Михаил Шардин
личный блог

19 августа 2025, 04:48

Безопасность активов начинается с подъезда. Как я анализировал видео с камер, чтобы знать всех в лицо

Да, понимаю эта статья не совсем в формате Смартлаба. Но решил поделиться экспериментом, который, на мой взгляд, перекликается с нашей общей темой — безопасностью и технологиями. Обычно мощные компьютеры ассоциируются с бэктестингом стратегий, анализом котировок или, в худшем случае, с компьютерными играми.

Каждый день мимо двери моего подъезда проходят десятки людей. Иногда это знакомые соседи, но чаще — курьеры или случайные гости.

Домофонная камера всё записывает, но вручную пересматривать часы видео бессмысленно. Мне стало интересно: можно ли разово прогнать архив записей через алгоритмы компьютерного зрения и посмотреть, как быстро GPU справится с такой задачей.

Это был чисто экспериментальный проект: не «система слежки», а тест производительности и возможностей CUDA в связке с dlib и face_recognition.

На словах всё выглядело просто, а на деле пришлось пройти целый квест из несовместимых программ, капризных драйверов и упрямой библиотеки распознавания лиц. Но в итоге я собрал рабочее окружение и хочу поделиться опытом — возможно, это поможет тем, кто столкнётся с похожими проблемами.

Проект выложен на GitHub.

Часть 1: Битва за dlib с CUDA-ускорением на Ubuntu

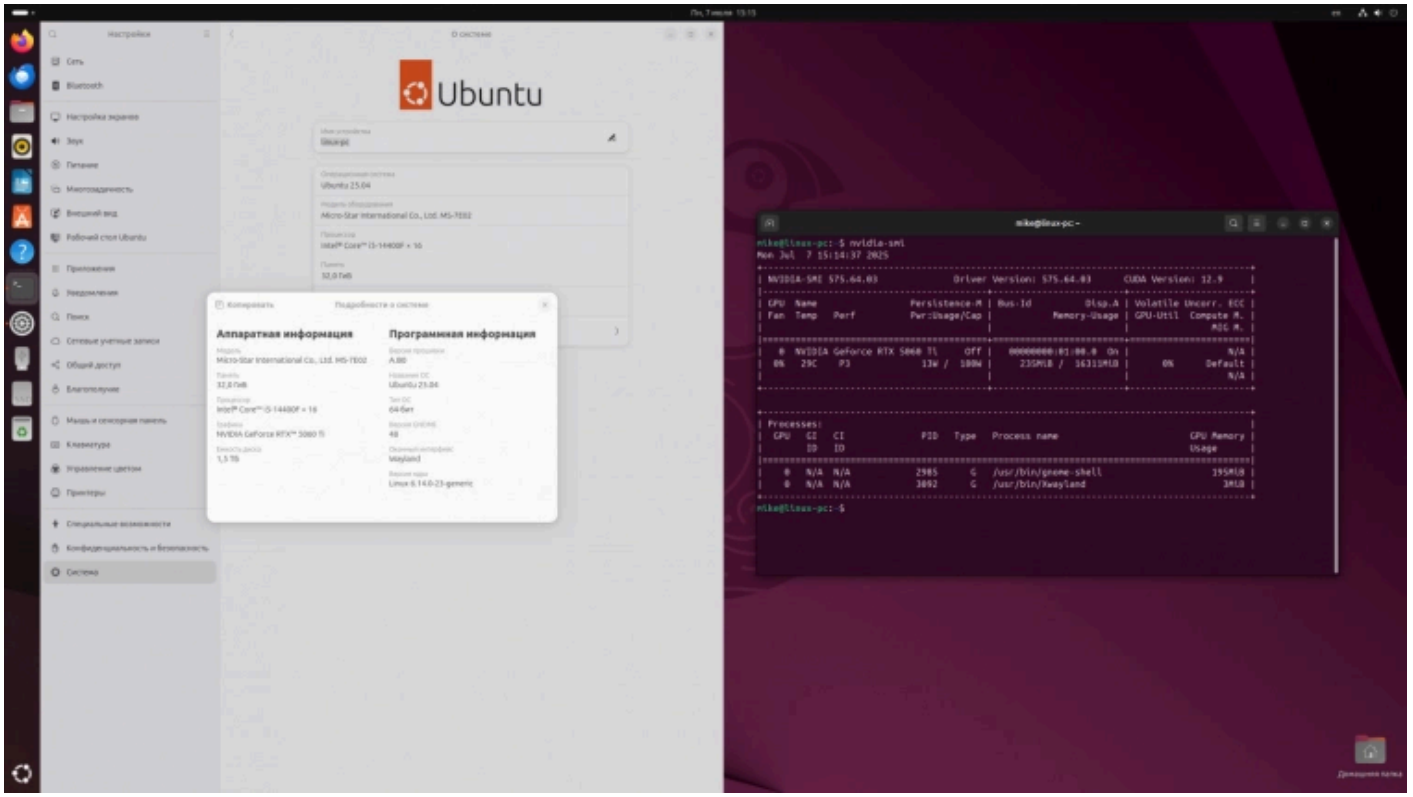
dlib — это популярная библиотека на Python для компьютерного зрения и машинного обучения, особенно известная своим модулем распознавания лиц. Она умеет искать и сравнивать лица. Однако «из коробки» через `pip` она работает только на CPU, что для задач с большим объёмом данных ужасно медленно.

