

Audion Cars Counter — руководство для ИИ-агента

Содержание

- [Слова](#)
- [Правила, которые действуют всегда](#)
- [Папка](#)
- [Как привести папку в рабочее состояние](#)
- [Где лежат данные](#)
- [Команды](#)
- [Работа по порядку](#)
- [Что измерено](#)
- [Окно](#)
- [Файлы](#)
- [Если что-то не так](#)

Этот файл — для ИИ-агента, который работает в папке программы вместе с её пользователем: Claude Code, Codex или любого другого, кто умеет запускать команды и смотреть картинки.

Пользователь запускает агента в папке программы и говорит: «Прочитай

`Docs\AI_AGENT_GUIDE_RU.md`». Окно программы считает и выгружает отчёт через команды движка, описанные здесь, и пишет те же файлы разметки, что и они. У обучения сети и у её проверки кнопок нет вовсе: это команды, и запускает их человек или агент.

Что делает программа: считает легковые автомобили на больших спутниковых снимках с привязкой к местности. На таком снимке (около 0,3 м в пикселе) автомобиль — мягкое пятно примерно 14 × 7 пикселей. Маленькая сеть для каждого пикселя отвечает, насколько вероятно, что здесь середина автомобиля; автомобиль — вершина этой карты выше порога. Сеть знает только то, что ей разметили глазом, поэтому работа идёт по кругу: разметить, обучить, проверить, посмотреть, где она ошиблась, разметить это.

С публичным выпуском программы идёт сеть `open-03`, обученная на свободных аэрофотоснимках трёх открытых государственных служб и ни на чём другом (на чём и как её проверяли — `system_core\nets\README.md`). Новый набор снимков считается ею сразу — и верить этому счёту нельзя, пока сеть не проверена на снимках самого пользователя: они другой источник. Программа, из которой сеть вынута, показывает снимки и принимает разметку, а считать не может, пока сети нет (см. «Сети ещё нет: первая сеть»). В приватной копии может лежать другая сеть; тогда новый набор снимков считается ею сразу, но снимки другого источника или местность, какой эта сеть не видела, нужно сначала проверить и, как правило, доучить. Какая это копия, говорит `engine.cmd where`. Об обеих работах и написано руководство.

Люди работают с программой через окно, `Start.exe`, и руководство пользователя (`Docs\USER_GUIDE_RU.md`). Пути ниже — от папки программы; `<work>` — рабочая папка набора снимков (см. «Где лежат данные»).

Слова

Слово	Что оно здесь значит
снимок	один файл <code>.jpg</code> набора с файлом привязки MapInfo <code>.tab</code> рядом; называется по концу имени файла, столбец-строка: <code>2-3</code>
набор	снимки одной папки, которые считаются вместе
участок	квадрат 240 × 240 пикселей снимка (около 75 × 75 м), размеченный целиком
метка	точка, поставленная глазом на участке: автомобиль, грузовик, «не разобрать»; все метки вместе — разметка
«земля»	всё на размеченном участке, у чего нет точки: сеть учат, что автомобиля здесь нет
сеть	файл <code><имя>.pt</code> , которым программа считает; называется по имени без расширения: <code>cars-05</code>
ответ	место, где сеть видит середину автомобиля, и её уверенность от 0 до 1
порог	уверенность, начиная с которой ответ идёт в счёт; в командах и файлах — <code>least</code>
прогон	счёт набора одной сетью: папка <code>count_<сеть></code> в <code><work>\runs</code>
проверочный снимок	снимок, размеченный только для проверки сетей; на нём не учат никогда

Слово	Что оно здесь значит
отметка программы	так раздел СЧЁТ и отчёт называют ответ сети не ниже порога
место	участок по снимку и левому верхнему углу: <code>2-3:8200,300</code>

Правила, которые действуют всегда

- **Снимки — пользователя.** Они только читаются. Не копировать ни их, ни вырезки из них в папку программы и никуда не отправлять.
- **Спросить, прежде чем смотреть.** Всё, что рассматривает агент, — уменьшенный снимок для выбора участков, увеличенный участок для разметки, лист образцов, страница отчёта — уходит в сервис, на котором агент работает. У спутниковых снимков обычно есть условия использования поставщика. До первой картинке спросите пользователя, можно ли смотреть эти снимки. Если нельзя — размечает в окне (раздел ПРОСМОТР) и смотрит листы пользователь, а агент запускает то, чему глаз не нужен: обучение, таблицы, счёт, отчёт.
- **Разметка не стирается и не перезаписывается пачкой.** Это часы работы глазом и единственное, что нельзя получить заново командой. Файлы в `<work>\labels` своими скриптами не удалять, не переносить и не переписывать. `patch seed` в размеченный участок не пишет; `patch set` пишет один участок заново — только намеренно и по одному участку; `patch fix` правит отдельные точки.
- **Сеть не перезаписывается.** Новая сеть получает новое имя; `net train` и `net join` откажут имени, которое уже занято в рабочей папке. Если в программе лежит сеть — пришла с ней или положена в `system_core\nets`; `engine.cmd where` перечисляет все сети по именам, — свою сеть её именем не называть: сеть с таким именем в рабочей папке берётся раньше той, что в программе.
- **На проверочных снимках не учат.** Договоритесь с пользователем, какие это снимки, запишите их список там, где он его найдёт, и называйте в `--hold-tile` при каждом обучении. Сеть проверяют только на участках, на которых её не учили.
- **Движок — только через `engine.cmd`** (он запускает `runtime\python.exe system_core\main.py`). Системный Python не использовать и ничего в него не ставить.
- **Один набор — одна рабочая папка.** Рабочую папку не направлять на другие снимки, а `config\settings.json` не править ради одной команды на другом наборе: этот файл читают окно и все остальные команды. Для этого есть переменные — см. «Где лежат данные».

