Vehicle ReID: сборка, запуск и воспроизведение результатов

Документация решения по разделам 7–9 и 12 ТЗ. С **20 сентября 2026 года** сдаётся d1_j48 (две модели + whitening, §2). Проверки API с Qdrant 15–16 сентября относятся к прежней конфигурации. Аудит актуального Git-клона 21 сентября подтвердил офлайн-сборку на Podman 5.8.1, веса и 60 тестов, но остановился перед пересчётом метрик из-за отсутствующих изображений; полный Compose-стенд тогда не проверялся. Условия нового пересчёта и его границы — §7–8. Ниже приведены команды **Docker** для жюри; отдельные проверки Docker Engine 29.7.2 / Compose 2.40.3 выполнены 22–23 сентября (отчёт).

В Git входят веса, runtime-зависимости в колёсах и фактические исходники обучения. Для пересчёта основных метрик нужны исходные изображения организатора; базовые образы для офлайн-сборки поставляются в репозитории. Полное воспроизведение всех исследовательских отчётов из одного git-репозитория пока не обеспечено. Недостающие материалы и непроверенные числа перечислены в разделах 8–11.

1. Назначение и архитектура

Сервис принимает изображение автомобиля и предоставленный bbox `(x, y, w, h)`, извлекает признак, ищет похожие объекты в галерее и возвращает кандидатов либо пустой ответ. Сдаваемый признак — двумодельный ансамбль **d1_j48**: публичный OSNet-AIN (OMZ 2022.1) плюс наш дообученный OSNet-AIN **combined_v1** (LP-FT на RoundaboutHD, MIT + CARLA, Apache-2.0 + train организатора; журнал обучения — training/combined/), с общим whitening. OCR, распознавание номера и детекция автомобиля в вычислительный путь не входят; влияние области пластины исследовалось отдельно для обеих моделей (training/combined/REPORT.md §5, plate-ablation/).

Компонент	Код	Ответственность
Подготовка изображения	preprocess.py	Чтение JPEG/PNG, bbox, RGB, изменение размера
Модель	model.py	Проверка SHA-256 трёх файлов весов (две ONNX-модели + матрица whitening), ONNX Runtime CPU, двумодельный признак со whitening
Пакетная обработка	batch.py	Последовательное чтение CSV, извлечение векторов, ранжирование, три сдаваемых файла
Переранжирование	rerank.py	Совместная обработка всех запросов и галереи по k-взаимным соседям
HTTP API	main.py	Получение файла/bbox или вектора, валидация, поиск, OpenAPI
Галерея	store.py, Qdrant	Векторы и метаданные, точный поиск по косинусу
Загрузчик галереи	load_gallery.py	Извлечение признаков галереи и пересоздание коллекции Qdrant
Клиент	index.html	Страница браузера, выдаваемая API; отдельного frontend-контейнера нет

Компонент	Код	Ответственность
Измерения	reid_metrics.py, scope_metrics.py	Ранговые метрики, отказ, явные варианты протокола

API и Qdrant запускаются отдельными контейнерами. Пакетный режим использует образ API и работает с файлами, без БД. Это позволяет проверить сдачу без предварительной загрузки галереи в сервер. Общий модуль извлечения признаков сохраняет одинаковый препроцессинг в обоих режимах.

2. Методы и рабочая конфигурация

Признак

Путь обработки: bbox в пикселях исходного кадра → RGB → PIL bilinear, `208×208` → `float32`, диапазон `0…255`, NCHW → **две модели на одном препроцессированном батче**: OSNet-AIN (OMZ 2022.1) и combined_v1 (наш дообученный OSNet-AIN) → L2-нормировка каждого сырого вектора `512` → покомпонентное среднее → L2-нормировка среднего → whitening `y = (x − m) @ P.T` в `float32` (матрица `P` и вектор `m` обучены на `train_fit` по дельтам ансамбля, ρ=0,5; обучение и проверка — training/combined/) → L2-нормировка с накоплением в `float64`, сохранение в `float32`. Порядок важен: среднее нормируется до whitening, потому что `m` обучена на нормированных входах. Внешняя ImageNet-нормировка не применяется: обе модели уже содержат нормировку входа. Инференс выполняется через `CPUExecutionProvider`.

Происхождение **combined_v1**: LP-FT дообучение OSNet-AIN (обучение только классификатора при замороженном backbone → частичная разморозка; полная разморозка предусмотрена воротами, но в этом прогоне пропущена) на объединённом наборе RoundaboutHD (лицензия MIT) + CARLA (лицензия Apache-2.0) + train организатора. Полный журнал: training/combined/journal.md, отчёт с метриками и бутстрэпом — training/combined/REPORT.md. Веса `model/osnet_ain_combined_v1.onnx` входят в Git вместе с исходной OSNet и whitening (SHA-256 — §9). Фактический trainer, оркестратор, конфиги, export и whitening перенесены с узлов в training/src/; там даны команды и границы подтверждённого происхождения.

Дельта от хранения whitening в `float32` вместо `float64`-оригинала измерена отдельно: метрики валидации не меняются ни в одной из 16 печатных цифр, максимальное отличие эмбеддинга `1.43e-08` (раздел 8).

Ранжирование и отказ

Пакетный режим по умолчанию применяет k-reciprocal re-ranking на объединении **всех запросов данного прогона и всей галереи**. Он смешивает жаккардову дистанцию между взаимными окрестностями и нормированную косинусную дистанцию. Внешняя оценка `s = 1 − d`: больше означает ближе. Сохраняемые эмбеддинги от этой операции не меняются.

Рабочая строка конфигурации (с 20.09.2026): `batch: k1=6, k2=3, λ=0.3, t_rr=0.5282812306342437; API / batch --no-rerank: t_cos=0.5141976914190476`. До 20.09 сдавалась одномодельная конфигурация OSNet+KR с `t_rr=0.49937235233589916`, `t_cos=0.5495953464415451` (её числа сохранены в истории раздела 6). Пороги перекалиброваны на эмбеддингах d1_j48 тем же правилом, что и раньше (раздел 6).

Исполняемый источник значений — config.py. После изменения калибровки обновляется строка выше; команды ниже читают значения из кода и заново рассчитывают порог. Фактическая конфигурация batch записывается в `run_info.json`, конфигурация API доступна в `/api/version`.

Область порога `t_rr`: калибровка относится к полному validation-сплиту 1110 запросов × 750 объектов (832 с парой, 278 без), d1_j48, KR(6,3,0,3), `market` + `presence`. Другой состав query/gallery, даже того же размера, требует проверки; минимального безопасного размера не установлено. На контроле 4×10 переранжирование приняло чужие автомобили с 0,541908 / 0,768959 и серый кадр с 0,801990; cosine отказал всем трём (аудит). Для малых наборов используйте API либо `batch --no-rerank` с косинусным порогом; F1/TNR большого сплита это не гарантирует. Rerank для нового набора требует своей размеченной калибровки. Автоматического fallback в сервисе нет.

HTTP API выполняет **точный косинусный поиск** Qdrant (`exact=True`). Его ответы нельзя сравнивать с пакетным переранжированием как с одним алгоритмом: состав остальных запросов влияет на пакетный результат. `--no-rerank` включает косинусный batch. В обоих режимах принимаются оценки `s >= t`; при равных оценках в batch сохраняется порядок строк gallery-CSV. Оценка близости не является вероятностью.

Порог задаётся `--threshold` для batch, полем `threshold` для API, либо переменными `REID_THRESHOLD` / `REID_THRESHOLD_RERANK` внутри процесса. Переменная, заданная только в shell хоста, не передаётся контейнеру автоматически: для `docker run` нужен `-e`, для Compose — настройка `environment`/override. API ограничивает выдачу параметром `top_k` (по умолчанию 10, максимум 100); `candidates.csv` содержит все прошедшие порог пары.

3. Подготовка к сборке и установке

Нужны Linux x86_64, Docker Engine с Compose V2, shell, Python 3.10+ для проверки входов и `curl`. GPU не требуется. Для validation достаточно её изображений; для test/галереи сервиса нужны соответствующие test-CSV и изображения. CSV из Git недостаточно. Указанные команды запускаются из корня полученного репозитория.

Отдельной компиляции Python-кода и сборки frontend нет: браузер получает готовые HTML/CSS/JS, а Dockerfile устанавливает готовые Python wheels, копирует код и проверяет веса. Порядок установки: получить данные (шаг 1), проверить веса (шаг 2), импортировать базовые образы (шаг 3), выполнить единую команду сборки и запуска. Установка Docker/Compose на хост и получение закрытого набора организатора этой командой не выполняются. Проверены Docker Engine **29.7.2** и Compose V2 **2.40.3**, а также Podman. Условия и результаты — аудит Docker-пути и проверка команд.

Шаг 1. Пути и данные

```
export REPO="$(pwd -P)"
export DATA_DIR="$REPO/data"
export OUT_DIR="$REPO/outputs/reproduce"
export COMPOSE_FILE="$REPO/04-solution/service/docker-compose.yml"
mkdir -p "$OUT_DIR"
python3 04-solution/reproduce/check_inputs.py --mode val --data-dir "$DATA_DIR" \
  --report "$OUT_DIR/inputs-val.json"
```

Архив данных получают через личный кабинет задачи № 7. Полный архив распаковывают непосредственно в `DATA_DIR`: `images/`, `train.csv`, `test_query.csv`, `test_gallery.csv`, `README.md`. Минимальные независимые наборы validation/test, способ получения для жюри и диагностика нехватки файлов — §7 и reproduce/README.md. Проверенного прямого URL архива, его SHA-256 и отдельной лицензии в поставке нет. Отпечатки распакованного набора приведены в разделе 9.

Шаг 2. Веса

```
(cd "$REPO/04-solution/service" && sh model/fetch_model.sh)
```

Скрипт проверяет по полному SHA-256 все три сдаваемых файла весов и скачивает по прямой ссылке OMZ только то, чего нет рядом (OSNet). После него файлы находятся в `04-solution/service/model/`. При несовпадении любого хеша сборку не продолжать. Оба ONNX и whitening отслеживаются Git. Исключение `!04-solution/service/model/*.onnx` в `.gitignore` включает эти ONNX вопреки общему `*.onnx`. В полном клоне скачивание не требуется.

Шаг 3. Образы

Для воспроизведения закреплённой среды используйте архивы из Git. Из корня репозитория:

```
(cd 04-solution/service/offline && sha256sum -c SHA256SUMS)
docker load -i 04-solution/service/offline/python-3.13-slim.tar.gz
docker load -i 04-solution/service/offline/qdrant-v1.15.5.tar.gz
```

После импорта переходите к §4: отдельный `docker build` для короткого пути не нужен. Сетевой `docker pull python:3.13-slim` может получить другую ревизию плавающего тега и потому не заменяет импорт проверенного архива при воспроизведении опубликованных версий.