Введите текст комментария

несколькими тысячами кадров на CPU может занять часы, тогда как с GPU — минуты. CUDA задействует сотни параллельных потоков, что особенно эффективно для матричных операций и свёрточных сетей, лежащих в основе `face_recognition`.

Именно поэтому моя цель была не просто «запустить dlib», а сделать это с полной поддержкой GPU.

Эта часть рассказывает о том, как простая, на первый взгляд, задача превратилась в двухдневную борьбу с зависимостями, компиляторами и версиями ПО.



Неподходящая Ubuntu 25.04, моя конфигурация полностью описана здесь

Расписываю по шагам — может быть кто-то найдёт эту статью через поиск и ему пригодится.

1. Исходная точка и первая проблема: неподходящая версия Python

Задача: установить `face_recognition` и его зависимость `dlib` на свежую Ubuntu 25.04.

- Предпринятый шаг: попытка установки в системный Python 3.13.
- Результат: ошибка импорта `face_recognition_models`. Стало ясно, что самые свежие версии Python часто несовместимы с библиотеками для Data Science, которые обновляются медленнее.

2. Вторая проблема: dlib работает, но только на CPU

- Предпринятый шаг: после настройки `pyenv` и установки зависимостей (`numpy`, `opencv-python` и т.д.), `dlib` и `face_recognition` успешно установились через `pip`.
- Результат: скрипт анализа видео работал ужасно медленно (несколько минут на одно видео). Мониторинг через `nvidia-smi` показал 0% загрузки GPU.
- Диагноз: стандартная установка dlib через `pip` скачивает готовый бинарный пакет («wheel»), который собран без поддержки CUDA для максимальной совместимости. Чтобы задействовать GPU, dlib нужно компилировать из исходного кода прямо на моей машине.

3. Третья, главная проблема: конфликт компиляторов CUDA и GCC

- Предпринятый шаг: попытка скомпилировать `dlib` из исходников с флагом `-DDLIB_USE_CUDA=1`.
- Результат: сборка провалилась с ошибкой. Анализ логов показал, что `cmake` находит `CUDA Toolkit 12.6`, но не может скомпилировать тестовый CUDA-проект. Ключевая ошибка: `error: exception specification is incompatible with that of previous function "cospi"`
- Диагноз: мой системный компилятор `GCC 13.3.0` (стандартный для Ubuntu 25.04) был несовместим с `CUDA Toolkit 12.6`. Новые версии GCC вносят изменения, которые ломают сборку с более старыми версиями CUDA.

4. Попытки решения конфликта компиляторов

- Шаг №1: установка совместимого компилятора. Я установил `gcc-12` и `g++-12`, которые гарантированно работают с CUDA 12.x.
- Шаг №2: ручная сборка с указанием компилятора. Я пытался собрать `dlib` вручную, явно указав `cmake` использовать `gcc-12`:

  `<code class="bash">cmake .. -DCMAKE_C_COMPILER=gcc-12 -DCMAKE_CXX_COMPILE`

  Результат: та же ошибка компиляции. `cmake`, несмотря на флаги, по какой-то причине продолжал использовать системные заголовочные файлы, конфликтующие с CUDA.
- Шаг №3: продвинутый обходной маневр (wrapper). Я создал специальный скрипт-обертку `nvcc_wrapper.sh`, который должен был принудительно «подсовывать» `nvcc`

