

**Отчёт о выполнении тестового задания
«DINO-Tracker: Taming DINO for Self-Supervised Point
Tracking in a Single Video»**

Выполнил: Дюжев В. Д.

Санкт-Петербург, 2024

ОГЛАВЛЕНИЕ

Введение	1
1 Краткое описание	1
2 Постановка задачи	1
3 Структура отчета	1
Принцип работы	2
1 Идея	2
2 Архитектура	2
2.1 DINOv2-ViT	3
2.2 DELTA-DINO	3
2.3 CNN-Refiner	5
2.4 Денормализация	5
3 Обучение	5
3.1 Обучающие наборы	5
3.2 Функции потерь	6
4 Обработка окклюзий	7
5 Гиперпараметры	8
5.1 Параметры обучения	8
5.2 Параметры работы	8
Запуск предобученной модели	9
1 Базовая визуализация	9
2 Метрики качества	10
3 Производительность	12
4 Мульти-трекинг	13
Запуск на собственных данных	15
1 Описание данных	15
2 Обучение	16
3 Результаты экспериментов	16
3.1 Видео 1	16
3.2 Видео 2	17
3.3 Видео 3	17
3.4 Видео 4	17
3.5 Проверка обобщаемости	18
Техническая информация	20
1 Настройка окружения	20
2 Загрузка датасетов	20
3 Получение метрик для Tapvid-Davis-480	20

4	Обучение	21
5	Инференс	21
	Выводы (In progress)	23

Введение

Краткое описание

Dino-Tracker — метод трекинга, созданный на основе модели DINOv2-ViT, которая используется как базовая для получения качественных обобщений изображений, содержащих важную семантическую информацию. Он позволяет добиться высокой точности, при этом относится к области self-supervised learning, что расширяет возможности его использования (т.к. нет необходимости разметки данных). Перед применением предполагается обучение на конкретном видео.

Постановка задачи

Дана последовательность кадров $\{I^t\}_{t=1}^T$, где T — длина видео. Задача состоит в обучении модели-трекера Π , принимающего на вход целевую точку x_q (query point) и предсказывающего набор $\{\hat{x}^t\}_{t=1}^T$ — траекторию данной точки (положения на всех кадрах). Целевая точка при этом задается положением и кадром (далее — начальный кадр). Таким образом итоговым результатом является возможность предсказания траектории любой точки любого кадра на протяжении всего видео.

Структура отчета

Данный отчет состоит из 5 секций:

1. **Принцип работы** — описание архитектуры нейросетевой модели и алгоритма классификации окклузий, а также процесса обучения.
2. **Запуск предобученной модели** — содержит базовые результаты экспериментов с демонстрацией основных возможностей метода на предложенных в оригинальном репозитории данных.
3. **Запуск на собственных данных** — результаты обучения модели на собранных вручную реальных данных.
4. **Техническая информация** — описание процесса взаимодействия с ПО, дополнения к оригинальному репозиторию.
5. **Выводы** — подведение итогов, выдвижение гипотез для улучшения метода.

Видео-материалы к выполненным экспериментам могут быть найдены по ссылке.

Принцип работы

Идея

Как говорилось ранее, в основе метода лежит предобученная модель DINOv2-ViT (визуальный трансформер) для формирования признаков, содержащих семантическую информацию (далее будем называть эту модель экстрактором признаков). В статье утверждается, что такие признаки, однако, имеют недостаточное содержание временной информации (для связей внутри видео). В качестве решения данной проблемы предлагается обучать дополнительную модель DELTA-DINO, формирующую добавки к выходу экстрактора признаков для формирования их уточнений. Уточненные (refined) признаки могут рассматриваться как обобщения для целой траектории, что в дальнейшем позволит сопоставлять точки на разных кадрах.

Архитектура

В простейшем случае входом сети является целевая точка x_q , ее начальный кадр I^k и кадр I^t для которого необходимо произвести предсказание положения \hat{x}^t . Обозначим выход модели DINOv2 как $\Phi_{DINO}(I)$, а выход DELTA-DINO как $\Phi_{\Delta}(I)$. Тогда в качестве уточненных признаков будем иметь:

$$\Phi(I) = \Phi_{DINO}(I) + \Phi_{\Delta}(I). \quad (1)$$

После получения представления (карт признаков) для обоих кадров ($\Phi(I^k)$ и $\Phi(I^t)$) производится процесс сопоставления. Целевой точке x_q на кадре I^k ставится в соответствие признак φ_q из $\Phi(I^k)$ путем билинейной интерполяции (т.к. размеры карты признаков уменьшены — результат тоекенизации в DINOv2-ViT).

Далее происходит построение карты корреляции S между признаками $\Phi(I^t)$ и φ_q путем расчета косинусного расстояния:

$$S(p) = \text{cos-sim}(\varphi_q, \Phi^t(p)) = \frac{\varphi_q^T \Phi^t(p)}{\|\varphi_q\| \|\Phi^t(p)\|}, \quad (2)$$

где p — обозначение положения на карте признаков.

Для получения итоговой карты распределения вероятности H находящаяся целевой точки карта корреляции подается на вход сверточной сети (CNN-Refiner) с функцией Softmax на выходе.

Предсказание положения формируется как взвешенная сумма координат точек на карте распределения в окрестности радиуса R точки с наи-

большей вероятностью ($x_{p_{\max}}$):

$$\hat{x}^t = \frac{\sum_{p \in \Omega} H(p) \cdot x_p}{\sum_{p \in \Omega} H(p)}, \quad (3)$$

где $\Omega = \{p : \|x_p - x_{p_{\max}}\|_2 \leq R\}$. Итоговым предсказанием является $\Pi(x_q, t) = \hat{x}^t$, и траектория для $x_q - \mathcal{T}_q = \{\hat{x}^t : \hat{x}^t = \Pi(x_q, t), t = 1 \dots T\}$.