Пакеты из PyPI на этом шаге не скачиваются: сборка идёт офлайн, из репозитория. В решении лежит каталог wheels/ — 30 файлов `.whl`, 59 146 237 байт: ровно те пакеты, что перечислены в разделе 10 (семь прямых зависимостей и их транзитивные). Колёс 30, а строк в разделе 10 — 31: `pip` приходит из базового образа и отдельным колесом не везётся. Колёса получены `pip download -r requirements.txt -c requirements-lock.txt` **в том же базовом образе** `python:3.13-slim`, поэтому платформа и версия Python у них те же, что у сборки; команда пересборки каталога — в wheels/README.md. Dockerfile по умолчанию собирается с `ARG PIP_SOURCE=offline`, то есть `pip install --no-index --find-links=/wheels -r requirements.txt -c requirements-lock.txt`. Это наш ответ на разд. 9 ТЗ: образ собирается в изолированной среде, без выхода наружу.

Для этого шага нужны два базовых образа: `python:3.13-slim` для сборки и `qdrant/qdrant:v1.15.5` для хранилища. Оба поставляются в offline/ — отдельным архивом каждый, 45,6 и 65,4 МБ, загрузка командой `docker load -i`. Архив, положенный 21.09 одним файлом, содержал только Python с двумя тегами (`podman save` без `--multi-image-archive` молча теряет второй образ); 22.09 образы разложены по отдельным архивам и проверены загрузкой в отдельное хранилище — встают оба, с правильными идентификаторами и размерами.

`--network none` изолирует только шаги `RUN`. Внешний процесс сборки может обращаться к реестру. В аудите 21.09 проверен **Podman 5.8.1**, с локальным Python base, `--pull=never` и дополнительным `unshare --net` вокруг процесса сборки. Команда после импорта base:

```
podman image exists docker.io/library/python:3.13-slim
podman build --no-cache --pull=never --network none \
  -t vehicle-reid-service "$REPO/04-solution/service"
```

Без base Podman останавливается на `image not known`; с ним офлайн-сборка прошла, freeze совпал с lock. **У `docker build --pull` — булев флаг**, `--pull=never` ему передавать нельзя (Docker reference). Для Docker заранее импортируйте base в хранилище выбранного builder, отключите внешнюю сеть **у Docker daemon / BuildKit**, затем выполните:

```
docker image inspect python:3.13-slim >/dev/null
docker build --no-cache --pull=false --network none \
  -t vehicle-reid-service "$REPO/04-solution/service"
```

`--pull=false` сам по себе не запрещает получение отсутствующего base. Отключение сети только у CLI, обращающегося к daemon, тоже недостаточно. Docker-вариант и Compose V2 проверены отдельным прогоном на Engine 29.7.2 / Compose 2.40.3 (отчёт).

Запасной путь — ставить из сети: при отсутствии каталога сначала `mkdir -p "$REPO/04-solution/service/wheels"`, затем `docker build --build-arg PIP_SOURCE=network -t vehicle-reid-service "$REPO/04-solution/service"`. Он нужен, если колёса разошлись с `requirements.txt` или требуется другая платформа. Сам каталог `wheels/` нужен в любом случае: `COPY wheels/` в Dockerfile общий для обеих ветвей, без каталога сборка не начнётся.

В сдаваемый образ колёса **не попадают**: зависимости ставятся в отдельной стадии сборки, наружу уходит только каталог установленных пакетов. Веса копируются внутрь образа и проверяются при сборке, затем повторно при загрузке модели. Транзитивные зависимости закреплены: `pip install` идёт с constraints-файлом requirements-lock.txt — полным `pip freeze --all` проверенной сборки (раздел 10). Не закреплены хеши wheel и тег базового образа `python:3.13-slim` (digest проверенной среды указан в разделе 10).

Замечание для хостов с SELinux (Fedora, RHEL, CentOS Stream и подобные)

Это касается **четырёх** команд ниже — всех, которые монтируют каталоги хоста: пакетный прогон (раздел 4), команды **T** и **R** (раздел 7), калибровка порога (раздел 8). Docker на таком хосте обычно расставляет метки сам; Podman — нет, и контейнер получает отказ в доступе к смонтированным каталогам. Проверенное решение — добавить в каждую такую команду `--security-opt label=disable`; альтернатива — суффикс `:z`/`:Z` у каждого `-v`.

Добавьте флаг до первого запуска, а не после отказа. `app.batch` пишет все три файла **в самом конце**, поэтому `PermissionError: [Errno 13] Permission denied: '/out/embeddings.npy'` приходит через 5,5 минут уже выполненного инференса — весь расчёт придётся повторить. Команда **T** падает быстро, но с сообщением не по делу: `ModuleNotFoundError: No module named 'test_protocols'` — файл на месте, просто каталог `/repo` не читается.

4. Запуск и офлайн-поставка

Сборка, установка зависимостей и запуск — одна команда

После подготовки данных и импорта двух базовых образов из §3, **из корня репозитория**, без предварительно собранного `vehicle-reid-service`:

```
docker compose -f 04-solution/service/docker-compose.yml up --build -d --pull never
```

`--build` собирает образ по Dockerfile и устанавливает зависимости из локальных wheels; Compose запускает Qdrant, API и одноразовый `loader`. Данные по умолчанию берутся из `data/` в корне. Для другого расположения перед командой задайте `DATA_DIR=/абсолютный/путь/к/data`. Назначение флагов сверено с Docker Compose reference.

Это полная команда сборки и старта приложения при подготовленных входах; полный прогон на Docker Engine 29.7.2 / Compose V2 2.40.3 **выполнен 22.09** (отчёт): пустое хранилище, импорт обоих архивов, сборка без кэша, запуск через Compose, отрицательный контроль сети демона. `--pull never` относится к образам запуска и не изолирует сеть BuildKit; условия офлайн-сборки описаны в §3. На SELinux требуется доступ контейнеров к bind-mount данных; текущий Compose не задаёт `:z`/`:Z`, и универсальный запуск без дополнительной настройки на таком хосте не доказан. `-d` возвращает управление до окончания загрузки галереи: готовность проверяют по выходу `loader` с кодом 0 и строке с числом загруженных объектов, как описано ниже. `localhost:8000` после запуска — локальный адрес, а не опубликованная ссылка для жюри.

Сервис с загруженной галереей — одна команда

Если образ `vehicle-reid-service` уже собран предыдущей командой либо импортирован из runtime-архива, повторный запуск без сборки (переменная `COMPOSE_FILE` задана в шаге 1 раздела 3):

```
docker compose up -d --no-build --pull never
```

Интерфейс: http://localhost:8000/. Swagger: http://localhost:8000/docs. OpenAPI: http://localhost:8000/openapi.json. Статические файлы Swagger поставляются локально. Загрузчик читает `$DATA_DIR/test_gallery.csv` и **пересоздаёт** коллекцию; повторный запуск заменяет её прежнее содержимое. Дождитесь завершения loader и проверьте:

```
curl --fail http://localhost:8000/api/health
curl --fail http://localhost:8000/api/version
```

В health должны быть `storage.reachable=true` и непустое `storage.gallery_points`. Отдельное поле `status="ok"` само по себе не означает готовность галереи. До загрузки поиск возвращает HTTP 409 (`{"detail":"галерея не загружена — выполните app.load_gallery"}`), при недоступной БД — 503. Оба ответа проверены.

`gallery_points` не доказывает, что отработала именно ваша загрузка. Том Qdrant именованный (`qdrant_storage`), а имя проекта Compose берётся из имени каталога с `docker-compose.yml` — это всегда `service`, одинаково у рабочего дерева и у любого клона. Поэтому `up` на машине, где сервис уже когда-то поднимали, видит **прежний** том: health отдаёт `gallery_points=750` ещё до того, как ваш загрузчик дошёл до первого кадра. Так и случилось при сквозной проверке из чистого клона.

Достоверный признак — **завершение самого загрузчика**, а не содержимое хранилища:

```
docker compose logs --no-log-prefix loader | tail -1
# {"collection": "gallery", "points": 750, "rows": 750}
```

Эту строку `app.load_gallery` печатает последним действием — после `upsert` и проверки `points == rows`; до неё процесс не завершается, а при несовпадении падает. Нет строки — загрузка не дошла до конца, чем бы ни отвечал health. Проверено и под podman-compose: вывод тот же. Код выхода контейнера виден в `docker compose ps -a loader` (столбец `State` — `exited (0)`).

Запуск с заведомо пустым хранилищем (нужен, если прежний том мог остаться от другого прогона):

```
docker compose down -v          # удаляет контейнеры И именованный том проекта
docker compose up -d --no-build --pull never
```

Проверено на чистом томе: `gallery_points` равен `null` всё время работы загрузчика и становится `750` сразу после его выхода; загрузка 750 объектов заняла 64,5 с. Если удалять чужой том нежелательно, дайте прогону собственное имя проекта — `docker compose -p reid-check up -d …`, тогда создаётся отдельный том `reid-check_qdrant_storage`, а `docker compose -p reid-check down -v` убирает за собой только его.

Загрузка галереи входит в `up`: `loader` — обычный разовый сервис Compose (профиля `tools` больше нет), он стартует после healthcheck Qdrant, отрабатывает и выходит. Готовность Qdrant проверяется healthcheck-запросом `/readyz` (в образе нет curl, поэтому проверка идёт через `/dev/tcp` bash), а `api` и `loader` объявлены зависимыми с `condition: service_healthy`. Дополнительно сам загрузчик ждёт доступности хранилища до извлечения векторов (`--wait`, по умолчанию 120 с) — это нужно для сред, где ожидание healthcheck не выполняется (поведение зависит от версии и стенда; podman-compose

1.6.0 в отдельном storage аудита ожидал healthcheck, но его systemd unit не видел это storage). Повторная загрузка на другом каталоге данных: `DATA_DIR=/путь/к/data docker compose run --rm loader` — эта форма блокирует терминал до конца загрузки и печатает ту же итоговую строку, то есть тоже годится как признак готовности.

`docker compose up -d` возвращает управление, пока загрузчик ещё работает. Замер 64,5 с выше относится к исторической проверке; для текущего ансамбля d1_j48 время зависит от лимита CPU и состояния машины. До конца загрузки поиск отвечает 409 — это ожидаемо, а не отказ сервиса.

Пакетный инференс — одна команда, сеть отключена

```
docker run --rm --network none \
  -v "$DATA_DIR:/data:ro" -v "$OUT_DIR:/out" \
  vehicle-reid-service python -m app.batch \
  --images-dir /data/images --query /data/test_query.csv \
  --gallery /data/test_gallery.csv --out-dir /out --threads 2
```

Для закрытого теста заменяются только входной каталог и CSV. Метки `vehicle_id` и `camera_id` инференсу не нужны. Выходной каталог должен быть доступен на запись. **На хосте с SELinux добавьте `--security-opt label=disable`** — см. замечание в конце раздела 3; там же сказано, почему отказ приходит только через 5,5 минут.