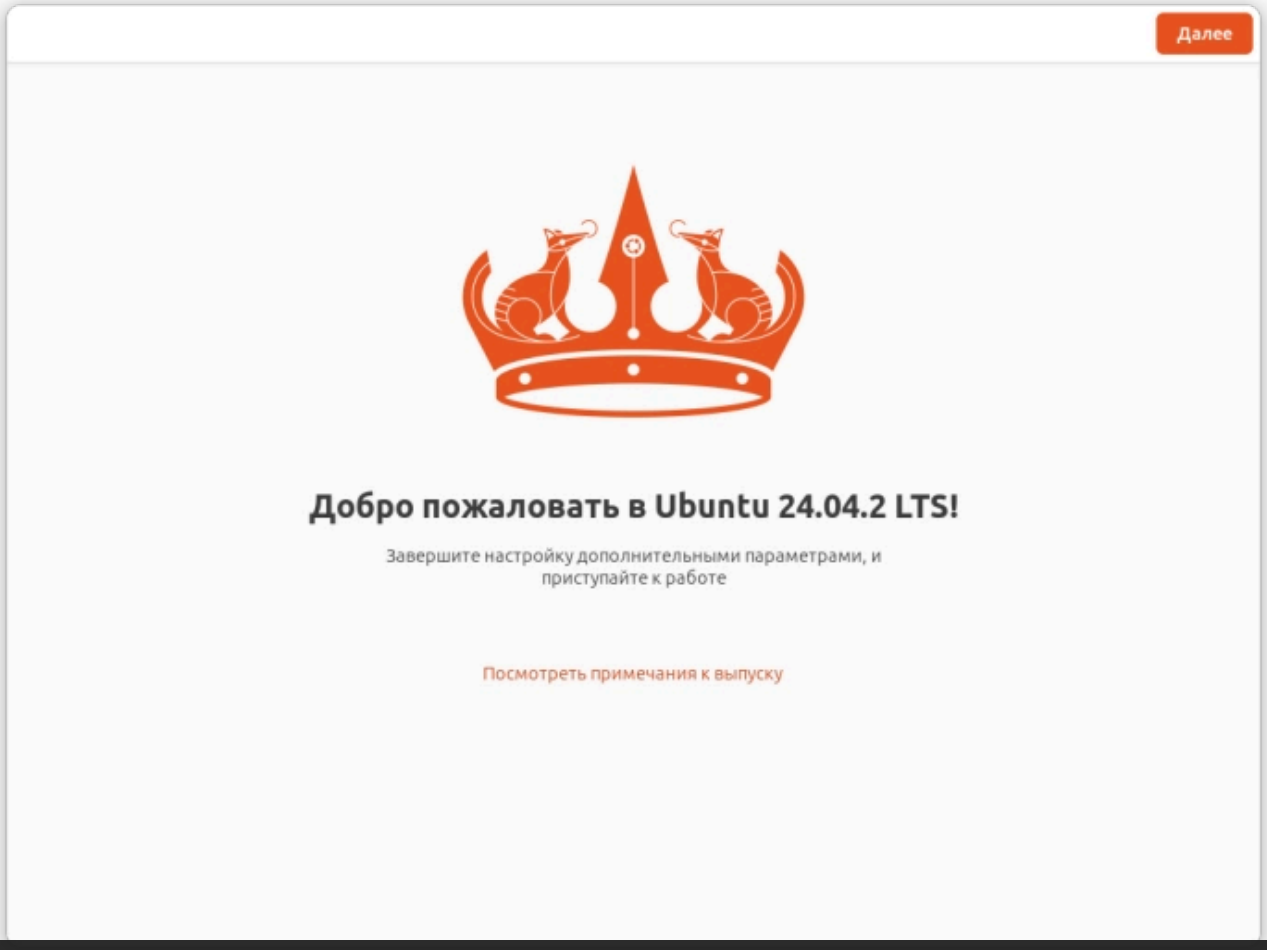
Капитуляция перед реальностью. Несмотря на все предпринятые шаги — использование pyenv, установку совместимого компилятора `GCC-12` и даже создание `wrapper`-скриптов — `dlib` так и не удалось скомпилировать с поддержкой CUDA на Ubuntu 25.04.

Похоже проблема была не в моих действиях, а в самой операционной системе. Использование не-LTS релиза Ubuntu для серьезной разработки с проприетарными драйверами и библиотеками (как CUDA) — это путь, полный боли и страданий.