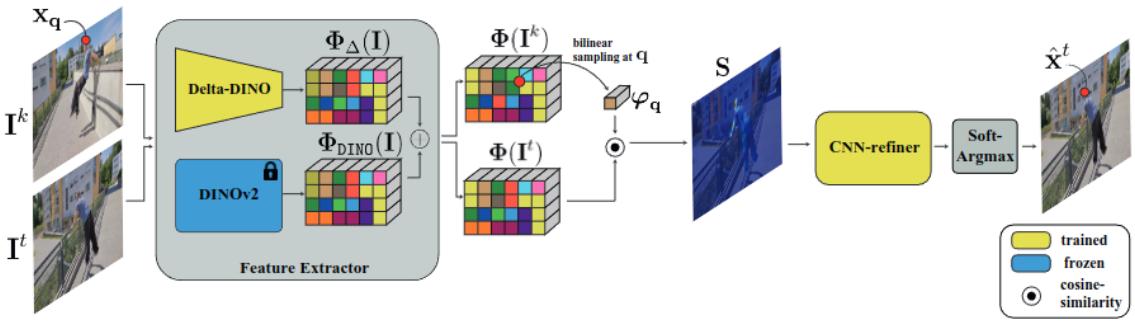


Рис. 1. Схема Dino-Tracker из статьи.

На рисунке 1 изображена схема работы метода от авторов статьи. Ниже приведены подробности реализации описанного алгоритма.

2.1 DINOv2-ViT

DINOv2-ViT — предобученный визуальный трансформер (используется модель ViT/14 с выходом на 16 слое), широко применяемый как базовая модель экстракции признаков в области компьютерного зрения. Являясь частью DINO-Tracker, его веса не участвуют в обучении. Единственным изменением, отличающим используемый экстрактор от оригинального является уменьшенный размер шага токенизации (7 вместо 14) для повышения разрешения карты признаков. Для изображений 480x854 карта признаков имеет размерность 1024x67x121 (1024 — размерность пространства признаков).

2.2 DELTA-DINO

Рассмотрим подробнее устройство модели DELTA-DINO. Эта сверточная сеть состоит только из 4 слоев со следующим преобразованием каналов: $[3 \rightarrow 64 \rightarrow 128 \rightarrow 256 \rightarrow 1024]$. Каждый слой, кроме последнего представляет собой комбинацию Conv2d (kernel:5, stride: 1, reflection padding: 2), BatchNorm2d, ReLU, BlurPool. Последний слой: Conv2d (kernel: 5, stride: 1, reflection padding: 4, dilation: 2), BatchNorm2d. На рисунке 2 представлена визуализация данной модели (размерности рассчитаны с учетом входного изображения 480x854).

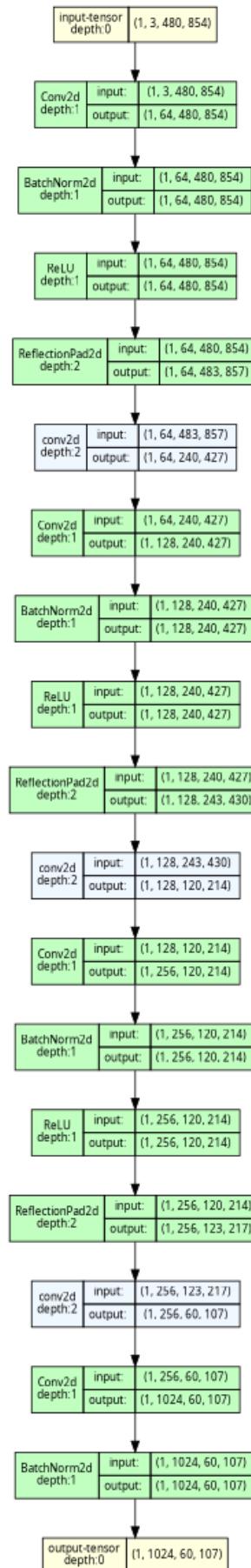


Рис. 2. Архитектура DELTA-DINO (визуализация — torchview).

Важно отметить, что размерность выхода DELTA-DINO (1024x60x107) не совпадает с исходной размерностью карты признаков DINOv2-ViT. Для расчета уточненных признаков необходимо совершить приведение к размерности признаков исходных. Данная задача решается методом линейной интерполяции координатной сеткой.

2.3 CNN-Refiner

Сверточная нейронная сеть на выходе имеет структуру: Conv2d (kernel: 3, padding: 1), ReLU, Conv2d (kernel: 3, padding: 1). Преобразование каналов: $[1 \rightarrow 16 \rightarrow 1]$. Таким образом, размерность карты распределения остается такой же как у карты корреляции.

2.4 Денормализация

Как было сказано в прошлом пункте, выходная карта распределения имеет размерность, совпадающую с картой корреляции и картой признаков. Для работы алгоритма в исходной постановке необходимо вернуть предсказанные положения в размерность входных изображений. Для этого производится линейная денормализация предсказанного положения. Именно для обеспечения большей точности и «гладкости» данной процедуры сопоставления при расчете \hat{x}^t применяется взвешенная сумма.

Обучение

Обучение DINO-Tracker происходит в парадигме self-supervised learning на конкретном видео. Это подразумевает нахождение и использование закономерностей в данных для выполнения задачи (возможно промежуточной и не являющейся целевой). Предобработка данных представляет из себя расчет оптического потока (ОП) в видео, а также нахождение семантически близких точек (best-buddies) в последовательности кадров на основе признаков DINOv2-ViT. Эти данные сформируют подобие обучающей выборки для трекинга точек.

3.1 Обучающие наборы

Набор соответствующих друг другу точек, найденный на основе расчета оптического потока Ω_{flow} состоит из близких по кадрам точек (т.к. надежность оптического потока теряется со временем).