- **Никогда не называть одно голое число.** Счёт — вместе с границами (отметки минус четверть — отметки плюс половина), процент — вместе с тем, на чём он стоит: 7 % от 109 автомобилей — это один шум.
- **Перед долгим запуском сказать, что это и на сколько:** обучение, счёт целого набора.

Папка

Путь	Что в нём
Start.exe, App\	окно
engine.cmd	движок из командной строки: engine.cmd <инструмент> ...
system_core\	движок: main.py и по файлу на инструмент; что принимает инструмент, сказано в первых строках его файла
system_core\nets\	место сети, которая пришла с программой или положена в неё, — файл <имя>.pt ; в публичном выпуске там open-03 , сеть, обученная на свободных аэрофотоснимках
config\counting_default.json	называет сеть из system_core\nets , которой считать, и её порог: {"net": "<имя>", "least": 0.20}
runtime\	свой Python 3.12 программы с PyTorch, NumPy, OpenCV, Pillow и opencv (ставит раздел УСТАНОВКА)
install\, builder_main.cmd	установщики и их меню
config\settings.json	что помнит окно; движок читает из него "Pictures", "Work" и "Data"
config\version.json	версия
logs\	журнал окна, по файлу на день
Docs\	это руководство и руководство пользователя

Снимков, разметки, сетей пользователя, итогов счёта и отчётов в папке программы нет.

Как привести папку в рабочее состояние

Программа публикуется скелетом: окно, код и сеть `open-03` — движок ещё не установлен. `Start.exe`, раздел УСТАНОВКА, **УСТАНОВИТЬ НУЖНОЕ** ставит Python 3.12 в `runtime\`, PyTorch 2.8.0 (с CUDA 12.8 на видеокарте NVIDIA с драйвером 570 или новее, иначе для процессора) и библиотеки: скачивается от 0,7 до 3,4 Гб, установленное с CUDA занимает около 7 Гб. Кнопка сначала спрашивает и говорит, что придёт под эту машину. Первый шаг установки проверяет, есть ли в Windows Microsoft Visual C++ Redistributable, который нужен PyTorch, и ставит его через winget, если нет (`install\Ensure-VCRedist.ps1`; единственное, что ставится в Windows, с разрешения администратора). Из командной строки: `builder_main.cmd install_all`. Потом:

```
builder_main.cmd verify      по строке на каждую проверку; должен закончиться строкой
«[OK] Everything is in place.» или «[OK] In place, with N warning(s).»
engine.cmd doctor           одна строка JSON: python, torch, device (cuda или cpu),
card, библиотеки, net, net_file, problems
engine.cmd where            папки, сколько снимков, размеченных участков и сетей,
какой сетью считается набор, какой torch и какая видеокарта
```

Команде `doctor` снимки не нужны. `"device": "cpu"` на машине с видеокартой NVIDIA: `builder_main.cmd check_machine` скажет почему (обычно драйвер), `builder_main.cmd torch_cuda` поставит PyTorch с CUDA. `builder_main.cmd run_tests` прогоняет собственные тесты движка на нарисованном снимке; снимки пользователя при этом не трогаются.

Программа без сети — не сломанная установка. `doctor` тогда отвечает `"net": ""` и `"net_file": ""`; в его `problems` — только библиотеки, и код выхода 1 бывает, только когда не хватает библиотеки. `verify` даёт на это строку `[WARN]`, а не `[FAIL]`. `where` печатает `counts with: no net yet (none taught in the work folder, none came with the program)`. В окне раздел УСТАНОВКА показывает строку «Обученной сети нет» и итог «Движок установлен. Обученной сети нет: считать пока нечем», а раздел СЧЁТ говорит «Обученной сети нет ни в рабочей папке, ни в программе: считать нечем.» Всё остальное работает: просмотр, разметка, обучение. Что тогда делать — ниже, «Сети ещё нет: первая сеть».

Из PowerShell команда пишется `.\engine.cmd where`.

Где лежат данные

На каждый род данных в разделе НАСТРОЙКИ есть поле; те же ключи файла `config\settings.json` читает движок.

Поле	Ключ	Если не задан	Что это
СНИМКИ	"Pictures": полный путь или имя папки внутри Pictures папки данных	папка со снимками внутри Pictures папки данных: та, на которой сделана папка Work; в новой папке данных — первая такая папка по имени или сама Pictures, когда снимки лежат прямо в ней	набор, который считается; только читается
РАБОЧАЯ ПАПКА	"Work": полный путь	Work папки данных — для набора по умолчанию, Work-<имя папки снимков> — для любого другого	разметка, итоги счёта, плитки просмотра, сети, обученные здесь
ОТЧЁТЫ	"ReportFolder" (только окно)	спрашивается при первой выгрузке	куда окно пишет отчёты и слои; команде report папку даёт --out

Папку данных программа ищет рядом со своей: Audion Counters Data\Cars Counter, а если её нет — Cars Counter (другую называет ключ "Data" или переменная CARS_COUNTER_DATA). Новый пользователь кладёт снимки в папку с любым именем внутри её папки Pictures (Pictures\Город), и рабочая папка Work появляется сама. Не полагайтесь на это: engine.cmd where печатает папки, которые он взял.