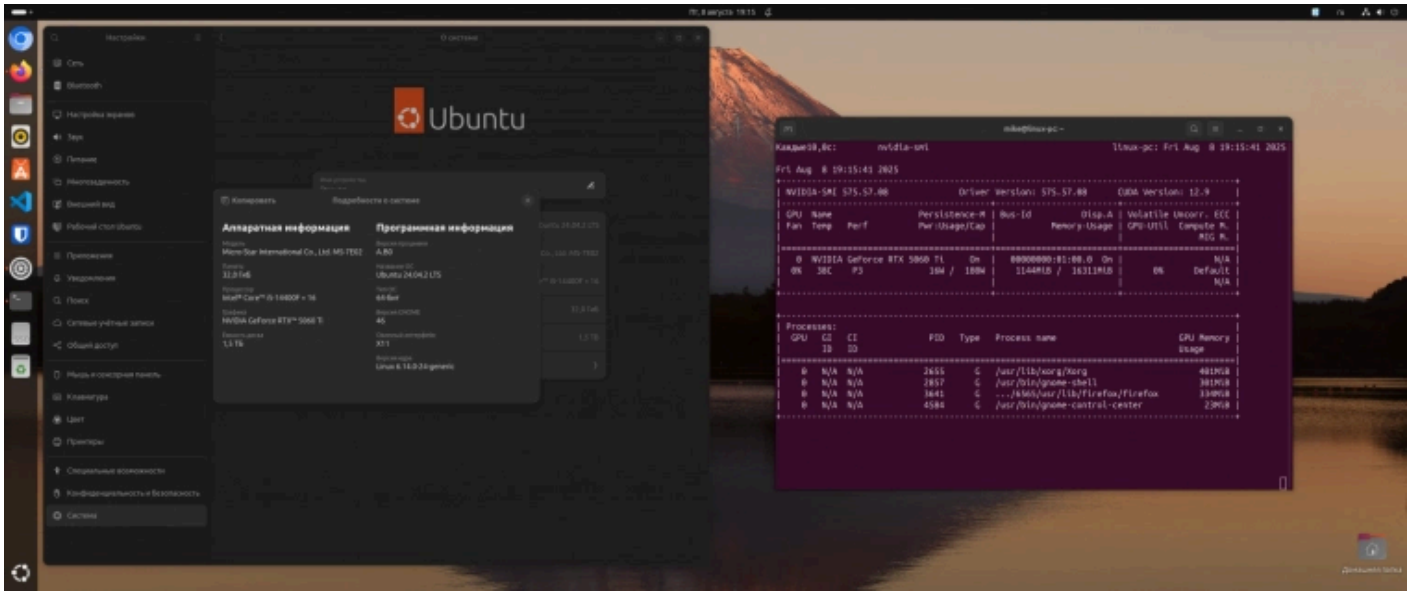
Принял решение установить Ubuntu 24.04 LTS, для которой NVIDIA предоставляет официальную поддержку `CUDA Toolkit 12.9 Update 1`.

Часть 2: чистый лист и работающий рецепт

Установил Ubuntu 24.04 LTS — систему с долгосрочной поддержкой, для которой NVIDIA предоставляет официальный CUDA Toolkit и драйверы. Это был шаг назад, чтобы сделать два вперёд.



ошибок, включая установку CUDA Toolkit и отдельно cuDNN (библиотеки для нейросетей, без которой dlib не видит CUDA), родился финальный, работающий рецепт.