Для сэмплинга набора семантически близких точек $\Omega_{\text{DINO-бв}}$ используется принцип ближайших соседей:

$$NN(\varphi_{\text{DINO}}^i, \Phi_{\text{DINO}}(I^j)) = \varphi_{\text{DINO}}^j \wedge NN(\varphi_{\text{DINO}}^j, \Phi_{\text{DINO}}(I^i)) = \varphi_{\text{DINO}}^i, \quad (4)$$

где $NN(\varphi, \Phi)$ — обозначение ближайшего соседа φ в карте признаков Φ , а I^i, I^j — выбранные кадры. В процессе обучения аналогичным образом формируется набор $\Omega_{\text{rfn-bb}}$ на основе уточненных признаков.

3.2 Функции потерь

Задача оптимизации ставится для агрегированной функции потерь, состоящей из нескольких компонентов:

$$\mathcal{L} = \mathcal{L}_{\text{flow}} + \lambda_1 \mathcal{L}_{\text{dino-bb}} + \lambda_2 \mathcal{L}_{\text{rfn-bb}} + \lambda_3 \mathcal{L}_{\text{rfn-cc}} + \lambda_4 \mathcal{L}_{\text{prior}}, \quad (5)$$

где $\{\lambda_i\}$ — весовые коэффициенты, $\mathcal{L}_{\text{flow}}$ — функция потерь ОП, $\mathcal{L}_{\text{dino-bb}}$ и $\mathcal{L}_{\text{rfn-bb}}$ — функции потерь семантически схожих точек (на основе сырых и уточненных признаков), $\mathcal{L}_{\text{rfn-cc}}$ — функция потерь циклической согласованности, $\mathcal{L}_{\text{prior}}$ — функция потерь априорной информации.

В некоторых последующих формулах применяется функция потерь Хубера:

$$L_H(x, y) = \begin{cases} 0.5\|x - y\|^2, & \|x - y\| < \delta \\ \delta(\|x - y\| - 0.5\delta), & \|x - y\| \geq \delta \end{cases} \quad (6)$$

В реализации, предложенной авторами $\delta = \frac{1}{32}$.

Потери ОП

Функция представляет собой вычисление суммы ошибок для прямого и обратоного потока:

$$\mathcal{L}_{\text{flow}} = \sum_{(x^i, x^j) \in \Omega_{\text{flow}}} L_H(\Pi(x^i, j), x^j) + L_H(\Pi(x^j, i), x^i) \quad (7)$$

Потери семантически схожих точек

Ставится цель увеличения корреляции между уточненными признаками семантически схожих точек и уменьшения корреляции с остальными. Для этого расчитывается «contrastive loss» (широко применяется в задачах self-supervised learning) между парами из $\Omega_{\text{dino-bb}}$:

$$l(\varphi^i, \varphi^j) = -\log \frac{\exp(\text{cos-sim}(\varphi^i, \varphi^j)/\tau)}{\sum_p \exp(\text{cos-sim}(\varphi^i, \Phi^j(p))/\tau)}, \quad (8)$$

где τ — параметр «температуры», регулирующий гладкость.

Итоговая функция потерь имеет вид:

$$\mathcal{L}_{\text{dino-bb}} = \frac{1}{|\Omega_{\text{dino-bb}}|} \sum_{(\varphi^i, \varphi^j) \in \Omega_{\text{dino-bb}}} \frac{1}{2} w_{\text{dino-bb}}^{ij} (l(\varphi^i, \varphi^j) + l(\varphi^j, \varphi^i)), \quad (9)$$

где $\{w_{\text{dino-bb}}^{ij}\}$ — веса, рассчитываемые на основе величины уверенности в сходстве.

$\mathcal{L}_{\text{rfn-bb}}$ определяется аналогичным образом.

Потери циклической согласованности

Важным свойством трекера должна быть циклическая согласованность, а именно: если $x^j = \Pi(x^i, j)$, то $x^i \approx \Pi(x^j, i)$. Функция потерь формулируется с учетом этого:

$$\mathcal{L}_{\text{rfn-cc}} = \sum_{(x^i, x^j) \in \Omega_{\text{rfn-cc}}} \frac{1}{2} w_{\text{rfn-cc}}^{ij} (L_H(\Pi(x^i, j), x^j) + L_H(\Pi(x^j, i), x^i)), \quad (10)$$

где $\{w_{\text{rfn-cc}}^{ij}\}$ — веса, рассчитанные на основе ошибки циклической согласованности.

Потери априорной информации

Признаки полученные напрямую от DINOv2-ViT обладают высокой обобщающей способностью и содержат важную семантическую априорную информацию. Для ее сохранения вводится функция потерь, поощряющая сохранение нормы и ориентации уточненных признаков (для каждого кадра).

$$\mathcal{L}_{\text{prior}} = \frac{1}{H' \cdot W'} \cdot \sum_p \left| 1 - \frac{\|\Phi(I)[p]\|_2}{\|\Phi_{\text{DINO}}(I)[p]\|_2} \right| + |1 - \text{cos-sim}(\Phi(I)[p], \Phi_{\text{DINO}}(I)[p])|, \quad (11)$$

где H', W' — размеры карты признаков.

Обработка оклюзий

Идея предсказания оклюзий на траектории \mathcal{T}_q состоит в формировании набора опорных (anchor) кадров $\{I^{k_i}\}$ и сравнения признаков предсказанных на них точек с остальными признаками точек на траектории. Под опорными кадрами подразумеваются такие, где косинусное расстояние между признаками целевой точки и точек на них предсказанных выше определенного порога:

$$\text{cos-sim}(\varphi_q, \varphi_{k_i}) > \tau_{\text{anch}} \quad (12)$$

Для каждого опорного кадра k рассчитывается медианная невязка предсказания траектории (в сравнении с другими опорными кадрами):

$$e_k = \text{med}_{k_i}(\|\Pi(\hat{x}^k, k_i) - \hat{x}^{k_i}\|) \quad (13)$$

Точка \hat{x}^t считается видимой если:

$$\begin{cases} \text{cos-sim}(\varphi_q, \varphi_t) \geq \tau_{\text{sim}} \\ \text{med}(\|\Pi(x_q, k) - \Pi(\hat{x}_t, k)\|) < \max_k(e_k) \end{cases}, \quad (14)$$

где τ_{sim} — порог косинусного расстояния.

