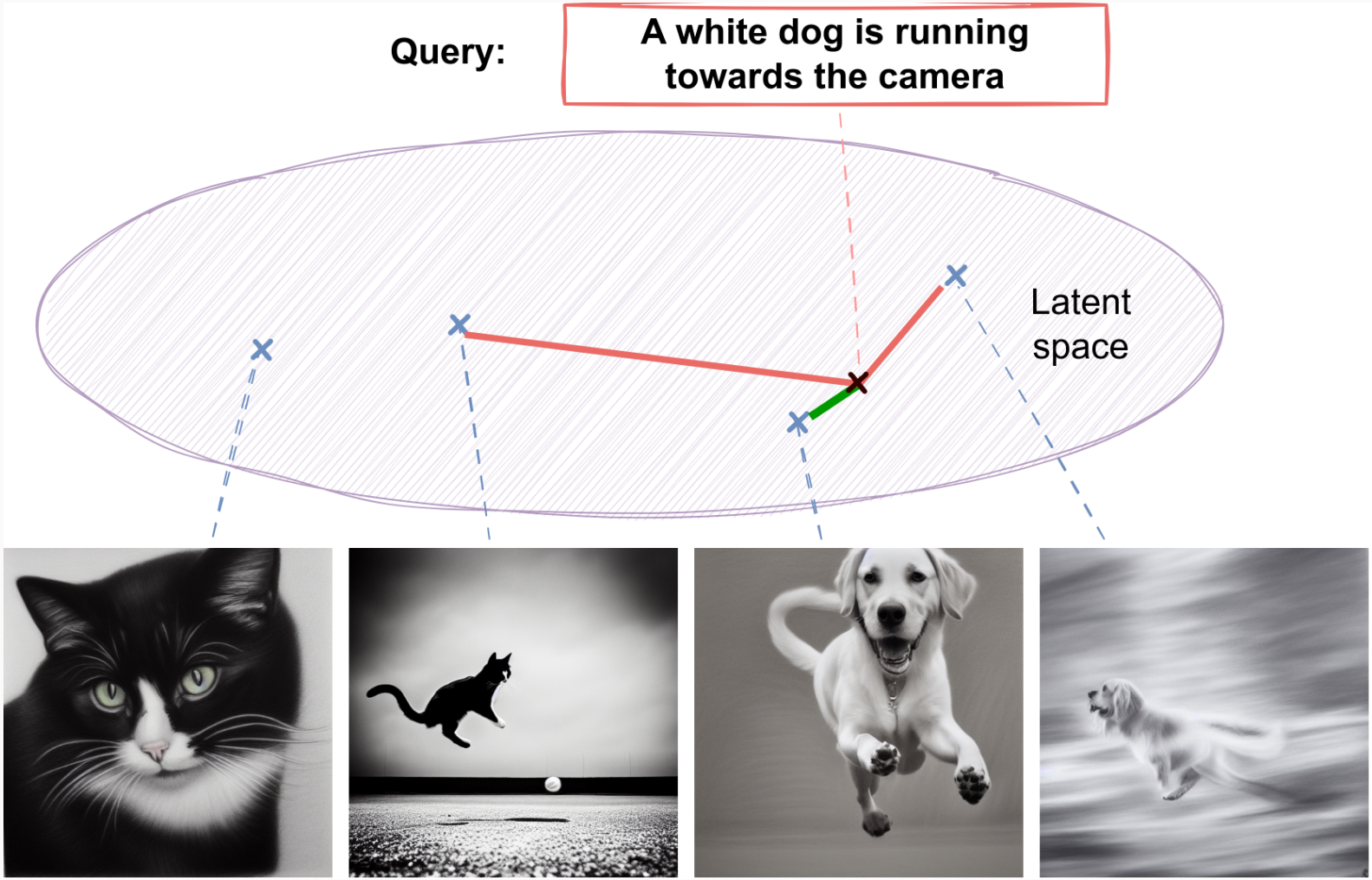
CLIP

- CLIP стал популярным источником совместных представлений; например, DALL-E 2 рисовал картинки в латентном пространстве CLIP:



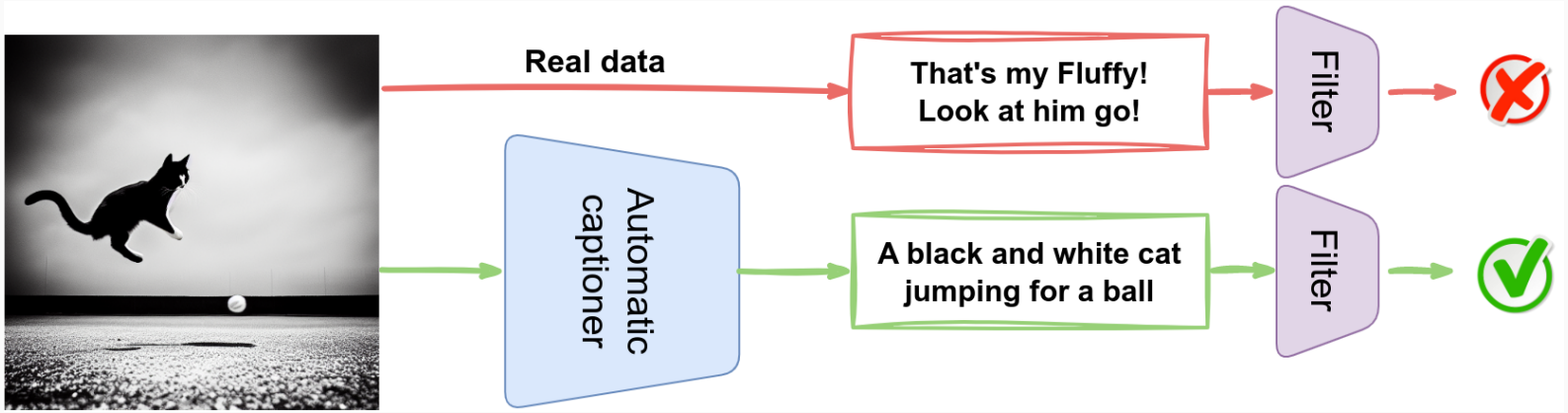
CLIP

- Основное применение CLIP — это обеспечение информационного поиска и порождающих моделей:



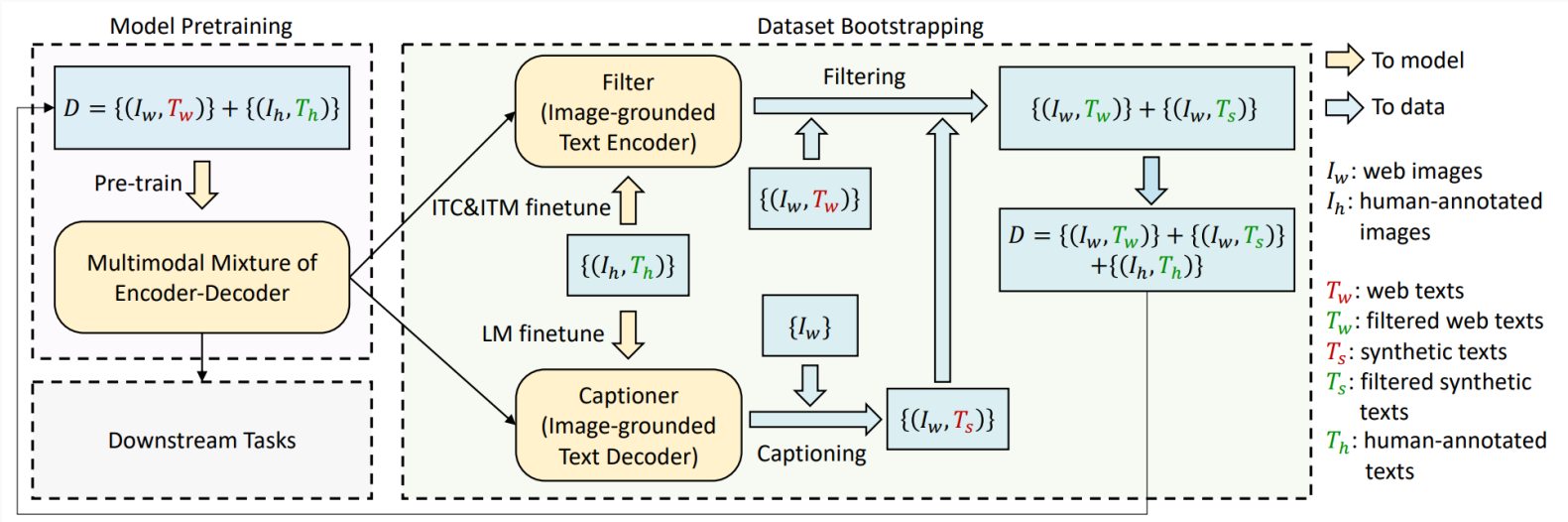
BLIP

- BLIP (Li et al., 2022) – Bootstrapping Language-Image Pretraining; порождаем синтетические подписи и фильтруем их