Передача на машину без интернета

На машине сборки сохраните **оба** образа:

```
docker save -o "$OUT_DIR/runtime-images.tar" vehicle-reid-service docker.io/qdrant/qdrant:v1.15.5
sha256sum "$OUT_DIR/runtime-images.tar" > "$OUT_DIR/runtime-images.tar.sha256"
```

Podman вместо Docker: `podman save -o файл образ1 образ2` молча кладёт в архив только один образ (оба тега при этом навешиваются на него) — нужен `podman save --multi-image-archive -o ...`; проверено на podman 5.8.2 для `vehicle-reid-service` + `qdrant/qdrant:v1.15.5`: с флагом в архиве два образа и **529 025 536** байт, без флага — один образ и **347 906 048**. Побайтово эти размеры не воспроизводимы и сверять их не нужно: слои образа содержат отметки времени файлов, поэтому две независимые сборки одного и того же Dockerfile дают чуть разные архивы. Контрольный повтор на другой сборке из чистого клона: **528 972 288** и **347 852 800** — те же величины с точностью 53 248 байт (0,01 %). Проверяемое утверждение здесь другое и оно выполняется точно: **без флага в архиве один образ с двумя тегами, с флагом — два образа**. Передайте архив образов, код вместе с тремя файлами `model/` и разрешённый комплект данных организатора. На целевой машине выполните `docker load -i /путь/к/runtime-images.tar`, задайте пути из шага 1, затем используйте команду запуска выше без `--build`. Для пакетного режима достаточно образа сервиса и тестовых данных. **Данные организатора передаются отдельно от Git.** Для пересборки без сети в текущем комплекте есть два исправленных архива `offline/python-3.13-slim.tar.gz` и `offline/qdrant-v1.15.5.tar.gz` (инструкция); старого `base-images.tar.gz` больше нет. Оба ONNX, whitening и wheels входят в Git; работающий образ сервиса содержит веса. Отдельная приёмка на Docker Engine 29.7.2 / Compose V2 2.40.3 выполнена; условия и результаты.

5. Формат сдаваемых файлов

Схема основана на `README.md` архива организатора, а не только на общих формулировках ТЗ.

Первичные требования: ТЗ §8, стр. 5 PDF, и раздел «Как сдавать решение → Артефакты в репозитории» выданного `data/README.md` (SHA-256 `b517186daf5186f8ace0b3d2feb9fbce328cd09e620f22b84ff9a2a292d4575c`). Этот README входит в архив данных, но не в Git; его точная схема раскрыта ниже. В ТЗ число строк определяется входными CSV: 1110/750 — размеры данного выданного набора, а не требование ко всякому закрытому тесту.

Файл	Содержимое
`submission.csv`	Заголовок `query_id,gallery_id_1,…,gallery_id_10`; строки query в порядке входного CSV, кандидаты по убыванию оценки; десять значений только при `N_gallery >= 10`
`embeddings.npy`	`float32`, `(N_query + N_gallery, 512)`; сначала все query, затем все gallery, порядок внутри блоков как в CSV
`candidates.csv`	`query_id,gallery_id,confidence`; все пары с исходной оценкой `s >= t`, без лимита десятью; уверенность записана с шестью знаками после точки; отсутствие строк означает отказ при уникальных query ID и успешном прогоне
`run_info.json`	Служебный протокол: модель, хеш, версии, порог, режим и шкала, размеры, счётчики, время

Переранжирование меняет `submission.csv` и `candidates.csv`. Оно не меняет `embeddings.npy`. Старые файлы baseline/artifacts сняты в другом режиме/при другом пороге и не служат эталоном новой выдачи. Каталог service/artifacts-final/ перевыпущен **21.09.2026**: прогон текущей конфигурации d1_j48 с рабочими порогами раздела 2, сверен побайтово с пересчётом из чистого дерева (раздел 8). Явный manifest выпуска с размерами, хешами и входными CSV — manifest.json, проверка — `sha256sum -c SHA256SUMS` в том же каталоге. У проверенного выданного теста 1110 query и 750 gallery: ожидается матрица `(1860, 512)`.

Детали сериализации и границы требований. Оба CSV выпуска используют запятую, ASCII-совместимый UTF-8 без BOM и окончания строк CRLF. Идентификаторы сохраняются строками из входных CSV. Организатор показывает заголовки с запятыми, но отдельно не регламентирует BOM, кодировку, CRLF/LF, число знаков уверенности, dtype/размерность NPY и числовой диапазон confidence. `float32`, 512, L2-нормировка и шкала `1 − d` — выбранные свойства этого решения.

Топ-10 обязателен только в `submission.csv`: при `N_gallery >= 10` все запросы, включая отказные, имеют десять кандидатов. При меньшей непустой галерее строки короче заголовка — это незакрытый дефект формата (F2 ниже). В `candidates.csv` ограничение десятью в первичных материалах не задано: записываются все принятые пары. Пустой ответ реализован отсутствием строк данного `query_id`; строки `query_id,,` , `NaN` или `-1` не добавляются. Организатор требует «пустой ответ», но его физический sentinel в CSV не уточняет; совместимость этой конвенции с закрытым парсером жюри документально не подтверждена. При чтении нужно проверить код завершения batch и соединить результат со **всеми** query из входного CSV; это восстанавливает отказы только при уникальном `image_id` в query. Разные bbox с одним ID по выходному CSV не различить (F4 ниже). Сравнение с порогом выполняется **до** округления confidence до шести знаков; повторная фильтрация по записанному confidence может изменить решение (F6).

Известные границы поведения

Эти ограничения относятся к `app.batch` текущего выпуска d1_j48. Они подтверждены аудитом формата и проверкой исправлений F1/F5. Численный профиль и порядок объектов дополнительно проверены независимым критиком границ. Ниже — оставленные границы, а не заявленные исправления.

- **F2 — галерея меньше десяти объектов.** При одном объекте заголовок `submission.csv` содержит 11 полей, строки — два; batch завершается с кодом 0. Вообще при `1 <= N_gallery < 10` пустые позиции не дополняются. Исправление отложено до согласования заполнителя с организатором: формат недостающих ID не определён. Для прямоугольного submission нужны минимум десять строк галереи; код 0 сам по себе не гарантирует эту схему.
- **F3 — пустая галерея.** При непустом query и `N_gallery = 0` batch завершается с кодом 1, новых файлов нет; продуктовый ответ «совпадений нет» не формируется. Пустой query также не поддержан. Обработка оставлена до согласования формата пустого комплекта. Перед запуском нужны непустые CSV; проверяйте код возврата, поскольку прежние файлы в `out-dir` могут остаться.
- **F4 — повторяющиеся `image_id`.** Разные bbox одного query имеют одинаковый ключ результата: join неоднозначен, отказ одного bbox может скрыться за принятым совпадением другого. Дубликаты gallery повторяют ID в топе и пары в `candidates.csv`. Исправление требует согласованного идентификатора объекта вместо одного ID кадра; схему выпуска не меняем. Для однозначной выдачи обеспечьте уникальность `image_id` отдельно в query и gallery до запуска: сам batch её не проверяет.
- **F6 — округление confidence.** Пара принимается по исходному `s >= t`, затем `s` записывается с шестью знаками. На границе записанное значение может оказаться ниже полного порога. Точность CSV оставлена прежней, чтобы сохранить формат выпуска и решение по неокруглённой оценке. Наличие пары в `candidates.csv` означает принятие; не фильтруйте её повторно по округлённому confidence.
- **F7 — больше десяти принятых пар.** `candidates.csv` содержит все пары, прошедшие порог; в аудите при явном `--threshold 0` и галерее из 15 объектов получено 15 на запрос (при штатном пороге на том же наборе — 3 и 0). Усечение не добавлено: первичные требования задают десять только для `submission.csv`. Читатель `candidates.csv` должен поддерживать произвольное число принятых строк; для первых десяти используется `submission.csv` независимо от отказа.
- **Остаток F1/F10 — отказ точной копии и численная чувствительность.** Число `0.43999999999999984` (около `0.44`) воспроизведено на синтетическом наборе форматного аудита **37 query × 37 gallery**, с исходным порядком строк, кодеками и bbox. Это не 37 реальных автомобилей организатора. Профиль: d1_j48, `--batch 32 --threads 2`, KR `(6, 3, 0.3)`, `OPENBLAS_NUM_THREADS=1 OMP_NUM_THREADS=1 MKL_NUM_THREADS=1`, Python 3.13.15, NumPy 2.5.3, Pillow 12.3.0, ONNX Runtime 1.30.0, worker-vm, runtime-образ `localhost/job72-jury:6179b69`. Строки с индексами **0, 3, 5, 12** (`rgb.jpg`, `progressive.jpg`, `exif_1.jpg`, `exif_8.jpg`, bbox `(0, 0, 128, 96)`) имеют побайтно равные query/gallery-векторы, но `self-score = best = 0.43999999999999984`; штатный порог `0.5282812306342437` отклоняет **4/37** запросов. На том же наборе `--no-rerank` даёт **0/37** отказов. При изменении только BLAS/OMP/MKL с 1 на 2 потока тот же KR также дал **0/37** отказов: отличия компонентов эмбеддингов до `5.21540641784668e-08` изменили решение. На свежих векторах профиля 1 все 20 проверенных перестановок только gallery меняли scores (максимум `|Δscore| = 0.56`, до 105 решений по парам и четырёх отказов запросов). Это численная проверка rerank на уже вычисленных векторах, а не 20 дополнительных инференсов. Состав, порядок строк и профиль потоков — существенные условия примера; `run_info.json` сам по себе не содержит достаточного профиля повторения. Входы, их SHA и команды приведены в отчёте критика выше. Защита от NaN исправлена, этот дефект ранжирования остался. Исправление отложено до отдельной проверки алгоритма KR и калибровки. Наличие точной копии не гарантирует

принятия; для малых наборов используйте рекомендацию по косинусному режиму в §2, без гарантии метрик большого сплита.

6. Протокол метрик и достигнутые значения

Валидация находится в split/files: 1110 запросов, 750 объектов галереи. У 832 запросов есть кросс-камерная пара, у 278 пары нет. Идентичности `train_fit` отделены от валидации. SHA-256 CSV закреплены в manifest.json.

Ранжирование оценивается с `camera_policy="market"`: из галереи запроса удаляются объекты **той же идентичности и той же камеры**. Другие автомобили с той же камеры остаются отрицательными кандидатами. ТЗ допускает и трактовку «удалить всю камеру»; она реализована как `all_same_camera`, но приведённые числа относятся к `market`.

AP — среднее precision в позициях всех верных допустимых совпадений; mAP — среднее AP по **832** запросам с парой. Rank-k — доля таких запросов с верным объектом среди первых k. mINP — среднее отношения числа верных объектов к рангу последнего верного объекта. Для mAP@10 сначала отбираются десять исходных кандидатов, потом применяется фильтр камеры; знаменатель AP — все верные в допустимой галерее. Если жюри усреднит AP по всем запросам с нулями для отсутствующих, результат будет другим; обе ветви записывает контур.