Выбор такого алгоритма подкрепляется предположением согласования траекторий, а именно: предсказания на основе видимой точки как целевой будут достаточно близки к исходным (т.к. их признаки близки).

Гиперпараметры

В ходе выполнения данного тестового задания не проводилось исследование влияния гиперпараметров на качество обучения ввиду ограниченного времени и высоких вычислительных требований метода. Ниже приведены основные из них, рекомендуемые авторами с предположениями относительно интуиции их выбора.

5.1 Параметры обучения

В ходе обучения используется оптимизатор Adam как наиболее распространенный вариант. Благодаря адаптивному шагу градиентного спуска достигается более быстрая и стабильная сходимость. Базовый шаг обучения выбран 0.01. Для части CNN-Refiner задано расписание уменьшения шага каждые 40 эпох: коэффициент затухания — 0.999. Это позволяет избежать переобучения модели и добиться большей стабильности.

Весовые коэффициенты в формуле функции потерь 5 эмперически выбраны следующими: $\lambda_1 = 25 \cdot 10^{-5}$, $\lambda_2 = 5 \cdot 10^{-5}$, $\lambda_3 = 0.5$, $\lambda_4 = 10^{-4}$.

Количество эпох выбирается в зависимости от длины видео. Для последовательностей около 100 кадров рекомендуется производить 10000 эпох, для более 250 кадров — 20000.

Параметр температуры в уравнении 8 установлен $\tau = 0.1$, что позволяет более явно выделить влияние пар точек с высокой корреляцией.

Функции потерь $\mathcal{L}_{\text{rfn-bb}}$ и $\mathcal{L}_{\text{rfn-cc}}$ применяются только после 5000 эпох, т.к. они расчитаны на уже частично обученные работающие уточненные признаки.

5.2 Параметры работы

Радиус окрестности наиболее вероятной точки на карте распределения из формулы 3 выбран $R = 35\text{px}$, что соответствует 5 токенам на карте признаков DINOv2 (с учетом шага токенизации 7).

Пороги используемые при выборе опорных кадров и предсказании оклюзий (12 и 14) установлены $\tau_{\text{anch}} = 0.7$ и $\tau_{\text{sim}} = 0.6$. Важно чтобы оба числа были достаточно велики (отражать сходство видимых точек с целевой), при этом $\tau_{\text{anch}} > \tau_{\text{sim}}$ для формирования набора опорных кадров как обладающих высшей уверенностью предсказания.

Запуск предобученной модели

В данной секции представлены результаты использования модели DINO-Tracker на последовательностях из датасета Tapvid-Davis-480. Данные доступны по ссылке, указанной в репозитории.

Базовая визуализация

В качестве примеров для демонстрации работоспособности базовой модели DINO-Tracker (предсказаний нейросетевой части, без предсказания окклюзий) были выбраны видео-последовательности с достаточно простым профилем движения, без явных окклюзий. На рисунках 3-5 изображены по 3 кадра (первый, центральный и последний) с демонстрацией промежуточных траекторий для целевых точек. Начальным кадром во всех случаях является первый.



Рис. 3. Траектории DINO-Tracker. Tapvid-Davis-480/29.



Рис. 4. Траектории DINO-Tracker. Tapvid-Davis-480/26.



Рис. 5. Траектории DINO-Tracker. Tapvid-Davis-480/21.

Можем заметить, что полученные траектории имеют естественный вид. Базовый метод может быть успешно применен и к трекингу в более сложных окружениях (рисунок 6), однако важно чтобы целевая точка находилась в достаточно уникальной части изображения и не подвергалась окклюзиям. Например, на рисунке 7 видно, что целевая точка сопоставляется неправильно с определенного момента времени (из-за появления схожего объекта и окклюзии первоначальной цели). Данные ограничения помогают во многом снять предсказание окклюзий, позволяющее определить валидность оценки.



Рис. 6. Траектории DINO-Tracker. Tapvid-Davis-480/13.

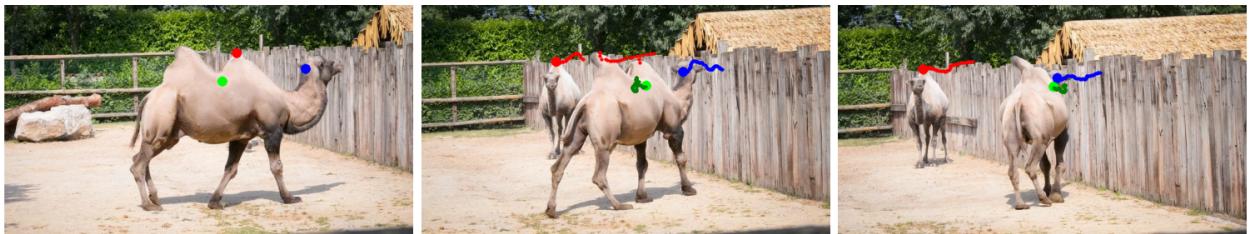


Рис. 7. Траектории DINO-Tracker. Tapvid-Davis-480/15.

Все эксперименты данной секции доступны в виде визуализаций на диске в двух вариантах — видео с трекингом точек, соответствующее рисункам (без проверки окклюзий) и видео трекингом множества семплированных точек (предложенные в датасете Tapvid-Davis-480 в качестве обучающих данных) с учетом окклюзий.

Метрики качества

Оценка качества трекинга производится на датасете Tapvid-Davis-480 согласно инструкции, указанной в репозитории (также есть возможность провести оценку на других предложенных наборах данных, однако ввиду длительности процесса был выбран один из них).