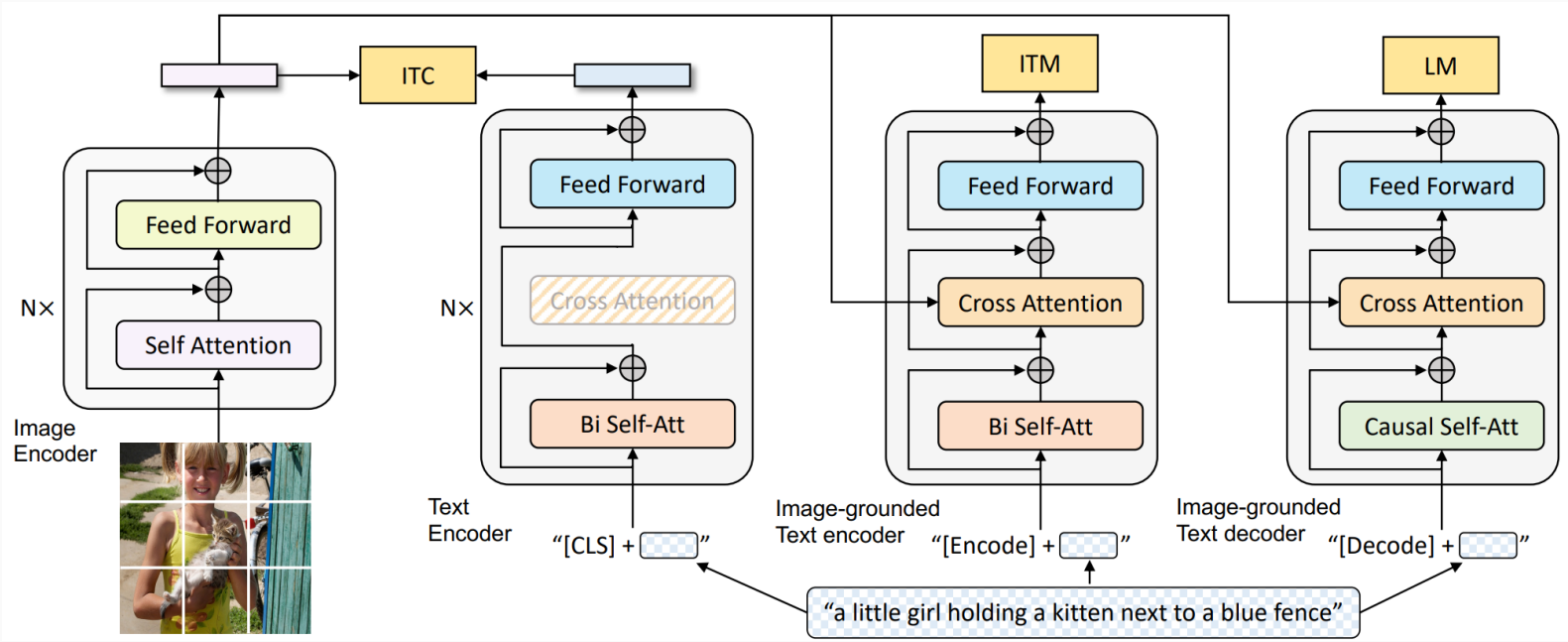
BLIP

- Структура обучения:



BLIP

- Три разных кодировщика и декодер: ITC – контрастивная функция потерь для изображения и текста (сопоставление представлений изображения и текста), ITM – функция потерь на соответствие изображения и текста (различение положительных и отрицательных пар), LM – языковое моделирование (авторегрессивное порождение подписей)



Что изменилось при переходе к зрению?

- Механизм внимания остался тем же; изменились токены, позиционная информация и выходная задача.
- ViT превращает картинку в последовательность патчей; Swin добавляет локальность и иерархию.
- DETR предсказывает множество объектов через запросы и сопоставление с разметкой.
- Главный вопрос при проектировании: какие взаимодействия нужны и сколько будет стоить внимание?

Спасибо за внимание!