Ubuntu 24.04.2 LTS

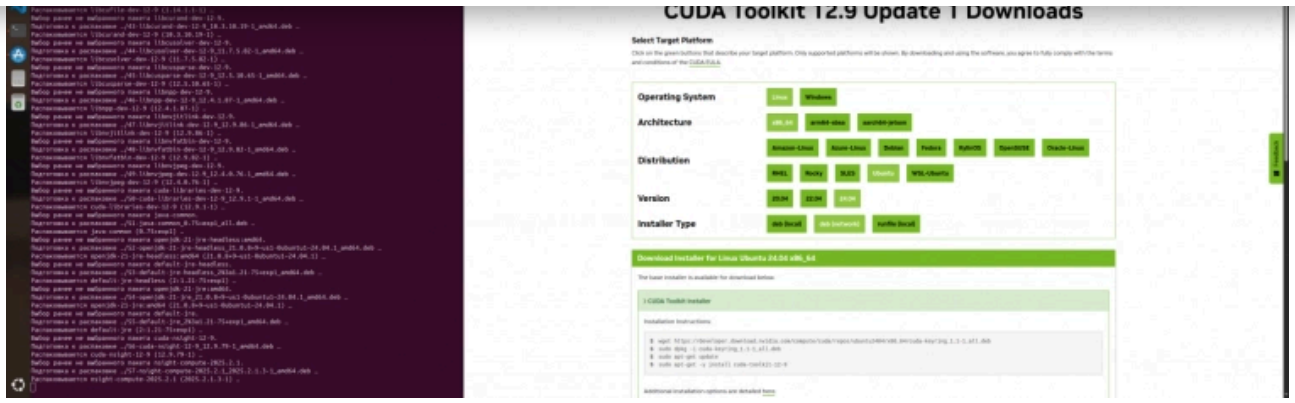
«Золотой» скрипт установки

Вместо того чтобы описывать десятки команд, которые я вводил вручную — собрал все шаги в единый установочный скрипт `setup_env.sh`. Что он делает:

1. Проверка `pyenv`. Скрипт начинается с проверки наличия `pyenv`. Это позволяет использовать нужную версию Python (3.11.9), а не системную, избегая конфликтов.

1. Установка системных библиотек. Для компиляции `dlib` из исходного кода необходимы инструменты сборки (`build-essential`, `cmake`) и библиотеки для работы с математикой и изображениями (`libopenblas-dev`, `libjpeg-dev`). Скрипт автоматически их устанавливает.

   Важно: скрипт предполагает, что CUDA Toolkit и отдельно cuDNN уже установлены по официальным инструкциям NVIDIA для вашей системы — они по ссылкам.



1. Создание чистого `venv`. Создаем изолированное виртуальное окружение, чтобы зависимости нашего проекта не конфликтовали с системными. Скрипт удаляет старое окружение, если оно существует, для гарантированно чистой установки.

1. Ключевой момент: установка `dlib`. Это сердце всего процесса. Команда `pip install dlib` с особыми флагами:

    - `--no-binary :all:` — этот флаг принудительно запрещает `pip` скачивать готовый, заранее скомпилированный пакет (`wheel`). Он заставляет `pip` скачать исходный код `dlib` и начать компиляцию прямо на вашей машине.
    - `--config-settings="cmake.args=-DDLIB_USE_CUDA=1"` — а это инструкция для компилятора `cmake`. Мы передаем ему флаг, который говорит: «При сборке, пожалуйста, включи поддержку CUDA».

Именно эта комбинация заставляет `dlib` собраться с поддержкой GPU на Ubuntu 24.04 LTS чтобы использовать видеокарту, а не в стандартном CPU-only варианте.

Вот сам скрипт `setup_env.sh`:

`<code class="bash">#!/bin/bashset -eVENV_DIR=".venv"PYTHON_VERSION_TARGET="3.11.`

`requirements.txt`:

`<code>numpyopencv-pythongit+https://github.com/ageitgey/face_recognition_modelst`

Часть 3: собираем все вместе

Камера, смотрящая на лифтовой холл. Фото из интернета

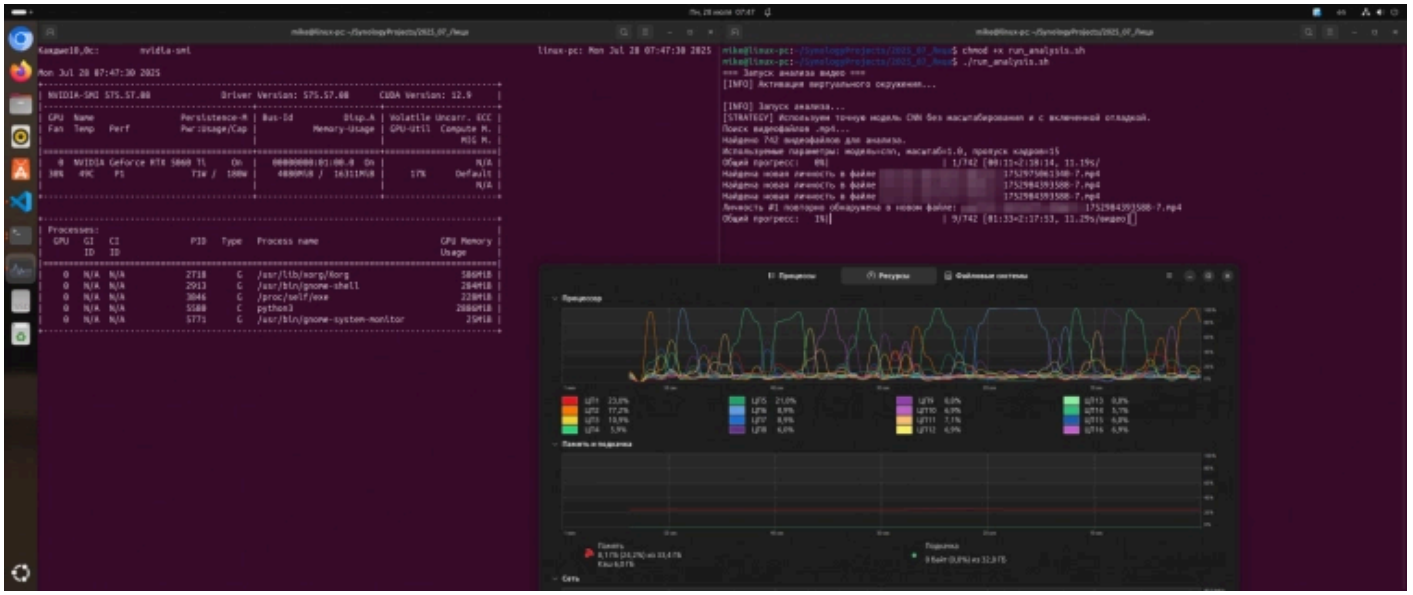
После победы над зависимостями у меня есть полностью рабочее окружение с CUDA-ускорением. Настало время применить его к реальным данным. Мои исходные данные — это архив видеозаписей с двух IP-камер, которые пишут видео на сетевой накопитель Synology Surveillance Station (есть аналоги). Для приватности я заменю реальные имена камер на условные:

- `podiezd_obshiy\` — камера, смотрящая на лифтовой холл.
- `dver_v_podiezd\` — камера из домофона, направленная на улицу.

Внутри каждой папки видео отсортированы по каталогам с датами в формате `ГГГГММДД` с суффиксом `AM` или `PM`. Сами файлы имеют информативные имена, из которых легко извлечь дату и время записи: `podiezd_obshiy-20250817-160150-....mp4`.

Камера из домофона, направленная на улицу. Здесь качество гораздо лучше потому что камера цифровая, а не аналоговая как у меня из квартирного домофона. Это фото из интернета

С данными разобрались, теперь перейдем к инструменту — Python-скрипту `face_report.py`. Скрипт служит разовым инструментом анализа архива видео, чтобы протестировать работу CUDA.



За работой

Общая архитектура скрипта

0.5) ускоряет обработку, но может пропустить мелкие лица.

- `--skip-frames`: количество пропускаемых кадров. Анализировать каждый кадр избыточно и медленно; достаточно проверять каждый 15-й или 25-й.

Скрипт находит все `.mp4` файлы в указанной директории и запускает основной цикл, обрабатывая каждый видеофайл.

1. Детекция лиц: HOG против CNN

`face_recognition` предлагает два алгоритма детекции: HOG (Histogram of Oriented Gradients) и CNN (Convolutional Neural Network). HOG — классический и очень быстрый метод, отлично работающий на CPU. CNN — это современная нейросетевая модель, гораздо более точная (особенно для лиц в профиль или под углом), но крайне требовательная к ресурсам.

Раз я так боролся за CUDA, выбор очевиден — будем использовать `cnn`. Это позволит находить лица максимально качественно, не жертвуя скоростью.

2. Уникализация личностей

Как скрипт понимает, что лицо на двух разных видео принадлежит одному и тому же человеку? Он преобразует каждое найденное лицо в `face_encoding` — вектор из 128 чисел, своего рода уникальный «цифровой отпечаток».

Когда появляется новое лицо, его «отпечаток» сравнивается со всеми ранее сохраненными. Сравнение происходит с определенным допуском (`tolerance`). Установил его равным 0.6 — это золотая середина, которая позволяет не путать разных людей, но и узнавать одного и того же человека при разном освещении или угле съемки.

3. Умный подсчет: один файл — один голос

Простая логика подсчета привела бы к абсурдным результатам: если курьер провел у двери 30 секунд, его лицо могло бы быть засчитано 50 раз в одном видео. Чтобы этого избежать, я ввел простое, но эффективное правило: считать каждое уникальное лицо только один раз за файл.