Для оценки используются метрики, предложенные в оригинальной статье «TAP-Vid: A Benchmark for Tracking Any Point in a Video»:

1. occlusion accuracy (OA) — отношение количества правильно предсказанных окклюзий в кадров к длине последовательности (стандартная accuracy классификации);
2. position accuracy ($< \delta^x$) — отношение количества правильно предсказанных положений видимых точек (без окклюзий) последовательности к количеству видимых точек, где правильными считаются предсказания, лежащие в заданной окрестности (в пикселях) действительных значений;
3. jaccard — объединенная метрика для оценки качества предсказаний окклюзий и положений, рассчитываемая как отношение количества правильно предсказанных положений видимых точек с соответствующим правильным предсказанием окклюзий последовательности к количеству видимых точек + количеству неправильно классифицированных невидимых точек или точек с неправильными предсказаниями положений, предсказанных видимыми.

Важно отметить, что метрики $< \delta^x$ и jaccard рассчитываются для всех кадров (кроме начальных для соответствующих ключевых точек) с учетом приведения к размеру 256x256. В качестве итоговых метрик выступают OA , average jaccard (AJ) и average position accuracy ($< \delta_{avg}^x$), где последние две рассчитываются как средние значения jaccard и $< \delta^x$ для окрестностей в 1, 2, 4, 8 и 16 пикселей.

id	OA	AJ	$< \delta_{avg}^x$	id	OA	AJ	$< \delta_{avg}^x$
25	0.86407	0.52461	0.71564	23	0.88914	0.70477	0.8492
7	0.84293	0.62454	0.82839	14	0.95139	0.64647	0.75761
20	0.83599	0.5811	0.77156	13	0.89146	0.59985	0.72462
1	0.83536	0.55336	0.77867	29	0.96237	0.78638	0.87435
26	0.92519	0.72271	0.84825	17	0.89871	0.61162	0.75912
5	0.87887	0.60303	0.8446	4	0.87391	0.66864	0.88173
2	0.69653	0.37687	0.72344	11	0.79149	0.57035	0.7935
19	0.82501	0.47515	0.70126	16	1.0	0.94087	0.96645
6	0.85056	0.63005	0.80656	10	0.94926	0.76182	0.86853
22	0.79057	0.54181	0.76219	3	0.91635	0.75579	0.8585
8	0.88787	0.69386	0.84002	21	0.98371	0.84777	0.90917
28	0.93551	0.73576	0.84587	24	0.94805	0.8512	0.9379
0	0.99963	0.64083	0.72042	15	0.96577	0.80356	0.87399
9	0.69041	0.29614	0.5251	18	0.81178	0.52368	0.71774
27	0.87957	0.62347	0.7801	12	0.99365	0.88775	0.93579
<i>avg</i>	<i>0.8855</i>	<i>0.65279</i>	<i>0.80668</i>	—	—	—	—

Таблица 1. Метрики качества. Датасет Tapvid-Davis-480.

В статье завлены следующие данные для рассматриваемого датасета: $OA = 0.881$, $AJ = 0.646$, $\delta_{avg}^x = 0.804$. Эксперимент подтверждает достижение таких значений.

Расширенный файл, содержащий метрики, доступен на диске.

Производительность

Эксперименты производятся на устройстве Lenovo Legion 7 (AMD Ryzen 9 5900HX, NVIDIA GeForce RTX 3080 120W 16Gb, 32 Gb RAM).

Для получения данных о производительности в инференсе метод был применен ко всем последовательностям из датасета. Эксперимент проводился только для нейросетевой части DINO-Tracker. Дальнейшее предсказание окклузий сильно зависит от контекста изображения, а точнее — от количества выбранных опорных кадров для обработки. На вход модели подавалась одна ключевая точка, соответствующая центру первого кадра. Принятый размер батча: 1. Ниже приведен график, иллюстрирующий зависимость времени обработки видео от количества кадров в нем.