Где ищется сеть, по порядку: сначала <work>\models — там лежит сеть, обученная для этого набора, и принятую называет models\counting.json; потом сама программа: config\nets (файл <имя>.credential.pt — сеть, которую хранят как ключ: инструменты, убирающие секреты из config, не кладут её в публичный архив) и system_core\nets (файл <имя>.pt) — там ту, которой считать, называет config\counting_default.json. Поэтому сеть, обученная в рабочей папке, считает набор этой папки, а сеть в программе — каждый набор, в рабочей папке которого сеть не названа.

Снимок — файл <что угодно>_<столбец>--<строка>.jpg с одноимённым .tab рядом. Размер снимка, размер его пикселя, площадь и любое место на местности берутся из .tab; его единицы программа принимает за метры. Снимок без .tab считается, но у него нет площади и нет места в слоях.

Рабочая папка принадлежит одному набору снимков. Разметка, плитки и прогоны находятся в ней только по концу имени снимка (2-3), а у другого набора — другого города, другой съёмки того же места — концы имён те же. Папка помнит снимки, на которых сделана

(`<work>\set.json`), и все инструменты, кроме `where` и `info`, отказывают ей с другими снимками. Числа двух наборов не складываются никогда.

Одна команда на другом наборе, не трогая настроек:

```
set CARS_COUNTER_PICTURES=<полный путь или имя папки внутри Pictures>
set CARS_COUNTER_WORK=<полный путь>
engine.cmd where
```

(в PowerShell: `$env:CARS_COUNTER_PICTURES = '...'`). Без `CARS_COUNTER_WORK` рабочая папка этого набора берётся по правилу из таблицы. `CARS_COUNTER_NETS` называет на одну команду одну папку вместо двух папок программы. `CARS_COUNTER_PYTHON` называет для `engine.cmd` другой `python.exe`; оставьте её незаданной.

Команды

Каждая команда — `engine.cmd <инструмент> ...` из папки программы. `T` — снимок (`2-3`), `СЕТЬ` — сеть по имени без расширения, `ПРОГОН` — прогон по имени папки.

where папки, снимки, размеченные участки, сети,
 torch и видеокарта
 doctor на чём работает движок, одна строка JSON
 layout как снимки лежат на местности и сколько
 общего у соседей
 geo T размер пикселя снимка, его угол, поворот
 overview T [x,y,w,h] [уменьшение] [частей] [сетка]
 снимок или его область в уменьшенном виде, с
 сеткой его пикселей
 patch show --tile T --at "x,y x,y ..." картинки участков для просмотра; в разметку
 ничего не пишется
 patch seed --tile T --at "x,y ..." --by СЕТЬ [--least 0.2]
 первые точки участков, у которых их нет, –
 сетью
 patch set --tile T --at x,y --points "px,py px,py,t ..." все точки одного участка заново, в пикселях
 участка
 patch fix --tile T --at x,y [--drop "3 7"] [--add "px,py ..."] [--move "5:px,py"]
 [--kind "4:t"]
 правка пронумерованных точек одного участка
 patch snap --tile T --at "x,y ..." --by СЕТЬ
 метки автомобилей сдвигаются на ответ сети в
 пределах 8 пикселей
 patch list все размеченные участки и итоги
 montage T ИМЯ "x,y x,y ..." четыре нарисованных участка на одном листе
 net train --name ИМЯ [--hold-tile "T T"] [--hold "T:x,y ..."] [--steps 2000] [--seed
 1] [--width 20] [--lessons 2|3]
 [--also СНИМКИ РАБОЧАЯ ["T T"]] ... учить ещё и на разметке другого
 набора снимков: папка его снимков,
 его рабочая папка, его отложенные
 снимки; наборы остаются порознь
 сеть, обученная на всех размеченных
 участках, кроме отложенных
 net join --name ИМЯ --of "СЕТЬ СЕТЬ ..." несколько сетей в одном файле, который
 отвечает их средним
 net check --weights СЕТЬ --at "T:x,y ..." [--each]
 сеть против меток названных участков, по
 порогам
 net count --weights СЕТЬ|auto --tile T [T ... | all] [--out ПРОГОН] [--region
 x,y,w,h] [--again] [--json]
 все ответы сети на целых снимках
 stability СЕТЬ одинакова ли ошибка от места к месту

shaken СЕТЬ ПОРОГ	держится ли счёт, когда снимок просто
выглядит иначе	
overlay СЕТЬ "T:x,y ..." [порог]	метки и ответы на одном участке
answers ПРОГОН T [порог] [образцов]	где лежат ответы посчитанного снимка и лист
образцов	
report ПРОГОН --out ПАПКА [--tile T ...] [--least 0.20] [--lang ru en] [--formats	
xlsx docx mif]	
[--cell 250] [--samples 18] [--json]	
	счёт файлами для людей и слоями для MapInfo
info, thumbs, pyramid T	то, что окно спрашивает у движка; печатают
строки JSON	

`--weights auto` — сеть, которой считается набор: та, что названа в `<work>\models\counting.json`, а если там сеть не названа — та, что названа в `config\counting_default.json` и лежит в программе; когда нет ни той ни другой, команда останавливается со словами `no net to count with: ...`. Сеть по имени ищется сначала в `<work>\models`, потом в `config\nets` и `system_core\nets`.