Метод	mAP, вся галерея	Rank-1	Rank-5	mINP	mAP@10
d1_j48, косинус	0.7316093422	0.6850961538	0.8822115385	0.6928460434	0.7242262382
d1_j48, переранжирование	**0.7740915539**	**0.7307692308**	0.8810096154	0.7488060310	**0.7684546226**
Прежняя сдаваемая конфигурация OSNet+KR, переранжирование (до 20.09)	0.6936584725	0.6634615385	0.7872596154	0.6608640365	0.6834008859

Прирост mAP от переранжирования у d1_j48 — **0.0424822117** (косинус → переранжирование). Значения воспроизведены сервисным конвейером из изображений и совпали с измерительным контуром до всех 16 цифр (валидационный батч сервиса, прогнанный через `tools/eval_split.py`, даёт ровно `0.7740915539438481 / 0.7307692307692307 / 0.7684546226343101` и косинус `0.7316093422310448`); источник чисел и методика — training/combined/s02_metrics.json и training/combined/REPORT.md. Числа полного ранжирования нельзя объявлять метрикой усечённого `submission.csv`. Метрики закрытого теста организатора неизвестны: в доступных test-CSV нет идентичностей и камер.

> **Пометка честности (переключение 20.09.2026).** Сдаваемая конфигурация сменилась: OSNet+KR (0.6937 / 0.6635) → d1_j48 (0.7741 / 0.7308). Отрыв d1_j48 от **прежнего чемпиона измерительного контура** d1 (0.7621 / 0.7175) составляет **+0.0120 — меньше заранее зафиксированной приёмочной σ(mAP) = 0.0134**, то есть по приёмочному правилу прирост против чемпиона **не засчитывается**. Парный бутстрэп при этом значим (Δ = +0.0120, CI95 [+0.0040, +0.0205], p = 0.003, B = 4000); переключение выполнено **по явному решению Артёма** при этом значимом бутстрэпе. Полное сравнение кандидатов и критерий приёмки — training/combined/REPORT.md §4–§6. Разрыв по камере (KR mAP: без исключения same-camera минус market) у сдаваемой конфигурации **+0.1103** — лучше, чем у прежнего чемпиона

> (+0.1144) и у исходного OSNet (+0.1417); абляция зоны номерной пластины для combined_v1 пройдена — снижение при её маскировании статистически неотличимо от контрольной зоны той же площади (все контрасты накрывают 0, REPORT.md §5).

Обоснование порога

Единица отказа в основной ветви — **запрос**, `refusal_mode="presence"`. Положительный класс означает наличие пары в допустимой галерее. Принятие определяется максимумом оценки после камерной фильтрации. Это не precision/F1 всех строк `candidates.csv` и не проверка правильности идентичности top-1.

Правило: выбрать порог, максимизирующий `min(TNR, F1_0.10, F1_0.25, F1_0.40)`. Индексы — предполагаемые доли запросов без пары. Для каждого порога вычисляются recall `r` и доля ложных принятий `f`; при доле отсутствующих `p` используется `F1_p = 2(1−p)r / ((1−p)(1+r)+pf)`, `TNR=1−f`. При изменении только доли классов TNR и recall сохраняются. Это обоснование выбора компромисса, а не модель распределения скрытого теста.

Формула следует из долей `TP=(1−p)r`, `FN=(1−p)(1−r)` и `FP=pf`, подставленных в `F1=2TP/(2TP+FP+FN)`. Критерий максимизирует худший из четырёх показателей: высокий F1 не компенсирует плохие отказы, а отказ всем запросам не компенсирует нулевую полноту. Доли 0.10/0.25/0.40 — наша сетка проверки устойчивости вокруг фактических `278/1110 ≈ 0.25045`, а не известная статистика скрытого теста и не обязательные веса из ТЗ. Максимум F1 на косинусе отвергает лишь 84 из 278 отсутствующих (TNR 0.3022); рабочий косинусный порог отвергает 203 (TNR 0.7302), ценой снижения recall с 0.9435 до 0.6803. Рабочее переранжирование даёт 218 верных отказов и 60 ложных принятий: `TNR=218/278`, `F1=1212/(1212+60+226)`. Тем самым порог имеет проверяемую цель и явную цену компромисса. Исходные точки — headline.json, полный перебор — summary.json.

Доопределение при равенстве — часть правила. Кандидаты — все различные значения уверенности top-1 на валидации и `nextafter(max_score, +inf)` для режима «отклонить всё»; в сохранённой калибровке по 1111 точек на каждую шкалу. Максимум не всегда единственный: когда минимум даёт `TNR`, целевая функция постоянна на участках сетки, где не сдвигается ни один запрос **без** пары. При равенстве берётся больший `F1`, затем больший `TNR`. На косинусе это не формальность: плато состоит из двух порогов — `0.5141976914190476` (TP 566, F1 0.7684996606) и `0.5145892426611729` (TP 565, F1 0.7676630435), у обоих `TNR` 0.7302158273; без доопределения выбор между ними произволен, а расходятся они на 3.9e-04 — в 400 раз больше допуска сверки в команде R. Правило записано так во всех реализациях: `calibrate_threshold.py` (им посчитаны рабочие пороги), `refusal/scripts/analyze.py`, `s06_refusal.py` и команда R (разд. 7).

Режим	TP / FP / FN / TN отсутствующих	F1	TNR	Recall	AUC-PR, трапеции
Максимум F1 на косинусе d1_j48 (не рабочая точка)	785 / 194 / 47 / 84	0.8669243512	0.3021582734	0.9435096154	0.9094743821
Косинус d1_j48, `t_cos`	566 / 75 / 266 / 203	0.7684996606	0.7302158273	0.6802884615	0.9094743821
Переранжирование d1_j48, `t_rr`	606 / 60 / 226 / 218	0.8090787717	0.7841726619	0.7283653846	0.9182837159

Значения перекалиброваны 20.09.2026 на эмбеддингах d1_j48 прежним правилом; инструмент — `tools/calibrate_threshold.py`, 187 прямых сверок объявленных точек с неизменённым контуром (max |ошибка| < 2e-12). До 20.09 (прежняя конфигурация OSNet+KR, калибровка refusal/): `t_cos` 0.5495953464415451 → F1 0.7330567082 / TNR 0.6978417266; `t_rr` 0.49937235233589916 → F1 0.8056112224 / TNR 0.7769784173.

Какая строка относится к сдаваемому режиму. По умолчанию batch работает с переранжированием, поэтому заявленные в задании **F1 0.8090787717 и TNR 0.7841726619 — это строка «Переранжирование d1_j48, `t_rr`»**, и в `metrics.json` она лежит в `rerank_refusal`. Пара 0.7684996606 / 0.7302158273 из строки «Косинус d1_j48, `t_cos`» относится к косинусной ветви — это режим HTTP API и `batch --no-rerank`, в `metrics.json` это `base.refusal`. Числа одинаковые по смыслу и разные по конфигурации; команда R (раздел 7) печатает **обе пары с подписями**, чтобы их нельзя было перепутать при сверке.

Косинусная калибровка прежней конфигурации описана в refusal/REPORT.md; текущие обе точки пересчитываются командой R и сверяются `calibrate_threshold.py`. Порог нельзя округлять перед сравнением или переносить между шкалами.

Параметры переранжирования выбраны на отдельном протоколе из `train_fit`, но рабочие пороги выбраны и показаны на одной валидации. Новая калибровка не имеет независимого holdout-замера. В сервисе нет скрытых идентичностей для камерной фильтрации: число непустых ответов сырого batch на validation может отличаться от TP+FP в таблице. На выданном тесте число отказов проверяемо, их правильность — нет.

7. Как повторить наши числа

В Git есть код, веса и CSV сплита, **но нет изображений организатора**. Validation-R требует ровно **1860 исходных JPG** по `image_id` из `04-solution/split/files/val_query.csv` (1110) и `val_gallery.csv` (750). Список **каждого имени, размера и SHA-256** — reproduce/inputs-manifest.json, ключ `data.val`; отпечатки CSV — `repo_files` и split/files/manifest.json. Test-R отдельно требует `test_query.csv`, `test_gallery.csv` и 1860 других кадров (`data.test` того же manifest). Отсутствие test не препятствует validation.

Участник получает исходный набор через личный кабинет задачи №7. Жюри использует **тот же выданный участникам набор** из своих материалов либо получает его у постановщика. Отдельного проверенного канала выдачи жюри и публичного прямого URL архива у нас нет. Если кабинет недоступен, данные нужно запросить у организатора; из этого репозитория восстановить изображения нельзя. Изображения не публикуются в Git. Закрытый тест жюри не является данным validation-сплитом и не воспроизводит наши validation-числа.

```
$REPO/04-solution/split/files/{val_query,val_gallery,train_fit,split_assignment}.csv
$DATA_DIR/
  images/<image_id>.jpg       # имена из manifest, без дополнительной папки images/images
  test_query.csv              # только для R-test / R-all
  test_gallery.csv            # только для R-test / R-all
  train.csv, README.md        # для подготовки обучения; R-val их не требует
```

SHA-256 test_query.csv — `97e1ed21942bae9c95b1ce2e5d339d9f635bf49fa484367e6b19349789bb9b4c`, test_gallery.csv — `a64ed21fa39bbfd8c7a172468d043415800500b6ed695cdb8162188cf9005496`. Полный исходный набор и агрегированный отпечаток изображений — §9.

Сначала проверить входы

На хосте достаточно Python 3.10+, без NumPy/torch/ONNX Runtime:

```
python3 04-solution/reproduce/check_inputs.py --mode val --data-dir "$DATA_DIR" \
  --report "$OUT_DIR/inputs-val.json"
```

Проверяются **все** нужные файлы, размеры и SHA-256. При нехватке данных — exit 2, число отсутствующих файлов, примеры имён и каталог назначения; полный список записывается в JSON. Инференс не начинается. Для test замените `val` на `test`. Эта же проверка встроена в каждую команду R ниже; флаг `--check-only` запускает только её. Подробнее — README контура.

R-val: валидация, метрики и оба порога

После сборки/импорта образа из §3–4, из корня репозитория:

```
docker run --rm --network none --cpus 2 --memory 4g \
  -e OPENBLAS_NUM_THREADS=1 -e OMP_NUM_THREADS=1 -e PYTHONDONTWRITEBYTECODE=1 \
  -v "$REPO:/repo:ro" -v "$DATA_DIR:/data:ro" -v "$OUT_DIR:/out" \
  vehicle-reid-service python -B /repo/04-solution/reproduce/run.py \
  --mode val --data-dir /data --out-dir /out
```

На SELinux добавьте `--security-opt label=disable` (§3). Скрипт использует исходники из смонтированного репозитория и зависимости образа; модели проверяет неизменённый Embedder. Переопределения `REID_*` для эталонного прогона запрещены. Он заново извлекает признаки, вызывает штатный контур метрик и правило выбора порогов, проверяет порядок векторов отдельным инференсом граничных строк. Готовые `.npy` ему не нужны. Выход: `val/`, `inputs-val.json`, `metrics.json`. Поля и арифметика validation сохранены из прежней команды R; допуск для mAP, Rank-1/5 и обоих порогов — `1e-6`. F1/TNR даны для `market` + `presence`.