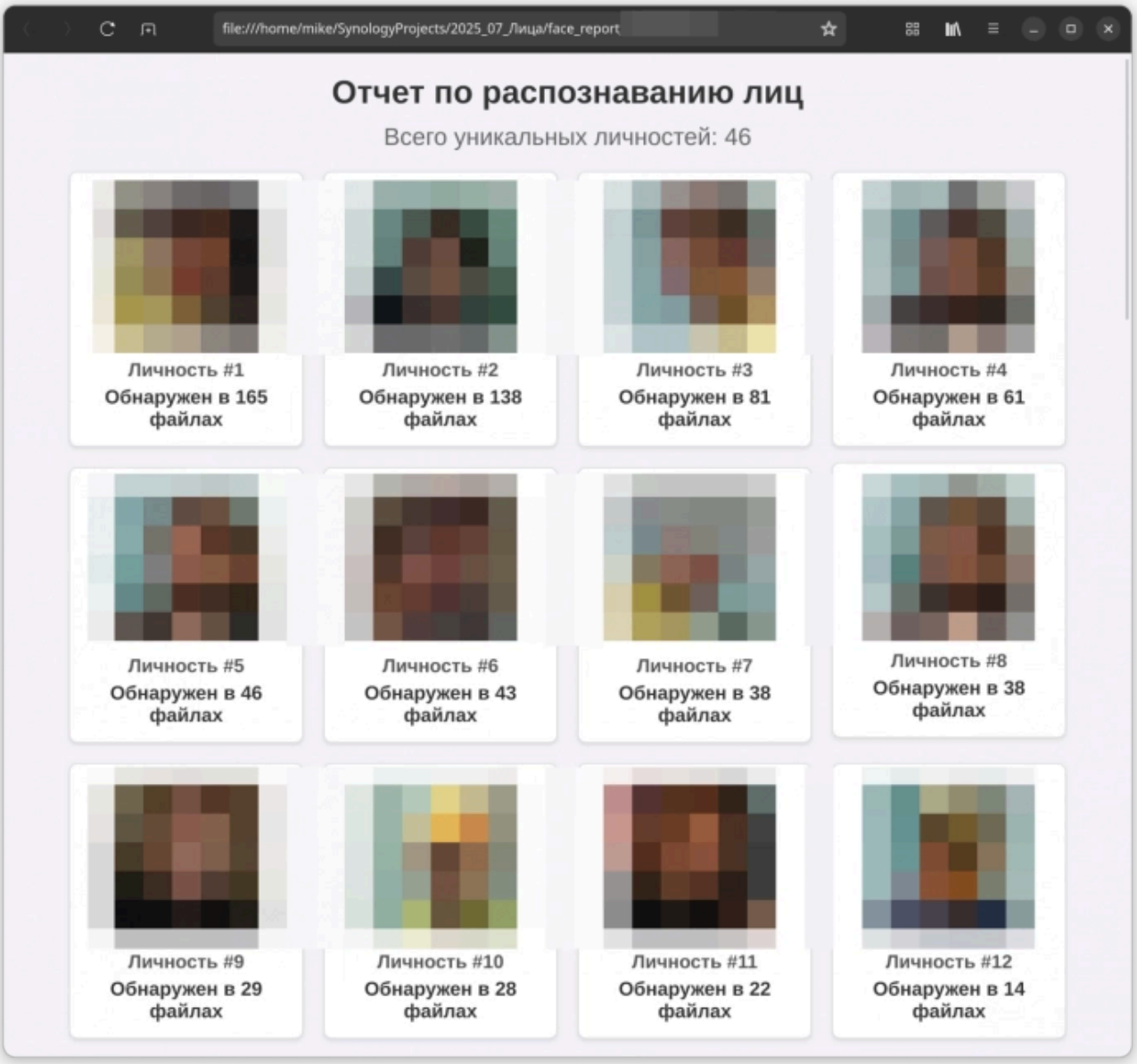
4. Создание красивых иконок

Чтобы в кадр попадала вся голова с прической и частью шеи, я добавил в функцию `create_thumbnail` логику с отступами. Она берет размер найденного лица и увеличивает область кадрирования на 50% по вертикали и горизонтали. Так превью в отчете выглядят гораздо лучше и живее.

уникальные личности в этом эксперименте отсортированы по частоте появлений.

Часть 4: результаты и выводы

Для эксперимента я посчитал уникальных людей в выборке. Скрипт я запускал разово, отдельно для каждой камеры — это не постоянно работающий сервис, а скорее любопытная исследовательская игрушка.



Результаты оказались наглядными, но и показали пределы технологии. Качество

Но и здесь есть нюанс: один и тот же человек в куртке и без неё, в кепке или с распущенными волосами, зачастую определяется как разные личности — видимо надо где-то крутить настройки. Поэтому цифры в отчёте стоит воспринимать не как абсолютную истину, а как любопытную статистику, показывающую общее движение людей, а не точный учёт.