Часть инструментов спрашивает ответы у сети: `patch seed`, `patch snap`, `net check`, `net count`, `stability`, `shaken`, `overlay`, — а `answers` и `report` читают прогон, посчитанный сетью. В программе без сети они вступают в дело после первого обучения; всё остальное работает с самого начала.

Работа по порядку

0. Где что лежит

```
engine.cmd where
```

Папки, сколько снимков, размеченных участков и сетей, какой сетью считается набор, какой torch и какая видеокарта. Строка `PROBLEM:` значит, что рабочая папка сделана на других снимках: назовите для этого набора другую.

Строка `counts with:` говорит, с чего начинается работа:

- `counts with: <сеть> at <порог> (<её файл>)` — сеть есть: дальше шаг 1;
- `counts with: no net yet (none taught in the work folder, none came with the program)` — сети нет: дальше по главе «Сети ещё нет: первая сеть».

Сети ещё нет: первая сеть

Без сети нечего проверять и нечем считать, и ни один инструмент, который спрашивает ответы у сети, ещё не работает: нет ни первых точек от сети, ни `snap`, ни таблиц. Каждая точка, на которой учится первая сеть, ставится глазом — агентом по нарисованным участкам или пользователем в окне. Шаги — те же пронумерованные, что ниже, в таком порядке:

1. **Разрешение смотреть и резкость.** Спросите пользователя, можно ли смотреть эти снимки («Правила, которые действуют всегда»). Потом нарисуйте несколько участков со снимков из разных концов набора и посмотрите на них:

```
engine.cmd patch show --tile 2-3 --at "8200,300 9000,300"      ->
<work>\runs\look\2-3\8200_300.jpg, 9000_300.jpg
engine.cmd geo 2-3                                             размер пикселя
```

`patch show` только рисует: в разметку ничего не пишется. Набор, который местами размыт, устойчивой ошибки не даст, чему его ни учи: скажите это пользователю до всякой разметки.

2. **Проверочные снимки.** Договоритесь с пользователем о двух-трёх снимках с разной местностью, запишите их список и откладывайте их при каждом обучении.
3. **Участки по карте** — шаг 2: ровная сетка на проверочных снимках; на остальных — сетка и места, на которых сеть обычно ошибается.
4. **Разметка глазом** — шаг 4; шаг 3 пропускается. Нарисуйте каждый участок командой `patch show`, прочитайте его автомобили по сетке и запишите их командой `patch set` (участок без автомобилей — `--points ""`); `patch fix` правит отдельные точки, `montage` кладёт четыре нарисованных участка на один лист, `patch list` даёт итоги. Или те же участки размечает в окне пользователь. Проверочные участки и участки для обучения размечаются одинаково. Для меры: сеть автора училась примерно на 180 участках, на которых около 530 автомобилей, а проверка различает две сети, только когда в ней несколько сотен автомобилей.
5. **Четыре сети, склеенные в одну** — шаг 5 без `patch snap`: пять его команд как есть, с проверочными снимками в `--hold-tile`.
6. **Таблицы** — шаг 6: `engine.cmd stability cars-05`, порог читается из этой таблицы, потом `engine.cmd shaken cars-05 <порог>`. С этого места сеть есть, и работает каждый инструмент.

7. **Целые снимки** — шаг 7: посчитайте один-два снимка с `--weights cars-05` и посмотрите на лист `answers`. То, что на нём не автомобиль, размечается участками без автомобилей; потом учатся ещё четыре сети под новыми именами (`cars-06` ... `cars-09`), склеенные в `cars-10` и ставятся рядом с `cars-05`. На этом втором круге новым участкам первые точки может дать сеть (шаг 3).
8. **Сеть, которой считается набор** — шаг 8: когда пользователь принимает ошибку, которую показывают таблицы, запишите `<work>\models\counting.json`. С этого момента этой сетью считают `net count --weights auto` и кнопка ПОСЧИТАТЬ СНИМКИ; попросите пользователя нажать ПЕРЕЧИТАТЬ.

1. Экзамен: что имеющаяся сеть делает на этих снимках

Этому шагу нужна сеть: та, что пришла с программой, та, что передали пользователю, или обученная раньше. Экзамен идёт до всякого обучения. Учить нужно только то, что он провалил.

1. **Резкость.** С разрешения пользователя нарисуйте несколько участков со снимков из разных концов набора (`patch show`) и посмотрите. Набор, который местами размыт, устойчивой ошибки не даст, чему его ни учи. `engine.cmd geo 2-3` печатает размер пикселя; сети автора учились примерно на 0,3 м.
2. **Взгляд на целые снимки.** Посчитайте два-три снимка с разной местностью и разложите образцы ответов на листы (`<сеть>` — имя из строки `counts with:`):

```
engine.cmd net count --weights auto --tile 2-3 4-1      ->
<work>\runs\count_<сеть>\2-3.csv, summary.json
engine.cmd answers count_<сеть> 2-3 0.2 80             -> 2-3_where.jpg, 2-
3_sample.jpg в папке прогона
```

Какая доля на листе — не автомобили и что это такое: это и есть то, что размечать дальше.