Ожидаемые основные строки (последние цифры порога зависят от CPU/потоков):

```
cosine: mAP 0.7316093422310448, Rank-1 0.6850961538461539,
        threshold ≈ 0.5141976914190476, F1 0.7684996605566871, TNR 0.7302158273381295
rerank: mAP 0.7740915539438481, Rank-1 0.7307692307692307,
        threshold ≈ 0.5282812306342437, F1 0.8090787716955942, TNR 0.7841726618705036
```

Источник эталонов: s02_metrics.json и calib-d1_j48/. R не восстанавливает камерные эксперименты, абляции и bootstrap всех исследовательских отчётов (§8, §11).

R-test: только сдаваемые файлы выданного теста

```
docker run --rm --network none --cpus 2 --memory 4g \
  -e OPENBLAS_NUM_THREADS=1 -e OMP_NUM_THREADS=1 -e PYTHONDONTWRITEBYTECODE=1 \
  -v "$REPO:/repo:ro" -v "$DATA_DIR:/data:ro" -v "$OUT_DIR:/out" \
  vehicle-reid-service python -B /repo/04-solution/reproduce/run.py \
  --mode test --data-dir /data --out-dir /out
```

Выход: `test/{submission.csv,embeddings.npy,candidates.csv,run_info.json}`, `inputs-test.json`, `test-report.json` с хешами. Метрики качества test не считаются: его `vehicle_id`/`camera_id` не выданы. Manifest финального выпуска **21.09.2026** — artifacts-final/manifest.json. `run_info.json` содержит время, поэтому его хеш пересчёта будет другим. Для общего прежнего пути используется `--mode all`; он заранее требует обе части и пишет `metrics.json` с обеими batch-записями. Для произвольного закрытого теста используйте обычный `app.batch` из §4: manifest R-test намеренно фиксирует именно выданный участникам набор, а не любой набор того же размера.

T: тесты контура; V: версии

```
docker run --rm --network none -e PYTHONDONTWRITEBYTECODE=1 \
  -v "$REPO:/repo:ro" -w /repo/04-solution/eval \
  vehicle-reid-service python -B -m unittest -q test_metrics test_protocols
docker run --rm --network none vehicle-reid-service python -m pip freeze --all
```

Дополнительно tools/calibrate_threshold.py выполняет 187 сверок точек с контуром (§8); tools/eval_split.py оценивает уже извлечённые validation-векторы. Эти инструменты читаются из Git, Dockerfile не копирует их в образ.

8. Таблица «число → команда» и границы воспроизведения

Проверка исправленного R-val 21.09.2026: на отдельном снимке текущих исходников, смонтированном read-only в сохранённый образ аудита job_72, из исходных 1860 JPG получены ровно mAP/Rank-1/F1/TNR таблицы §7. В каталоге данных не было test-CSV и test-кадров. Оба порога пересчитаны с отклонением менее 1.1e-8; 60 тестов метрик и 5 тестов ранней проверки входов прошли. Протокол и полные результаты. Этот прогон не является новой сборкой, свежим Git-клоном или повтором полного Docker/Compose-стенда. Ниже исторические проверки отдельно отмечены датами. Отдельный R-test тоже прошёл: три сдаваемых файла совпали по SHA-256 с выпуском 21.09, run_info отличается только временем. Дополнительная калибровка через код сервиса на новых validation-векторах прошла 187 сверок, max_abs_error=0.0.

Числа/утверждение	Как получить	Результат проверки
1110 query, 750 gallery, 832 с парой, 278 без пары; размерность 512	R → `split`, `batch`, `row_order`	Пересчитано; SHA-256 сплита и разделение идентичностей проверены
Все mAP, Rank-1/5, mINP, mAP@10 из §6 и прирост 0.0424822117	R → `base`, `reranked`, `delta_mAP`	Совпали с s02_metrics.json; допуск проверки `1e-6`. На референс-окружении VM сервисный батч совпал с контуром точно, до всех 16 цифр (`tools/eval_split.py`)
Оба рабочих порога, F1/TNR/Recall, TP/FP/FN/TN, AUC-PR из §6	R → `calibration`, `rerank_calibration`, `base.refusal`, `rerank_refusal`, `old_threshold`	Пересчитаны через настоящий `evaluate()`; сверены с калибровкой 20.09 (service/calib-d1_j48/)
Параметры `(6,3,0.3)` как оптимум подбора	**Не команда, а историческая справка.** Сетку считал `s04_grid.py <label> <query.npy> <gallery.npy> <query_meta.csv> <gallery_meta.csv>` — пять обязательных аргументов, без них `ValueError` на разборе `argv`. Векторов, которые он ждёт (`04-solution/postproc/out/*.npy` из `s03_extract.py`), в git нет: `.gitignore` исключает `*.npy`. Сохранён результат: grid_tune_base.json	Применение параметров проверено командой R; исходная сетка заново не воспроизведена из чистого git
8 836 743 байта и SHA-256 сдаваемого OSNet	R → `model`; `sha256sum 04-solution/service/model/*.onnx`	Скачано заново, размер и хеш совпали

Числа/утверждение	Как получить	Результат проверки
Второй ONNX combined_v1: 8 742 779 байт, `b1ba5021…02efb2`; матрица whitening f32: 1 051 114 байт, `eb4433ff…4b3c5b5`; f64-оригинал матрицы: 2 101 738 байт, `4b02f158…267eb625`	`sha256sum 04-solution/service/model/osnet_ain_combined_v1.onnx 04-solution/service/model/lw_ens_j48_rho0.5.npz 04-solution/training/combined/lw_ens_j48_rho0.5_f64.npz`	Совпали с `config.py` и проверкой в Dockerfile; f64-оригинал нужен только для пересчёта дельты f32
Дельта метрик от f32-хранения whitening	скрипт прогона job_55 (сравнение A f64 / B f32-хранение / C f32-сервис на одних векторах валидации)	`d_kr_mAP = 0.0`, `d_cos_mAP = 0.0` (все 16 печатных цифр совпали); max | Δэмбеддинг| = 1.4282077284710759e-08
Порядок `(query; gallery)` в `embeddings.npy`	R → `row_order`, независимый batch=1 на границах	Проверено переизвлечением, а не чтением отчёта
60 тестов метрик	T	Пройдены (15–16.09), повторены 20.09 и 21.09; новый лог
187 прямых сверок новой калибровки (20.09)	`docker run --rm --network none -v "$REPO:/repo:ro" -v "$OUT_DIR:/out" vehicle-reid-service python -B /repo/04-solution/service/tools/calibrate_threshold.py /out/val/embeddings.npy /out/calibration`	Пройдены на независимо извлечённых векторах конфигурации d1_j48; результат — headline.json и `verification.json`
Точные версии всех пакетов образа	V	Полный freeze проверенной сборки — §10
Исторические 43.8 мс/объект и 0.55 с на rerank	**Не команда, а историческая справка.** Замер делал `s09_timing.py`. Ему нужны исследовательское окружение (раздел 10), каталог изображений (`REID_DATA_DIR`) и промежуточные векторы `04-solution/postproc/out/*.npy` из шага `s03_extract.py`; `.npy` в git не входят. Из чистого клона скрипт падает трейсбеком `FileNotFoundError` на первом же недостающем входе — запускать его не нужно. Сохранён результат: s09_timing.json	Точные исторические тайминги не воспроизведены. R выполняет новый ограниченный замер, с другим объёмом выборки и лимитом CPU
Эффект зоны пластины около −0.003 mAP, 95% CI [−0.011; +0.004]; расширенная маска: верхняя граница +0.016	**Не команда, а историческая справка.** Считал `eval_variants.py` по векторам 13 вариантов из `plate-ablation/work/emb/`; их в git нет. Запущенный в исследовательском окружении (раздел 10), скрипт это и сообщает — понятным сообщением, а не трейсбеком: `нет векторов вариантов в …/work/emb (13 из 13, например base)`	**Не воспроизведено из git:** нет `work/emb`, детекций и исходной ручной разметки. Числа только из ablation.json и отчёта
Качество локализации пластины, bootstrap/crossfit и расширенные EDA-числа	Команды соответствующих plate, refusal, EDA отчётов	Полная цепочка не восстановлена; не включать в перечень независимо воспроизведённых результатов
Размеры, SHA-256 и версии внешних источников	`sha256sum FILE`, `wc -c < FILE`, V; manifest внешних eval-исходников	Измерены для локальных файлов; неизвестные версии и хеши датасетов явно отмечены в §9

Про строки «историческая справка». Скрипты `04-solution/postproc/scripts/` и `04-solution/plate-ablation/scripts/` — рабочие записи исследования, а не часть сдаваемого пути; **запускать их для проверки решения не нужно.** Из чистого клона они и не отработают: в образ сервиса они не копируются, системный `python3` даёт `ModuleNotFoundError: No module named 'onnxruntime'` (им нужно исследовательское окружение из раздела 10), а в этом окружении они упираются в промежуточные входы, которых в git нет (`*.npy`, `work/`, каталог изображений). Проверено: `eval_variants.py` сообщает об этом понятной строкой, `s09_timing.py` и `s04_grid.py` — трейсбеком `FileNotFoundError` и `ValueError` соответственно. Всё, что из этих цепочек вошло в заявленные числа, сохранено рядом в `out/*.json`, а сдаваемые метрики пересчитывает R-val без этих исследовательских entry point и их кэшированных эмбеддингов (библиотека common.rerank используется напрямую).

Скорость зависит от оборудования и его загрузки; значения R — справочный замер, не FPS стандартизованного скрипта жюри. Ранжирование работает квадратично по памяти: одна матрица `float64` занимает $8(Q+G)^2$ байт, одновременно используется несколько. Заявления о галерее на миллион объектов или стабильном real-time-потоке не проверены.

9. Все использованные внешние модели, данные и исходники

Ниже раскрыты восемь известных источников данных, включая предобучение внешних моделей; семейство датасета не считается точной версией его архива. Для наших входов без номера релиза используются SHA-256 файлов/архивов. Это позволяет сравнить два полученных экземпляра, но не восстанавливает неизвестную ревизию upstream и не обеспечивает доступ к закрытым данным. Неизвестные версии явно помечены: требование о **всех** версиях пока закрыто не полностью. Библиотеки и окружения перечислены в §10.