Заключение

От простой идеи — «разово прогнать архив записей через алгоритмы компьютерного зрения и посмотреть, как быстро GPU справится с такой задачей» — я прошёл путь через череду технических ловушек: несовместимые версии Python, упёртый dlib, капризы CUDA и GCC.

К тому же это не сервис, а исследовательская проверка возможностей GPU видеокарты.

Автор: Михаил Шардин
Моя онлайн-визитка
Telegram «Умный Дом Инвестора»

19 августа 2025

Nvidia cuda

Показать 2 ответа

Andy
19 августа 2025, 05:13

На 25 убунту вернулся после сборки?

Показать 1 ответ

Хрен Столовый
19 августа 2025, 05:13

Не ну база лиц это точно статья, сделайте лучше чтоб строилась модель черепа по имеющимся данным, во первых фиг докажут что данные приватные, а во вторых многим посетителям этого сайта такая программа прям необходима

Показать 11 ответов

ПлощадьДНР
19 августа 2025, 07:48

👍
хороший пост!
дальше как использовать библиотеку лиц?
хожу я к примеру к «любовнице» несколько раз в неделю, выгляжу каждый раз по-разному:
одеваю разные худи с капюшоном,
или головной убор с очками,
несу в руках пакет с «гостинцами» которым перекрываю видимость моего лица..
смысл понятен, остаться инкогнита! без морали, ну так были случаи ..
выкидывал из подъезда «писающего соседа».
после сосед меня пытался караулить 🥳
, по причине обиды и нанесения легкого увечья. откуда он вспомнит?
все конечно все эти технологии заняты, вот только мало применимы в данных случаях. для обслуживания и постоянных переналаживаний нужен техник, а не дешевле посадить консьержку. если дом не из эконом конечно.

Показать 1 ответ

Ещё 1 комментарий

ОТПРАВИТЬ

Читайте на SMART-LAB

ЗА ПОЛУГОДИЕ, ЗАКОНЧИВШЕЕСЯ 30 ИЮНЯ 2025 ГОДА (НЕАУДИРОВАННЫЙ)
(в миллионах российских рублей, за исключением прибыли на акцию)

	Приме-чания	За полугодие, закончившееся 30 июня 2025 года	30 июня 2024 года
ВЫРУЧКА	16	171 233	201 884
СЕБЕСТОИМОСТЬ РЕАЛИЗАЦИИ	17	(144 244)	(160 257)
ВАЛОВАЯ ПРИБЫЛЬ		**26 989**	**41 627**
Коммерческие, общехозяйственные и административные расходы	18	(35 722)	(38 177)
Прочие операционные доходы	19	1 383	585
Прочие операционные расходы		(857)	(115)
Изменение справедливой стоимости инвестиционной недвижимости		128	-
ОПЕРАЦИОННЫЙ УБЫТОК /ПРИБЫЛЬ		**(7 559)**	**3 720**
Финансовые доходы	20	505	443
Финансовые расходы	20	(26 632)	(16 527)
УБЫТОК ДО НАЛОГА НА ПРИБЫЛЬ		**(33 686)**	**(12 364)**
Доходы по налогу на прибыль		8 527	2 036

М.Видео - есть ли в мультивселенной компания, где миноритариям что-то достанется?

М.Видео меняет формат допэмиссии. Ранее акционеры одобрили допэмиссию до 500 млн новых акций по закрытой подписке — о...

Mozgovik 03:51

📉 ДАЙДЖЕСТ ПО РЕЙТИНГОВЫМ ДЕЙСТВИЯМ В ВЫСОКОДОХОДНОМ СЕГМЕНТЕ, ПОРТФЕЛЕ PROBONDS ВДО И РОЗНИЧНЫХ ИНВЕСТИЦИОНЫХ ОБЛИГАЦИЙ ЗА...

🟢ПАО «ЕВРОТРАНС» АКРА подтвердило кредитный рейтинг на уровне A-(RU) ПАО «ЕвроТранс» — один из крупнейших независимых топливных операторов на рынке Московского региона....

Иволга Капитал 13.09.2025

Банк России снизил ключевую ставку до 17% — что делать? Мнение аналитиков

Сегодня на заседании Банк России снизил ключевую ставку на 1 п.п., до 17%. Теоретически, для рынков акций и облигаций — это определенно благоприятное событие, но на практике рынок...

БКС Мир инвестиций 13.09.2025