3. **Числа.** Положите ровную сетку проверочных участков на два-три снимка (шаг 2), возьмите первые точки у сети (шаг 3), поправьте их глазом (шаг 4) и запустите

```
engine.cmd stability <сеть>
```

Для сети, которая пришла с программой, все ваши участки новые, поэтому каждый размеченный снимок стоит в таблице двумя половинами. (Таблица узнаёт «обученное» по имени снимка и углу участка: ваш участок, у которого совпало и то и другое с участком, на котором сеть училась, окажется в столбце `taught`.) Читать её — как сказано в шаге 6. Метки,

начатые с черновика самой сети, тянутся к тому, что сеть видит: «пропущено» тогда — оценка снизу, поэтому ищите автомобили без точки.

4. **Вывод.** Мерило, под которое программа сделана, — не точное число, а ошибка, которая держится: 5, 10 или 20 % годятся, когда они одни и те же всюду, потому что их исправляет один коэффициент; ошибка, которая меняется от места к месту, не годится. Спросите пользователя, какая ошибка его устроит, и назовите ему три вещи из шага 6 вместе с числом автомобилей, на котором они стоят.

2. Выбрать участки по карте

```
engine.cmd overview 2-3                весь снимок, линия через каждые 2000
пикселей  -> <work>\runs\overview\2-3_0.jpg
engine.cmd overview 2-3 8000,0,4490,3000 0.4    область, крупнее
```

Выбирают по тому, что видно на карте, а не по тому, где стоят автомобили:

- **участки для проверки** — ровная сетка по застройке проверочных снимков, берётся каждая ячейка, пустые тоже. Половина окажется лесом и крышами; пустой участок — тоже ответ. Проверочный снимок одного вида местности проверяет этот вид: берите несколько — открытую стоянку, жилые кварталы, старый квартал в тени;
- **участки для обучения** — тоже сетка, на других снимках, и места тех видов, на которых сеть ошибается: стоянки в тени дома, плотные ряды, дворы, дорога и трудная «земля» (шаг 7).

Проверка, в которой несколько десятков автомобилей, не различает ничего тоньше десятков процентов: цельтесь в несколько сотен автомобилей, размеченных на проверочных снимках.

3. Первые точки — сетью

Этому шагу нужна сеть: **<сеть>** — та, что пришла с программой, или та, которую вы обучили последней. До первой сети первых точек нет: участки рисуют командой **patch show** вместо **patch seed**, и каждая точка ставится глазом (шаг 4).

```
engine.cmd patch seed --tile 2-3 --at "8200,300 9000,300 9800,300" --by <сеть> --
least 0.2
engine.cmd montage 2-3 town "8200,300 9000,300 9800,300"
```

seed записывает ответы сети метками автомобилей и рисует каждый участок увеличенным в шесть раз и осветлённым: **<work>\runs\look\2-3\8200_300.jpg**, с сеткой через 20 пикселей и номерами пикселей участка сверху и слева; **patch show** рисует ту же картинку и меток не

пишет. `montage` кладёт четыре нарисованных участка на один лист (`<work>\runs\sheets\town_0.jpg`), чтобы быстро пройти редкие. Точки сети — черновик: она пропускает тёмные автомобили и принимает за автомобили другое.

4. Разметить глазом

Смотрите на картинку участка и записывайте каждый его автомобиль. Координаты — участка, их читают по сетке.

```
engine.cmd patch set --tile 2-3 --at 8200,300 --points "48,150 112,60 130,64,q
201,33,t"
engine.cmd patch fix --tile 2-3 --at 8200,300 --drop "3 7" --add "88,120 95,131,q" --
move "5:60,44" --kind "4:t"
engine.cmd patch set --tile 2-3 --at 9000,300 --points ""
engine.cmd patch list
```

`set` пишет все точки участка заново; `fix` правит пронумерованные точки, которые видны на картинке; пустой `set` говорит «автомобилей здесь нет, всё это „земля“». Обе команды рисуют картинку заново — посмотрите на неё ещё раз: точка должна стоять на автомобиле, а номера точек — те, что на последней нарисованной картинке.

Вид	Что это	Для сети
(нет) или <code>c</code>	автомобиль: легковой, внедорожник, фургон, пикап	автомобиль
<code>t</code>	грузовик, автобус, прицеп, техника	в счёт не идёт; «землём» на нём не учат
<code>q</code>	не разобрать: тёмное пятно в тени, наполовину под деревом, пятно, которое может быть автомобилем	то же
точки нет	всё остальное на участке	«земля»

Правила, за которые уже заплачено:

- **Участок размечается целиком.** Всё, у чего нет точки, идёт в обучение как «земля», поэтому автомобиль без точки учит сеть, что такие автомобили — «земля».
- **Сомнительное — `q`.** Неверная метка учит хуже, чем пропущенная. В старом квартале в длинных тенях меток `q` может быть больше, чем автомобилей, — таков снимок, это не

ошибка.

- **Точка ставится на середину автомобиля**, между стеклом и кузовом, а не на нос. В ряду стоящих автомобилей считайте вдоль ряда: они стоят через 7–9 пикселей.
- **Тёмные автомобили в тени дома там есть**. Картинка для этого и освещена; ищите пары светлых точек стёкол в ряд.
- **Не автомобили**: надгробия рядами, вентиляция и люки на крыше, грядки и теплицы, штабели поддонов, лодки, крыши сараев и гаражей, одиночные светлые кроны. Крыша гаража — в два-три автомобиля шириной.
- Очень плотный участок рассматривают отдельно, а не на листе из четырёх.