Модели и веса

Назначение	Версия, источник и лицензия	Размер и SHA-256
Сдаваемый OSNet-AIN (модель 1 конвейера)	Open Model Zoo **2022.1**, `vehicle-reid-0001`. Прямой ONNX, карточка релиза 2022.1.0. MIT для оригинальной модели, текст лицензии	**8 836 743 байта** (8.427 MiB; 8.837 MB). `4aaad3e5db648618b0df3d2ff21c61323985ff9e50194c3d2edd4fb87c92d91f`
Сдаваемый combined_v1 (модель 2 конвейера)	Наш дообученный OSNet-AIN: LP-FT на RoundaboutHD (MIT) + CARLA (Apache-2.0) + train организатора. Внешнего URL нет; журнал — training/combined/. Поставляется в Git (`!04-solution/service/model/*.onnx`)	**8 742 779 байт**. `b1ba5021275b34079a1653608bdfd215fc9404306dc909852e4cdfa03402efb2`
Сдаваемая матрица whitening	Наш артефакт: `P`, `m` (ρ=0,5), обучены на `train_fit` по дельтам ансамбля; поставляется в `float32`, f64-оригинал — training/combined/lw_ens_j48_rho0.5_f64.npz	**1 051 114 байт**. `eb4433ffd5e38d3751d5cb04e234090060a83be6d1720b47bcf2fa1274b3c5b5` (f64: 2 101 738 байт, `4b02f158fa54d89a54f9ad30d7e471f1baa14d7019ea9ea26b42ea7c267eb625`)
Сравнение второй модели, в сервис не включено	FastReID SBS ResNet50-IBN, веса релиза **v0.1.1**. Прямая загрузка, Model Zoo. Код Apache-2.0; отдельная лицензия checkpoint в исследовательском пакете не найдена	**198 261 759 байт**. `57fb9c17d88911ea64390bf5427f43511435e7f88f6eed9dbc969d4b611e53cd`

Назначение	Версия, источник и лицензия	Размер и SHA-256
Локальная производная предыдущего checkpoint	`veri_sbs_R50-ibn.model_only.pth`, сохранение `ckpt["model"]` скриптом s07_fastreid_extract.py; отдельного внешнего URL нет	**99 273 627 байт**. `8595fa79eeb09f35c565729be1e8826e3b1de7951c86bff0aff048c78529076f`

Весовой лимит ТЗ (≤ 2 ГБ) проверяется по сумме трёх сдаваемых файлов: 8 836 743 + 8 742 779 + 1 051 114 = **18 630 636 байт**. Оба ONNX и матрица whitening входят в Git и проверяются по SHA-256 при сборке образа и при загрузке модели. FastReID использовался при исследовании, поэтому раскрыт, хотя в runtime не нужна. Отчёт постобработки объясняет отказ от ансамбля и TTA на том этапе; итоговый ансамбль d1_j48 — training/combined/REPORT.md.

Обучающие источники публичных весов

OSNet обучен не только на VeRi. Upstream README.rst, commit `ea27fd23c962addbd24d8586c5aaf8afe60db0db` перечисляет VeRi, VERI-Wild, CompCars и VMMRdb. В папке автора находится тот же ONNX с ID, указанным в `original_source` карточки OMZ, и конфигурация обучения: `sources=[['veri','veriwild'],['universemodels']]`. Конфигурация имеет 2019 байт, SHA-256 `d939a7b35268e41420b4520b6763bf437bc417e971d8560bf1a847c31b2b231a`. Имена и версия экспортированного графа проверяются по самому ONNX; эта конфигурация не является полным журналом экспорта.

Датасет	Как использован	Версия, доступ и условия
Набор организатора LCT 2026, задача №7	Подбор, калибровка, абляции; вход дообучения combined_v1 (train-часть)	Именованной версии нет; идентификация по SHA-256 ниже. Доступ через кабинет организатора. Отдельная лицензия/прямой URL архива не приложены
RoundaboutHD	Дообучение combined_v1: 511 ID / 6 068 кадров ReID-сабсета	MIT по карточке репозитория данных Bath (с оговоркой о формулировке — datasets-open/REPORT.md §6); полный локальный архив проверен 21.09: 8 075 314 494 байта, SHA-256 `fad611b70065bb67dbd7eb6a1c03280b8c1252fe46c291537ccd0b9230f1dd54`; точный отбор кадров и связь с combined — training/src/README.md
CARLA-ReID	Дообучение combined_v1: 605 ID / 7 260 кадров	Apache-2.0 на репозиторий (подтверждена GitHub API); данные лежат на Dropbox без отдельного файла лицензии — та же оговорка; локальный архив: 2 529 020 889 байт, SHA-256 `55153a24d2c6c95eb88d9eb44b93b03f7cddc05f28dbc58e03c4c4a5e913908d`; рецепт и отбор — training/src/README.md
VeRi-776	Предобучение OSNet и сравнивавшегося FastReID	Семейство VeRi-776; точный архив автора весов, размер и SHA-256 **неизвестны**. Официальный доступ по запросу; некоммерческое использование

Датасет	Как использован	Версия, доступ и условия
VERI-Wild	Предобучение OSNet	В upstream не зафиксировано, какой выпуск/ревизия архива; нельзя автоматически подставлять 2.0. Размер и SHA-256 неизвестны. Доступ по запросу, некоммерческое использование
CompCars	Предобучение OSNet, объединение с VMMRdb	Датасет 2015 года; точные части и ревизия набора автора весов неизвестны. Инструкция загрузки. Только некоммерческие исследования, ограничено дальнейшее распространение; размер/SHA-256 использованных архивов неизвестны
VMMRdb	Предобучение OSNet, объединение с CompCars	Датасет из публикации 2017 года; полный набор или подмножество автора весов не указаны. Архив автора. В репозитории MIT, отдельное подтверждение условий для всех изображений отсутствует; размер/SHA-256 использованного архива неизвестны
ImageNet	FastReID Model Zoo указывает ImageNet-предобучение backbone	Точная ревизия и исходный checkpoint не указаны в переданных материалах. Изображения нами не скачивались. OSNet YAML также включает `pretrained=True`, но происхождение его исходной инициализации полностью не установлено

Изображения RoundaboutHD и CARLA-ReID нашей командой скачаны и использованы для дообучения combined_v1 (состав и хэши собранного набора — training/combined/journal.md, datasets-open/REPORT.md §5); VeRi, VERI-Wild, CompCars, VMMRdb — только готовые веса, сами изображения нами не скачивались. Публичный checkpoint с MIT/Apache-лицензией не заполняет отсутствующие сведения о версиях обучающих данных. Полная цепочка их происхождения остаётся документированным пробелом по разделам 7 и 12 ТЗ; нельзя писать, что все использованные данные безусловно открыты и воспроизводимы.

Отпечатки фактически выданного набора

```
train.csv         9556 строк  536677 байт
bd1df45b052ae9aabb7fd356898e244875f8cad2f111a3f76977c1b90bce9268
test_query.csv    1110 строк   54288 байт
97e1ed21942bae9c95b1ce2e5d339d9f635bf49fa484367e6b19349789bb9b4c
test_gallery.csv   750 строк   36693 байт
a64ed21fa39bbfd8c7a172468d043415800500b6ed695cdb8162188cf9005496
```

Изображения: **11 416 файлов, 7 503 286 826 байт**. SHA-256 текстового манифеста по всем изображениям — `8274bf0d792e3b4fee1009f6fae7649466c296cffcb359c670150a587473cc4a`. Чтобы получить тот же манифест независимо от локали, выполните из `DATA_DIR`:

```
python3 - <<'PY'
import hashlib
from pathlib import Path
```

```
manifest = hashlib.sha256(); count = total = 0
for p in sorted(Path('images').iterdir()):
    if not p.is_file():
        continue
    with p.open('rb') as f:
        digest = hashlib.file_digest(f, 'sha256').hexdigest()
    manifest.update(f'{digest}  images/{p.name}\n'.encode())
    count += 1; total += p.stat().st_size
print(count, total, manifest.hexdigest())
PY
```

Эта дополнительная команда требует Python 3.11+ и читает все изображения. Сам `data/README.md` имеет SHA-256 `b517186daf5186f8ace0b3d2feb9fbce328cd09e620f22b84ff9a2a292d4575c`.

Внешний код и статические ресурсы

Источник	Закреплённая версия / лицензия / проверка
FastReID для сравнения модели	commit `c9bc3ceb2f7a6438b62fb515ea3df6d1e999e95d`, Apache-2.0; внешний клон не включён в git решения
FastReID / Torchreid для сверки метрик	commits `7ed6240e2cb5e56e5ccd61744a3045f24ea7e62d` / `6b8fe56638d81c36b7872e8e41efe18232335f2f`; Apache-2.0 / MIT. Manifest с прямыми URL, размерами и полными SHA-256, исходники и LICENSE включены в `eval/sources/`
MATLAB `compute_AP.m`, `evaluation.m` для сверки соглашения	person-re-ranking commit `ca27f37dd88b27ee1f9dbc5668a9d0f45989ca71`; те же URL/хеши в manifest. Отдельная лицензия этих файлов не установлена
Алгоритм k-reciprocal	Zhong et al., CVPR 2017; порт в проекте, NumPy float64. Проверяемый Python-исходник, commit `278f2c704e7f033b767f4158dff4febc03a0b78a`, 4250 байт, SHA-256 `04bf1b5886ef06275d88be4d68d8f8cfd3ebc892e026f41e848ce294ab192b8e`. У исходного репозитория нет корневого LICENSE; конкретная ревизия первоначального портирования не зафиксирована. Лицензия FastReID автоматически к этому файлу не приписывается
Swagger UI	**5.29.5**, Apache-2.0, релиз, LICENSE. Локальные файлы — `service/app/static/vendor/`; отдельный LICENSE рядом с ними не приложен
Qdrant	**v1.15.5**, исходники, Apache-2.0; Docker-образ `docker.io/qdrant/qdrant:v1.15.5`

Отпечатки Swagger: `swagger-ui-bundle.js` — 1 510 312 байт, `a646692ba5c95a74f99bb2c15ac879dec9a0001a72aed133ad65068da9e90c97`; `swagger-ui.css` — 155 212 байт, `bc5e8d5c013477cf1f35e2fb8ba1dff66be0f72f24e669a509635657145e1acb`; `favicon.png` — 628 байт, `3ed612f41e050ca5e7000cad6f1cbe7e7da39f65fca99c02e99e6591056e5837`. Проверка: `sha256sum 04-solution/service/app/static/vendor/*`.

10. Полный список библиотек и версии среды

Сдаваемый контейнер

Python **3.13.15**, Linux x86_64; корневая исследовательская `.venv` — Python **3.13.13**. Исходный образ сборки: `python:3.13-slim`, проверенный image ID `51cce855bb6e44a8ff6ed0f46ded8850f246bfc7549801459f83fc34b80c210f`, registry digest `sha256:9d2e5553305c7c7b0097999bb17187c69b921ccd6bc9d40e4bb5ebe652c00285`. Проверенный Qdrant digest: `sha256:0fb8897412abc81d1c0430a899b9a81eb8328aa634e7242d1bc804c1fe8fe863`. Dockerfile/Compose сейчас используют теги, поэтому эти digest описывают проверенную среду, но не принуждают будущую сборку к ней.

Прямые зависимости: ONNX Runtime 1.30.0 (MIT), NumPy 2.5.3 (BSD и лицензии включённых компонентов), Pillow 12.3.0 (MIT-CMU), FastAPI 0.141.1 (MIT), Uvicorn 0.53.0 (BSD-3-Clause), python-multipart 0.0.32 (Apache-2.0), qdrant-client 1.19.0 (Apache-2.0). Версии из requirements.txt установлены и проверены `pip check`.