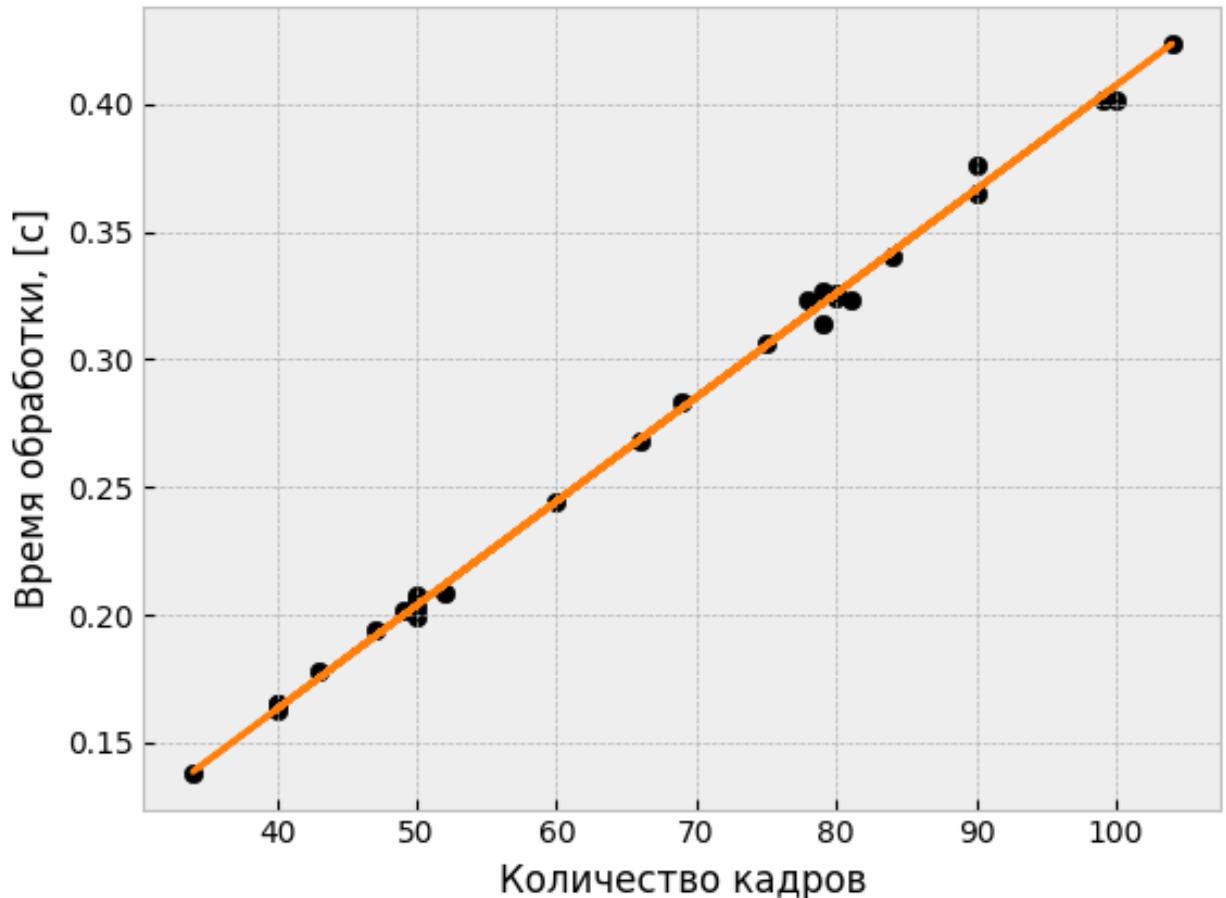


Рис. 8. Зависимость времени инференса нейросетевой части DINO-Tracker от количества кадров.

Алгоритм обработки видео последовательно предсказывает положение целевой точки на кадрах. Данные подтверждают, что зависимость является линейной. Аппроксимировав ее моделью линейной регрессии без смещения получим, что на предсказание позиции целевой точки в одном кадре тратится в среднем 0.0041 [с]. В процессе инференса в среднем затрачивалось 14.5 Гб видеопамяти.

Стоит уточнить, что процесс инференса производится с учетом заранее расчитанных карт признаков DINOv2-ViT и по сути представляет из себя работу DELTA-DINO и CNN-Refiner. Более подробное техническое описание представлено в секции «Техническая информация».

Мульти-трекинг

Задача множественного трекинга объектов подразумевает наличие механизма сопоставления объектов с траекториями движения (ассоциация данных). Базовый метод DINO-Tracker не использует данную процедуру. Как видно на примерах, представленных выше (рисунки 6, 7) в такой конфигурации он может быть применен при отсутствии явных окклузий и достаточном количестве семантически уникальных точек на объектах.

При использовании предсказания окклузий область применения метода для задач множественного трекинга можно существенно расширить. В алгоритме обработки окклузий частично применяется подход ассоциации — проверяется схожесть траекторий на разных участках видео согласно установленному критерию. На рисунке 9 показаны положения целевых точек (1 и 30 кадр), изначально расположенных на объектах.



Рис. 9. Трекинг точек на нескольких объектах. Tapvid-Davis-480/8.

Можно предположить, что произведя изначальную детекцию объектов и сформировав наборы целевых точек, соответствующих им, производить трекинг объектов путем расчета среднего положения всех их видимых точек. Однако такой подход требует усовершенствования для обработки новых объектов появляющихся на видео (что в итоге возвращает к задаче верхнеуровневой ассоциации данных). Кроме того, даже с учетом обработ-

ки окклузий семантически близкие точки могут неверно сопоставляться, ухудшая качество оценки положения.

Запуск на собственных данных

В данной секции приведены результаты экспериментов на данных, снятых на телефон. Видео были приведены к формату, используемому в датасете Tapvid-Davis-480 (изображения 480x854, 10fps, длина до 200 кадров) и разбиты на кадры (архивы могут быть найдены на диске).

Описание данных

Всего записано 4 видео, демонстрирующих различные ситуации, в которых может быть применен трекинг:

1. взаимодействие объектов с простым профилем движения;
2. взаимодействие объектов со сложным профилем движения;
3. движение объекта с изменением плана;
4. движение объекта с сохранением плана.

На каждом видео из собранных присутствуют незначительные окклюзии. На видео 2 и 4 присутствуют семантических схожие объекты.