Разметка — это файлы: `<work>\labels\<снимок>\<x>_<y>.csv`, столбцы `x,y,kind` в пикселях снимка. Файл `<work>\labels\<снимок>\ground.csv` (столбцы `x,y,w,h`, по прямоугольнику в строке, каждый шире и выше 96 пикселей) называет прямоугольники, где автомобилей нет вовсе, — лес, поле; из них для обучения нарезается простая «земля». Команды для него нет, он пишется руками.

Человек размечает те же участки в окне (раздел ПРОСМОТР) щелчками. Файлы одни и те же, поэтому метки, поставленные командой и щелчком, смешиваются свободно; см. «Окно».

5. Обучить

```
engine.cmd patch snap --tile 3-3 --at "700,4700 2100,4700" --by <сеть>
```

`snap` — для участков, где метки автомобилей поставлены глазом с нуля: он сдвигает каждую метку автомобиля на ответ самой сети в пределах 8 пикселей, чтобы метки всех участков значили одно — середину автомобиля. Ничего не добавляет и не убирает. Участкам, чьи метки начаты с `seed`, он не нужен. Ему нужна сеть, которая уже есть: до первого обучения его пропускают и держатся правила из шага 4 — точка ставится на середину автомобиля.

```
engine.cmd net train --name cars-01 --steps 8000 --seed 1 --hold-tile "2-4 3-1"
engine.cmd net train --name cars-02 --steps 8000 --seed 2 --hold-tile "2-4 3-1"
engine.cmd net train --name cars-03 --steps 8000 --seed 3 --hold-tile "2-4 3-1"
engine.cmd net train --name cars-04 --steps 8000 --seed 4 --hold-tile "2-4 3-1"
engine.cmd net join --name cars-05 --of "cars-01 cars-02 cars-03 cars-04"
```

`train` учит новую сеть с нуля на всех размеченных участках рабочей папки, кроме отложенных (`--hold-tile` — целые снимки, `--hold` — места), пишет `<work>\models\<имя>.pt` и печатает проверку на отложенном. Разметка, на которой училась сеть, пришедшая с

программой, с программой не идёт: ваша сеть знает только разметку пользователя.

Откладываете **каждый** проверочный снимок, каждый раз. Имена — пример: занятому имени движок откажет, поэтому следующие четыре — `cars-06` ... `cars-09`, склеенные в `cars-10`.

`--seed` — случайное начало обучения: первые веса и то, какие квадраты участков показываются. Одно обучение — один бросок кости: та же разметка с другого начала даёт сеть, которая при том же пороге считает до 13 % иначе. Поэтому сеть для счёта делается из четырёх, и `join` пишет один файл, который отвечает средним своих участников; любая команда берёт его по имени, как обычную сеть. Сравнивайте между собой только такие четвёрки, а не две одиночные сети.

Время, в работе автора: 8000 шагов шли шесть минут на RTX 5070, четыре обучения рядом — десять; 2000 шагов на процессоре ноутбука шли 40 минут. `--width` и прочие ключи оставьте как есть: сеть шире (`--width 32`) не дала ничего сверх шума, а больше шагов на той же малой разметке дали сеть, которая знает свои участки, а не автомобили.

6. Проверить: держится ли ошибка

```
engine.cmd stability cars-05
```

Столбцы: участки, на которых сеть училась (`taught`), потом каждый снимок, на котором она не училась, двумя половинами (`2-4 a`, `2-4 b`). Строки — пороги. В ячейке: сколько насчитано, ошибка против разметки, `m` — пропущено автомобилей, `e` — ответов на «земле». Под таблицей порог выбирается на одной половине — там, где счёт сходится с разметкой, — а другая половина показывает, какой ошибки ждать; потом наоборот.

Читать три вещи, в таком порядке:

1. **ошибку на другой половине**, в обе стороны. Два числа должны быть близки; расстояние между ними — та часть ошибки, которая не держится;
2. **насколько сдвигается счёт, когда порог сдвигается на 0,05** (строка «with the threshold 0.05 lower / higher»). Уверенная в себе сеть сдвигается на несколько процентов; счёт, который гуляет на треть, уйдёт от любой перемены света между полосами мозаики;
3. `m` **против** `e`. Счёт, который сошёлся потому, что 25 пропущенных погашены 25 лишними, устойчивым не считается: в другом месте они гасить друг друга не обязаны.

Потом сравните столбец обученных участков с новыми: порог, при котором каждый сходится со своей разметкой. Сеть, которая на своих участках сходится при более высоком пороге, знает

свои участки лучше, чем автомобили; больше шагов этот разрыв растят, больше разметки — закрывают.

Порог принадлежит сети: его читают из её собственной таблицы (в работе автора склеенные сети сходились с разметкой при 0,20, их участники — при 0,25). Этот порог и берите для `seed`, `overlay`, `answers`, `shaken`, счёта и отчёта.

Чего таблица сказать не может:

- **В половине снимка может быть 25–60 автомобилей.** Разница в десять процентов между такими половинами — это кость, а не сеть. Две сети различимы, только когда в проверке несколько сотен автомобилей; до тех пор говорите «одинаковы в пределах проверки».
- **«Лишний» — не значит «ложный».** Многие ответы, которые таблица называет лишними, стоят на машинах: тёмный автомобиль в тени, чья метка `q` дальше 7 пикселей от ответа, второй автомобиль пары, грузовик. Посмотрите на них (`overlay`), прежде чем называть сеть шумной; ложные ответы считают глазом по листу `answers`.

```
engine.cmd shaken cars-05 0.20
```

Те же новые участки делаются светлее, темнее, плоче, жёстче, мягче, резче, с оттенком, сильнее сжатыми в JPEG — двенадцатью способами — и считаются при этом пороге. На местности ничего не меняется, значит, не должен меняться и счёт. Последняя строка говорит, как далеко он ушёл. Это ошибка, которой ждать между полосами мозаики, выглядящими по-разному.