Полный результат `pip freeze --all`, включая транзитивные зависимости и установщик; те же 31 строка лежат в репозитории как requirements-lock.txt и подставляются в сборку constraints-файлом:

```
annotated-doc==0.0.5
annotated-types==0.8.0
anyio==4.15.1
certifi==2026.7.22
click==8.5.0
fastapi==0.141.1
flatbuffers==25.12.19
grpcio==1.84.0
h11==0.16.0
h2==4.4.1
hpack==4.2.0
httpcore==1.0.9
httpx==0.28.1
hyperframe==6.1.0
idna==3.19
numpy==2.5.3
onnxruntime==1.30.0
packaging==26.3
pillow==12.3.0
pip==26.2.1
portalocker==3.2.0
protobuf==7.36.1
pydantic==2.13.5
pydantic_core==2.46.5
python-multipart==0.0.32
qdrant-client==1.19.0
starlette==1.6.0
typing-inspection==0.4.4
typing_extensions==4.16.0
urllib3==2.7.0
uvicorn==0.53.0
```

Этот список фиксирует фактическую проверенную сборку и подключён к Dockerfile как constraints — в обеих ветвях: офлайн `pip install --no-cache-dir --no-index --find-links=/wheels -r requirements.txt -c requirements-lock.txt` и сетевой `pip install --no-cache-dir -r requirements.txt -c requirements-lock.txt`. Проверено пересборкой (офлайн, `--no-cache --pull=never --network none`) — `pip freeze --all` нового образа совпал с файлом построчно, `pip check` без замечаний. Полным lock-файлом с хешами wheel он при этом не является. Все имена пакетов соответствуют их публикациям на PyPI, например onnxruntime 1.30.0, NumPy 2.5.3, qdrant-client 1.19.0. Хеши wheel-файлов в текущем файле

зависимостей отсутствуют. Системные библиотеки фиксируются сохранением образа; отдельный список Debian-пакетов можно получить `docker run --rm --network none vehicle-reid-service dpkg-query -W`.

Фактическое обучение combined_v1

Обучение шло **19.09 на Windows**, `D:\lct-reid\py312\python.exe`, Quadro M2000M. Исторический журнал подтверждает torch **2.5.1+cu118**, Python 3.12. Проверка этой сохранившейся среды 21.09: **Python 3.12.8**, torch **2.5.1+cu118**, CUDA runtime **11.8**, cuDNN **90100**, драйвер **538.18**, NumPy **2.1.3**, Pillow **11.0.0**, ONNX **1.17.0**. Полный freeze всех установленных пакетов и результат повторного экспорта — windows-environment.json. Это текущий снимок сохранённой среды, **не записанный в момент старта lock**; историческую неизменность всех транзитивных версий подтвердить нельзя.

Seed **20260916**, torch seed по этапам `seed + stage`, dev/sampler отдельно описаны в training/src/README.md. Trainer включает `cudnn.benchmark=True`: побитовое повторение независимого обучения не гарантируется. Конфиги этапов 1/2, полные журналы и исходники с проверенными SHA-256 — там же; этап 3 по воротам не запускался. Повторный экспорт сохранённого `stage2.pt` совпал с поставляемым ONNX по SHA-256. Whitening считался отдельно на CPU `worker-vm`, среда job_45 сейчас: Python **3.14.3**, NumPy **2.4.6**, ONNX Runtime **1.30.0**, Pillow **12.3.0**; полный снимок. Среда FastReID с torch 2.14.0+cpu ниже к обучению combined_v1 не относится.

Исследовательские окружения

Это окружение **не нужно** ни для сборки образа, ни для запуска сервиса, ни для команд R/T/V: они целиком живут в контейнере. Оно нужно только тем, кто хочет открыть исследовательские скрипты `04-solution/postproc/`, `04-solution/plate-ablation/`, `04-solution/eval/` вне контейнера. Само по себе оно их не «чинит»: промежуточные `.npy` и каталоги `work/` в git не входят (раздел 11), так что скрипты всё равно упрутся в недостающие входы — часть понятным сообщением, часть трейсбеком (см. раздел 8). Окружение убирает только одну помеху — `ModuleNotFoundError: No module named 'onnxruntime'`.

Корневая `.venv` в git не входит; создаётся с нуля так (нужна сеть):

```
python3 -m venv "$REPO/.venv"
"$REPO/.venv/bin/pip" install onnxruntime==1.30.0 numpy==2.5.3 pillow==12.3.0 onnx==1.22.0
```

Команда закрепляет четыре прямые зависимости, но не транзитивные. Список ниже — исторический снимок, а не обещание точного результата новой установки: в проверке 23.09 resolver выбрал `protobuf==7.36.2` вместо `7.36.1`. `pip freeze --all` показывает фактически установленное окружение; версия `pip` зависит от используемого Python/venv (в сохранённой среде — 26.2.1). Дальше скрипты запускаются своим интерпретатором, например `"$REPO/.venv/bin/python" 04-solution/plate-ablation/scripts/eval_variants.py`.

Сохранённый freeze корневой `.venv` (инспекция графа, baseline, метрики, постобработка, абляции; последующее добавление FAISS отражено ниже):

```
flatbuffers==25.12.19
ml_dtypes==0.6.0
numpy==2.5.3
onnx==1.22.0
onnxruntime==1.30.0
packaging==26.3
pillow==12.3.0
pip==26.2.1
protobuf==7.36.1
typing_extensions==4.16.0
```

Зависимость `onnx==1.22.0` нужна для инспекции ONNX и не включена в runtime. Файл eval/requirements.txt задаёт диапазон `numpy>=1.24,<3`, а не точную версию; опубликованные численные сравнения проверены с NumPy 2.5.3.

Полный freeze отдельного окружения сравнения FastReID:

```
filelock==3.32.3
fsspec==2026.7.0
Jinja2==3.1.6
MarkupSafe==3.0.3
mpmath==1.3.0
networkx==3.6.1
numpy==2.5.3
pillow==12.3.0
pip==24.3.1
PyYAML==6.0.3
setuptools==78.1.0
sympy==1.14.0
tabulate==0.10.0
termcolor==3.3.0
torch==2.14.0+cpu
typing_extensions==4.16.0
yacs==0.1.8
```

Построение исследовательских графиков также использует **matplotlib**, отсутствующий в корневой `.venv` и runtime. При аудите в системном Python обнаружены: Matplotlib 3.10.9, NumPy 2.2.6, Pillow 12.3.0, contourpy 1.3.3, cycler 0.11.0, fonttools 4.61.0, kiwisolver 1.5.0, packaging 24.2, pyparsing 3.1.2, python-dateutil 2.8.2, six 1.17.0. Это снимок доступной вспомогательной среды; единый lock исходного построения всех рисунков не сохранён.

Обход Python-импортов и файлов зависимостей проведён по всем каталогам `04-solution/`, включая review и проверочные исходники. Неиспользованные модели/датасеты из обзорных списков `02-research/` не являются зависимостями результата. combined_v1 действительно обучался на GPU, а не является заготовкой. Его фактические исходники теперь в `training/src/`; `.worker-payload/` содержит также установочные материалы этой среды и не входит в поставку.

Дополнительные зависимости исследований и проверок

Повторная сверка импортов и сохранённых окружений 22.09 выявила зависимости, которые не входили в приведённый выше runtime и ранее не были раскрыты здесь:

Использование	Библиотека / инструмент и версия	Подтверждение и границы
Измеренный ANN-бенчмарк	**faiss-cpu 1.15.0**, NumPy 2.5.3, packaging 26.3; Python 3.13.15 на worker	scaling/REPORT.md, поле `environment.packages` в прогоне на 10^6 векторов. Это исследовательский индекс, HTTP API остаётся на Qdrant
Съёмка UI через Chrome CDP	**websocket-client — историческая версия неизвестна**; версия браузера также не записана	Импорт `websocket` в ui-checks/shoot.py; lock этого прогона в репозитории не найден. Текущая установленная версия не доказывает историческую
Ранние PyTorch-эксперименты	torchvision **0.20.1+cu118**, torch **2.5.1+cu118**	Импорты в training/attempt-2/reid_train.py; точные wheel-версии и SHA — снимок Windows. Это снимок от 21.09, а не lock всех прежних прогонов

Использование	Библиотека / инструмент и версия	Подтверждение и границы
Вспомогательные облачные smoke-скрипты	open_clip_torch **3.3.0**, timm **1.0.29**, transformers **5.17.0** известны только из Windows-снимка; версии этих пакетов в облачном прогоне **неизвестны**	Импорты в smoke.py. Наличие в Windows не подтверждает установку тех же версий в облаке и использование для d1_j48
Облачная заготовка окружения и Blender smoke	onnxruntime-gpu **1.26.0**, torch **2.14.0+cu126**, Blender / `bpy` **4.5.14 LTS** — версии из рецепта	cloud/REPORT.md, blender-smoke.py. Полный freeze GPU-окружения отсутствует; это не среда, в которой получены сдаваемые векторы

Полные сохранённые списки Windows (46 пакетов) и CPU-postprocess (7 пакетов) даны в полях `pip_freeze_all` файлов windows-environment.json и postprocess-environment.json. Они включают транзитивные зависимости, установщик и вспомогательные пакеты; их наличие в среде не означает вызов каждого пакета при обучении. Заимствованные исследовательские рецепты в `02-research/training-v2/recipe-details/` также импортируют NVIDIA Apex; подтверждения их исполнения и версии Apex для сдаваемого результата нет. Импорт в сохранённом чужом рецепте не приписывается рабочему trainer.

Сверка 22.09: **31/31** строка runtime в этом разделе совпала с `requirements-lock.txt`; metadata **30/30** поставляемых wheels совпали по именам и версиям с lock, `pip` приходит из базового образа. Обнаруженные внешние зависимости перечислены, но полного исторического lock исследовательских сред и списка версий всех системных библиотек в Git нет. Поэтому требование §12 о полном перечне с версиями выполнено для Python runtime, а для всего жизненного цикла решения остаётся частичным.

11. Известные ограничения и недостающие материалы

- **Обучение combined_v1 раскрыто:** фактические исходники, конфиги, команды, SHA-256 и ограничения provenance. Полное переобучение в исправлении не выполнялось; отсутствие исторического source-lock и полного freeze на дату запуска не заменено предположениями.
- **Полная исследовательская воспроизводимость пока не достигнута.** Часть скриптов содержит `/home/artem/projects/...` и ссылки на временные `work/`, `metadata/`, `evaluator_snapshot/`, `.ids`, `.npy`, которых нет в git. Команда R обходит эти зависимости для основных метрик, но не восстанавливает все старые эксперименты.
- **Абляция пластины — не доказательство полного отсутствия использования ГРЗ, но проверена для обеих сдаваемых моделей.** Для combined_v1 маскирование зоны номера статистически неотличимо от контрольной зоны той же площади (все контрасты накрывают 0, training/combined/REPORT.md §5); для исходного OSNet — plate-ablation/. Исходной ручной разметки/детекций/эмбеддингов вариантов в git нет — числа берутся из сохранённых json. Верхнюю границу `+0.016 mAP` нельзя называть «меньше процента метрики».
- **Не полностью известны версии внешних обучающих архивов и часть условий их распространения.** RoundaboutHD и CARLA-ReID теперь идентифицированы хешами полных локальных архивов и точным отбором в training/src/; исторический журнал содержал только хеши производных combined-файлов. Ревизия HuggingFace/неизменяемый удалённый URL не были записаны; скачанный ныне архив надо сверять с опубликованным SHA-256; цепочка происхождения предобучения OSNet (VeRi, VERI-Wild, CompCars, VMMRdb) остаётся документированным пробелом. Все обнаруженные источники раскрыты в §9; отсутствующие сведения названы отсутствующими.
- **Веса, Python wheels и базовые образы для офлайн-сборки поставлены.** Данные организатора предоставляются отдельно; Docker-путь проверен на Engine 29.7.2 / Compose

2.40.3. Пакеты ставятся из `wheels/`, PyPI на сборке не нужен, базовые образы лежат в offline/ отдельными архивами: `python-3.13-slim.tar.gz` 45,6 МБ и `qdrant-v1.15.5.tar.gz` 65,4 МБ. Оба проверены манифестом и загрузкой в отдельное хранилище. Единый архив от 21.09 содержал только Python с двумя тегами и заменён 22.09; раздельные архивы исключают саму ошибку и укладываются в лимит GitHub в 100 МБ на файл. Docker и Podman проверены независимо (отчёт Docker); наличие исходных данных остаётся условием инференса.