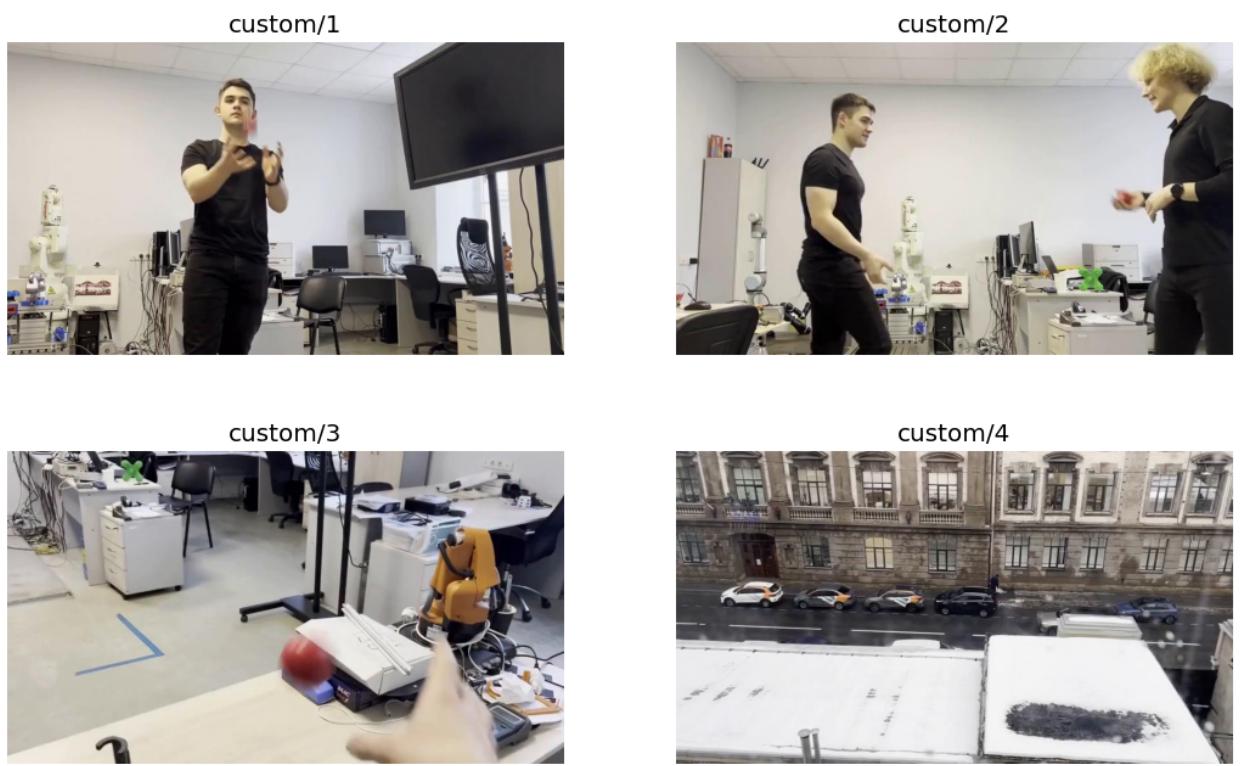


Рис. 10. Примеры кадров из собранных видео.

Обучение

Первым этапом является предобработка изображений — вычисление карт признаков DINOv2-ViT, расчет оптического потока и нахождение семантически схожих точек. Данный процесс занимает в среднем 30-40 минут. Сам процесс обучения длится около полутора часов (10000 эпох). Затрачиваемое количество видеопамяти достигало 21 Гб. В связи с этим оно проводилось на отличном от выше указанного оборудования: Intel 11th Gen Core i7-11700, NVIDIA GeForce RTX 3090 450W 24Gb, 32Gb RAM.

Результаты экспериментов

Все эксперименты проводились путем сэмплирования 100 точек в зоне интереса и предсказания траекторий для них, а также предсказанием окклюзий (на изображениях показаны 3 случайных точки объекта, в видео-материалах отображены все). Инференс производился на том же оборудовании, что и обучение. Размер батча был выбран 50 как оптимальный по времени работы и удовлетворяющий требованиям памяти в большинстве случаев для видео из датасета Tapvid-Davis-480. Видео-визуализации с результатами испытаний доступны на диске.

3.1 Видео 1

- количество кадров: 119;
- время обработки: 2 мин 20 с.

На рисунке 11 продемонстрированы траектории на части видео (для удобства визуализации). Сэмплинг целевых точек происходил в зоне взаимодействия руки и мяча. Однако в итоге видимыми (без окклюзий) были классифицированы только целевые точки, находящиеся на руке. Их трекинг оказался стабильным и точным. Среди возможных причин неудачи трекинга мяча стоит выделить малый размер объекта и высокую скорость движения — из-за этого изображение теряет исходный вид и семантическую информацию.

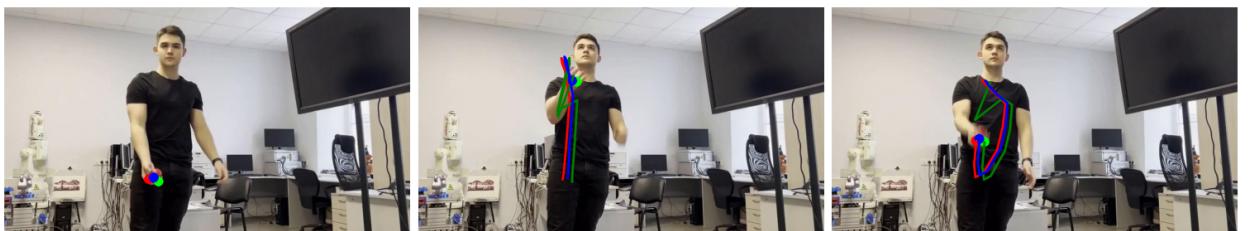


Рис. 11. Траектории DINO-Tracker. Custom/1.

3.2 Видео 2

- количество кадров: 157;
- время обработки: —.

Обучение для видео 2 не удалось запустить (требовалось более 24 Гб видеопамяти).

3.3 Видео 3

- количество кадров: 69;
- время обработки: 22 с.

На рисунке 12 изображены предсказанные положения мяча. Как и в видео 1 трекинг мяча становится несостоятельным как только он начинает быстрое движение или изменение размера (из-за отдаления) и теряет четкость. В видео-визуализации трекинга можно заметить, что предсказания остаются стабильными первые несколько кадров до момента броска.



Рис. 12. Траектории DINO-Tracker. Custom/3.

На основе данных экспериментов 1 и 3 можно выдвинуть предположение, о чувствительности метода к искажениям изображения при быстром движении объекта малых размеров. Также нежелательно наличие значимых окклузий в данном случае. Пример стабильного и точного трекинга руки (также являющейся небольшим объектом) в эксперименте 1 демонстрирует возможность работы метода в отсутствии указанных выше эффектов.

3.4 Видео 4

- количество кадров: 50;
- время обработки: 57 с.

На рисунке 13 представлен трекинг движущегося автомобиля. Это пример простого профиля движения с окклузиями. Можем наблюдать стабильный и качественный трекинг объекта. На видео-визуализации заметно,

что траектории нескольких точек определяются некорректно после контакта объекта с семантически близким ему (другим автомобилем).



Рис. 13. Траектории DINO-Tracker. Custom/4.

3.5 Проверка обобщаемости

В статье авторы не приводят данных об обобщающей способности обученной модели (качество работы для семантически схожих окружений и объектов). В этом эксперименте предлагается проверить работоспособность моделей, обученных на видео из датасета Tapvid-Davis-480 при обработке представленного выше видео 4.