Увидеть, в чём сеть ошибается на участке, — кольца это метки, кресты — ответы, зелёные, когда нашли размеченный автомобиль:

```
engine.cmd overlay cars-05 "2-3:11690,1960 2-4:9400,700" 0.2 -->
<work>\runs\overlay\2-3_11690_1960.jpg
```

Сеть лучше, когда лучше все три вещи из таблицы, а не когда одно число ближе к нулю. Таблицы только печатаются: сохраняйте их текст для пользователя (например, в `<work>\runs\log\`), чтобы рядом можно было поставить следующую сеть.

7. Посчитать целые снимки и посмотреть на ложные ответы

```
engine.cmd net count --weights cars-05 --tile 3-2 ->
<work>\runs\count_cars-05\3-2.csv, summary.json
engine.cmd answers count_cars-05 3-2 0.2 80 -> 3-2_where.jpg, 3-
2_sample.jpg
```

`count` записывает каждый ответ от уверенности 0,1 с его местом на снимке и на местности (из `.tab`), поэтому порог выбирается по таблице уже после счёта. `answers` рисует, где они лежат, и лист из 80 случайных с уверенностью сети. Смотрите на лист: то, что на нём не автомобиль, — следующее, что размечать, участками в этих местах (шаги 3–4; участок с надгробиями — это `patch set --points ""`). Так они становятся трудной «землёй» для следующего обучения.

Снимок каждого нового вида местности посчитайте целиком, прежде чем доверять на нём сети.

8. Весь набор и сеть, которой он считается

```
engine.cmd net count --weights cars-05 --tile all
engine.cmd layout
```

Одна папка прогона, по файлу csv на снимок, один `summary.json` со счётом каждого снимка и набора при порогах 0,15, 0,20, 0,25, 0,35, 0,45 и 0,55; остановленный счёт продолжается со снимков, которые он ещё не записал. Прогон — счёт одной сетью и одного набора. `layout` показывает, что общего у соседей: каждый снимок считается целиком, на стыках ничего не вычитается, и автомобиль на общей полосе считается дважды (у набора, на котором программа сделана, эта полоса — несколько пикселей).

Сеть, которой считается набор, названа в `<work>\models\counting.json`:

```
{"net": "cars-05", "least": 0.20}
```

Команды, которая пишет этот файл, нет: его пишет тот, кто принимает сеть. Её берут `net count --weights auto` и кнопка ПОСЧИТАТЬ СНИМКИ и считают в прогон `count_cars-05`. Записывайте туда сеть только после её таблиц `stability` и `shaken` и только если она измеримо лучше той, что считала раньше, — иначе число по набору сдвинется ни за что. У первой сети набора предшественницы нет: записывайте её, когда пользователь увидел её таблицы и принимает ошибку, которую они показывают. `least` должен быть одним из шести порогов `summary.json`: окно показывает числа прогона только при них. Чтобы вернуться к

сети, которая пришла с программой, назовите там её с её собственным порогом — и то и другое записано в `config\counting_default.json`.

Этот файл — одного набора: сеть, обученная здесь, лежит в `<work>\models` и считает этот набор. Чтобы считать ею другие наборы — по слову пользователя, — положите копию её файла в `system_core\nets` и назовите её в `config\counting_default.json` так же: `{"net": "cars-05", "least": 0.20}`. Точно так же кладут файл сети, который пользователю передали. Сеть в программе считает каждый набор, в рабочей папке которого сеть не названа, поэтому новый набор снимков считается сразу — а когда его источник или местность новые, сначала проходит экзамен (шаг 1).

Когда счёт на проверке устойчиво смещён, коэффициент — это «насчитано / размечено» на проверке при этом пороге. Назовите пользователю счёт, коэффициент и то, на чём он стоит; сама программа коэффициентов не применяет.

9. Отчёт и слои

```
engine.cmd report count_cars-05 --out <папка>          весь прогон: документ
Word и ведомость Excel
engine.cmd report count_cars-05 --out <папка> --tile 2-3 2-4  только эти снимки
engine.cmd report count_cars-05 --out <папка> --formats mif --cell 250
                                                         слои для MapInfo
```

Здесь ничего не считается: числа — те, что у прогона при `--least` (по умолчанию порог, которым считается набор). Файлы названы по тому, что в них и когда они сделаны, на языке `--lang` (по умолчанию `ru`; окно передаёт свой язык).

Документ Word: числа (снимков, отметок программы, автомобилей не меньше и не больше, площадь, отметок на квадратный километр), как понимать число, схема снимков со счётном на каждом, таблица по снимкам, потом по странице на снимок — снимок с точкой на каждой отметке и 18 отметок, взятых наугад, увеличенно. Ведомость Excel: таблица по снимкам, отметки при каждом пороге, каждая отметка с местом на снимке и на местности и лист «Служебное» — сеть, порог, папка прогона.