12. Что подготовить к передаче жюри

По сообщению организаторов, переданному 27.09, платформа уже принимает ссылки на исходники и требует заполнить все обязательные поля. Ссылки следует подать заранее; улучшения за ними допускаются **строго до 29.09.2026, 23:59 МСК**. Любые зафиксированные после этого срока изменения приводят к аннулированию работы. Ожидаемые ответы на вопросы кейса не отменяют этот срок.

Вход в документацию — корневой Readme.md, с буквальным именем из ТЗ и прямыми ссылками на разделы этого файла. Содержимое `SOLUTION.md` не копируется во второй документ. В комплект также входят три test-артефакта, три файла весов с SHA-256, Dockerfile/Compose, колёса `service/wheels/` и два базовых образа `service/offline/`. Оба ONNX и whitening поставляются в Git; данные организатора передаются отдельно (§7).

Требование ТЗ §13	**Фактическое состояние на 28.09.2026**
Ссылка на репозиторий	Репозиторий команды в SourceCraft и приватный GitHub обновляются в `main`. SourceCraft содержит курируемый сдаваемый код и manifest сверки с исходным SHA; релизная модель — d1_j48. Анонимное открытие репозиториев вернуло 404; доступ под учётной записью жюри нужно подтвердить
Ссылка на презентацию	Полные PDF на 16 слайдов и PPTX опубликованы с HTTPS и проверены по хешам; обязательные 1–5 сохраняют страницы 7–11 шаблона. Они исключены из Git из-за страницы контактов. Технический PDF без контактов `05-presentation/public/presentation.pdf` остаётся в репозитории. Заполнение поля презентации на платформе не подтверждено
Ссылка на работающий прототип	Лендинг, сервис и запасной путь опубликованы с HTTPS на VPS `84.201.146.191`. Внешняя проверка прошла: реальный поиск, отказ, 750 объектов галереи, демо-кадры, изображения кандидатов и выгрузки. Состав и откат. Отправка ссылки в форме организатора ещё не подтверждена
Документация `.doc` или `.pdf`, если вынесена за пределы Readme.md	Полный текст `SOLUTION.md` поставляется как 06-documentation/SOLUTION.pdf и публичный PDF. PDF собирается штатным генератором; Subject содержит SHA-256 исходника, `--check` проверяет свежесть. Загрузка URL в личный кабинет не подтверждена

Недостающие исходники исследовательских протоколов, лицензии и provenance нужны до заявления о полном воспроизведении **всех** результатов. Публикация сайта не подтверждает отправку ссылок через форму организатора.

13. Локальный кандидат 25–26.09: сервис, эксперимент, развёртывание

Демонстрация через настоящий API

Дополнительно к основному `docker-compose.yml` сервис предлагает `GET /api/demo` с двумя разрешёнными query-кадрами и bbox из `test_query.csv`: один пример поиска и отдельный пример отказа. Кадр скачивается как оригинальные байты; клиент вызывает обычный `/api/search`. Ответ не записан заранее. Если локальный набор отсутствует, список демонстрации пуст. `GET /api/materials` возвращает только реально установленные по allowlist документы и артефакты с размером и SHA-256; `GET /materials/{id}` скачивает их. Полный deck с контактами в этом каталоге не размещается. После загрузки другого кадра/bbox либо нового поиска старый ответ, разбор и выгрузка не должны относиться к новому запросу. После визуальной проверки прежний запрос с почти совпадающим кадром галереи заменён. В новом примере первые кандидаты сняты с других точек, рамка охватывает машину целиком; проверка и хеши. У закрытого test нет меток личности и камер: этот экран показывает реальное ранжирование, а не доказанный межкамерный успех.

Интерфейс различает отказ (успешный ответ с пустым списком), ошибку файла, 409 без gallery и 503 при недоступном хранилище. На первом экране есть реальный демо-путь, крупные кадры, bbox мышью/касанием/точными числами, парное сравнение, честная шкала косинусной близости и экспорт текущего результата. Объяснение относится к первой OSNet; сходство поиска — к ансамблю с whitening, поэтому цифры не выдаются за одну и ту же оценку. OCR не выполняется, но область номерной пластины остаётся частью изображения; абляция в §11 ограничена.

Отдельный ML-эксперимент и решение о релизе

Исходный `combined_v1` уже является нашей дообученной сетью. Новый эксперимент distill-20260925 добавил обученную residual-голову к замороженному combined_v1: вход/выход 512, скрытый слой 256, distillation нормированных векторов и межкамерных отношений. Один seed, 30 эпох основного прогона и одна абляция 30 эпох. Train/dev разделены по vehicle_id (1071/100); исторический dev уже использовался при отборе backbone, а подтверждающая val многократно применялась в прошлых экспериментах. Это существенное ограничение силы вывода, не новый нетронутый holdout.

На одном протоколе val 1110 query × 750 gallery, 832 с парой, market camera исключение: d1_j48 cosine/KR full-ranking mAP **0,731609/0,774092**; student **0,707940/0,745935**; fusion с весом student=0,25 **0,731071/0,766010**. Отдельный CPU batch-1 p95 с двумя потоками, 40 попарно чередуемых примеров: d1_j48 **619,39 мс**, student **452,38 мс**, fusion **791,37 мс**. Student снизил p95 примерно на 27 %, но дельта KR mAP −0,02816, парный CI95 по vehicle_id [−0,04397; −0,01309]; cosine CI также ниже нуля. Ни student, ни fusion не выполняют заранее записанные критерии приёмки. **Сдаваемой остаётся d1_j48**, checkpoint/ONNX и честный отрицательный отчёт сохранены отдельно. Старые пороги не перенесены на student; его отказ калибровался на dev отдельно для cosine/KR. Историческое обучение release whitening использовало dev, что прямо отмечено в отчёте.

Самостоятельное ядро «из кадра в признак»

Residual student выше обучался на готовом признаке и не является самостоятельным image encoder. По отдельному заранее зафиксированному протоколу реализован и обучен CrossViewCore: один MobileNetV3-Small с публичной ImageNet-инициализацией, нашими global/local ветвями 256+128+128, межкамерным sampling/loss и полным fine-tune backbone. ONNX принимает RGB-кроп 208×208 и выдаёт нормированный вектор 512 без вызова OSNet или whitening. Один CPU-прогон на данных организатора: 6639 fit-кадров, 609 dev-кадров, 14 эпох × 600 батчей, 43 минуты, checkpoint SHA `22fc2fdd92519ae21f08eb6feef3812dc92bb5dd0191b2f6c7f1bbdc6a966e1a`. GPU worker не смог начать реальный батч из-за нехватки RAM хоста; его чужие процессы не завершались. Обученный ONNX — 4,90 МБ. Все метрики и входные хеши сохранены в результатах.

На том же повторно использованном val mAP cosine/KR: отдельное ядро **0,276749/0,311518**, исследовательская fusion 0,25 **0,725141/0,755009**, сдаваемая d1_j48 **0,731609/0,774092**. Для core val TNR отказа равен **0** при выбранных только на dev порогах; высокий presence F1 этого не исправляет. Парный bootstrap по vehicle_id: KR ΔmAP core −0,462573, CI95 [−0,509323; −0,415570]; fusion −0,019082, CI95 [−0,035499; −0,003246]. Обученная core на одинаковых входах/CPU быстрее по batch-1 p95: **2,19 мс против 71,64 мс** (≈32,7×), но fusion даёт 93,17 мс. Эта граница замера — готовый тензор → признак, без JPEG/API/Qdrant/KR; её нельзя сопоставлять напрямую с указанным выше бенчмарком student на другом узле. Torch↔ONNX проверены на всех 1860 кадрах: top-1 и отказ совпали на 1110 запросах для cosine/KR. Независимая проверка подтвердила отрицательное решение. **Ни core, ни fusion не проходят заранее заданные критерии; релиз остаётся d1_j48.** Этот результат не доказывает невозможности другого компактного ядра.

После этого ядро **действительно дообучено**, а не только спроектировано: с сохранённого checkpoint выполнены 12 эпох на 6639 собственных fit-кадрах с дистилляцией дескриптора и межкамерных сходств от релизного ансамбля. Затем заранее объявленное второе продолжение прошло 8 эпох на 19 967 fit-кадрах. Учитель использовался только при обучении одиночного image encoder; новый ONNX самостоятельно строит 512D признак из RGB-кропа. Обе попытки были заморожены до новой оценки val, веса fusion и пороги выбраны только по dev. Протоколы, SHA, таблица cosine/KR, отказ, скорость и независимая проверка — в отчёте продолжения.

На той же **повторно использованной** val одиночное ядро после own/combined продолжений достигло KR mAP **0,323997/0,327450** против **0,774092** у `d1_j48`; combined fusion с релизом — **0,762604**, парная Δ mAP −0,011487 с CI95 [−0,026954; +0,003198]. TNR отказа обоих одиночных вариантов равен нулю. На одинаковом CPU в парном тесте готового тензора combined core имело batch-1 p95 **1,727 мс** против **43,861 мс** у релиза; это не полная задержка HTTP. Холм по восьми заявленным сравнениям и проверка Torch↔ONNX выполнены. Значимого выигрыша fusion и нужного качества одиночного ядра не получено; **сдаваемая d1_j48 не меняется**. Checkpoint, ONNX и fit-цели опыта хранятся локально вне комплекта сдачи.

Репетиция и VPS

Развёртывание фиксирует VPS, внешний HTTPS-маршрут, измеренные задержки, офлайн-кандидат, seed gallery, healthcheck, Compose и откат. Проверенный bundle `fcb038a` содержит d1_j48; текущий API-образ `6e8c1c9` меняет только демонстрационные входы, а лендинг и документы размещены отдельно. Сервис проверен извне; ссылки на SourceCraft и прототип должны быть отправлены через форму организатора отдельно. Организаторы подтвердили предел правок по ссылке **29.09 23:59 МСК**; коммиты после срока аннулируют работу. Длительность доступности прототипа для жюри остаётся внешним вопросом.