Видео Tapvid-Davis-480/1 обладает схожими чертами с видео 4 (также демонстрирует движение автомобиля — рисунок 14). В предположении об инвариантности DINOV2 признаков к размеру была выдвинута гипотеза о применимости модели к видео 4.



Рис. 14. Пример кадра. Tapvid-Davis-480/1.

Обработка видео заняла 10 с. Результат трекинга изображен на рисунке 15. Качество трекинга заметно ухудшилось.



Рис. 15. Траектории DINO-Tracker (модель обученная на семантически схожем видео). Custom/4.

Для сравнения эксперимент был также проведен на видео Tapvid-Davis-480/15, обладающим совершенно отличным контекстом. Время обработки составило 11 с. Результат представлен на рисунке 16. Качество детекции сопоставимо с предыдущим результатом — предположительно априорные признаки DINOv2 довольно хорошо сохраняют свою структуру после уточнения моделью DELTA-DINO, привносящей в основном информацию о временных связях в конкретном видео для которого было проведено обучение.



Рис. 16. Траектории DINO-Tracker (модель обученная на семантически отличном видео). Custom/4.

Заметим, что время обработки заметно снижено по сравнению с использованием тренированной на видео 4 модели. Это связано с уменьшенным количеством опорных кадров при обработке окклузий (из-за сниженной корреляции между признаками разных кадров их отбирается меньше).

На основе данных экспериментов можно предположить, что для качественного обобщения требуется более близкая семантика изображений или обученная DELTA-DINO в целом обладает плохой обобщающей способностью.

Техническая информация

В данной секции рассматривается процесс установки необходимого ПО и его использования для проведения экспериментов, описанных в предыдущих секциях. В ходе работы над тестовым заданием был создан fork оригинального репозитория, содержащий некоторые утилитарные скрипты и интерактивные блокноты.

Для начала работы с репозиторием:

```
git clone https://github.com/diuzhevVlad/dino-tracker-assignment.git
```

Настройка окружения

В оригинальном репозитории авторы предлагают использовать окружение conda на основе python3.9. При использовании официальной инструкции возникают проблемы совместимости версий библиотек. Необходимо указать версию *pytputr 1.26.0* в файле *requirements.txt*. Данное исправление сделано в указаном выше репозитории.

Для создания окружения:

```
conda create -n dino-tracker python=3.9  
conda activate dino-tracker  
pip install -r requirements.txt
```

Для удобного добавления репозитория в *PYTHONPATH*:

```
conda install conda-build  
conda develop .
```

Загрузка датасетов

Для использования датасетов их файлы архивов достаточно распаковать в директорию *dataset*. В ходе выполнения задания использовались custom и Tapvid-Davis-480. Первый содержит только сырье кадры 4 четырех снятых видео. Второй включает в себя помимо этого сегментационные маски для построения репрезентативных визуализаций, а также предобученные модели DELTA-DINO и CNN-Refiner (tracker head).

Получение метрик для Tapvid-Davis-480

Для упрощения процедуры оценки качества на рекомендуемом авторами датасете были написаны два скрипта-оркестратора, оперирующие другими предоставляемыми авторами скриптами.

Запустить расчет DINoV2 признаков для всех видео датасета:

```
python3 benchmarking/preprocess_benchmark.py
```

В ходе работы будет выполнен скрипт *preprocessing/save_dino_embed_video.py* для всех видео датасета. В результате во всех директориях *dataset/tapvid-davis/{i}* должна появиться папка *dino_embeddings*, содержащая файлы признаков.

Запустить расчет траекторий ключевых точек, для которых предусмотрены ground-truth данные:

```
python3 benchmarking/inf_benchmark.py
```

В ходе работы будет выполнен скрипт *inference_benchmark.py* для всех видео датасета. В результате во всех директориях *dataset/tapvid-davis/{i}* должны появиться папки *occlusions* и *trajectories*, содержащие файлы предсказаний траекторий и оклюзий для целевых точек.

Для расчета метрик и сохранения в файл:

```
python3 ./eval/eval_benchmark.py \
    --dataset-root-dir ./dataset/tapvid-davis \
    --benchmark-pickle-path ./tapvid/tapvid_davis_data_strided.pkl \
    --out-file ./tapvid/comp_metrics_davis.csv \
    --dataset-type tapvid
```

В зависимости от мощности используемого оборудования весь процесс может занять до 4 часов.

Обучение

Процесс обучения подробно описан в оригинальной инструкции и состоит из запуска скриптов предобработки (*preprocessing/main_preprocessing.py*) для расчета DINOv2 признаков, а также оптического потока и нахождения семантически схожих точек (best-buddies), и непосредственно скрипта обучения *train.py* с соответствующими параметрами.

Оригинальные скрипты не предполагают процесса логирования (в том числе в *wandb*). Единственным индикатором выполнения является *progressbar* с текущим статусом.

Инференс

Все эксперименты, связанные с инференсом моделей, а также утилитарные функции, упрощающие взаимодействие с базовыми классами, представленными в репозитории, произведены в интерактивном блокноте *visualization/playground.ipynb*. Использование данного файла предполагает проведение этапа инференса бенчмарка Tapvid-Davis и обучение моделей на датасете Custom (суммарно файлы весов и предсказанных траекторий-оклюзий для всех видео занимают слишком большое количество памяти для загрузки на диск, однако могут быть предоставлены по просьбе).

Алгоритм проведения визуализации множества точек в виде сетки подробно описан в официальной инструкции, однако стоит отметить, что такой вид визуализации подходит при наличии ground-truth сегментационных масок (иначе данные плохо интерпретируемы).

Пример визуализации нейросетевой модели посредством библиотеки *torchview* расположен в файле *visualization/model_visualization.ipynb*. Из-за сложной структуры исходных классов моделей их не удалось конвертировать в формат *onnx* для более качественной визуализации.

Выводы (In progress)