`--formats mif` пишет в системе координат файлов `.tab` снимков (строка `CoordSys` первого снимка переносится слово в слово):

Файл	Объект	Столбцы
Автомобили_отметки_<когда>.mif + .mid	точка на отметку	снимок, уверенность, пиксель на снимке
Автомобили_сетка_250м_<когда>.mif + .mid	квадрат <code>-- cell</code> метров на местности	номер квадрата, отметок, автомобилей от и до, доля под снимками (%), площадь этой доли, отметок на км ² этой доли
Автомобили_снимки_<когда>.mif + .mid	контур снимка	снимок, отметок, автомобилей от и до, площадь, отметок на км ²
Автомобили_растр_плотности_<когда>.png + .tab	сетка картинкой, квадрат закрашен по числу отметок	привязка раstra: MapInfo открывает <code>.tab</code> как есть
Автомобили_карта_плотности_<когда>.png , Автомобили_карта_плотности_тёмная_<когда>.png	та же карта со шкалой цветов и подписью	для отчёта и для экрана
Автомобили_слои_описание_<когда>.txt		что это за слои и столбцы

(С `--lang en` имена английские.) Квадраты кладутся по местности — `floor(x / cell)`, `floor(y / cell)`, — поэтому у двух подсчётов квадраты одни и те же. Квадрат на краю набора покрыт частично: его плотность считается по покрытой доле. Снимок без `.tab` в слои не попадает; если `.tab` нет ни у одного, команда говорит об этом и ничего не пишет. Таблицы — MIF/MID, текстовый обменный формат MapInfo (Таблица > Импорт), а не родные таблицы `.TAB`. **Файлы прочитаны обратно и сверены до цифры, но в самом MapInfo не открывались ни разу: скажите это пользователю**, а то, что он сообщит из MapInfo, исправляйте первым.

Границы — отметки минус четверть и отметки плюс половина — заданы в программе (`BAND` в `system_core\report.py`; окно берёт те же). Они получены на проверочных снимках набора, на котором программа сделана: там меток «не разобрать» оказалось больше, чем уверенных автомобилей. Для снимков пользователя они заново не выводятся — это тоже нужно сказать.

Слова отчёта — слова читателя: «отметка программы», «автомобилей от ... до ...», «снимок».

Рассказывая результат тем, кто пользуется числом, говорите так же; «сеть», «порог» и «прогон» — для служебного листа.

Что измерено

В работе автора, с сетями автора, на наборе, на котором программа сделана: один город, один источник, около 0,3 м в пикселе. Сеть автора в публичный выпуск не входит (в приватной копии она может быть), поэтому это не свойства копии, в которой вы работаете: это то, что дала сеть, обученная таким способом, — числа, которых ждать от сети, обученной вами, и рядом с которыми её ставить. Ни одно из них не обещание для других снимков.

- Сеть автора — четыре сети, обученные на одной разметке (около 180 участков, около 530 автомобилей) и склеенные; её порог — 0,20. «Сеть» ниже — эта сеть.
- На трёх проверочных снимках, которых она не видела, уверенно размечено 511 автомобилей. Насчитано 509; пропущено 45, лишних 43. По снимкам: завод с большой открытой стоянкой –2 % (345 из 353, пропущено 16, лишних 8); жилая застройка — посёлок, деревня, производственная площадка — +12 % (55 из 49); старый город в длинных тенях 0 % (109 из 109), но там 25 автомобилей пропущено и 25 ответов лишние, и они гасят друг друга.
- При порогах 0,15 и 0,25 вместо 0,20 счёт на снимке со стоянкой — +6 % и –6 % к разметке; в старом городе тот же шаг сдвигает счёт на +39 % и –18 %.
- Снимок, который просто выглядит иначе (**shaken**), сдвигает счёт до 4 % на открытых стоянках и до 16 % в старом городе в тенях.
- В старом городе сам глаз оставил больше меток «не разобрать», чем автомобилей, и сеть отвечает и на них. Отсюда границы числа; сузить их сетью нельзя — только более резким снимком того же места.
- Глазом по листам из 80 ответов на двух целых снимках: на снимке города около 5 % — не автомобили и 6 % — грузовики и фургоны; на снимке без единой метки около 7 % — не автомобили или техника.
- Местность вида, которого нет в разметке, даёт ложные ответы сотнями: снимок завода дал 954 ответа, около половины ложных, пока его крыши не разместили; кладбище дало ранней сети 94 ответа, а сети, обученной с десятью участками трудной «земли» этого снимка, — 15.
- Другой источник снимков того же города — ниже фактическое разрешение, местами размыт: по меткам, поставленным на нём глазом (167 автомобилей), сеть нашла 58, около трети; светлые автомобили она брала, тёмные пропускала. Затемнение этих снимков и подгонка их тона не помогли.

- Ещё разметка того же вида перестаёт помогать: на двадцати новых участках центра города сеть нашла 90 автомобилей из 92 ещё до того, как на них училась, а сеть, обученная и на них, вышла «одинаковой в пределах проверки».
- Готовые детекторы для съёмки сверху пробовали на этих снимках при увеличениях от $\times 1$ до $\times 10$: лучший нашёл около двух третей автомобилей одной стоянки, почти только светлые, и один автомобиль из 22 во дворе. Предлагать их как есть не нужно.
- Таблицы одной сети на видеокарте и на процессоре сошлись с точностью до одного-двух автомобилей.

Окно

Start.exe. Оно показывает то, с чем работают команды, и пишет те же файлы разметки; здесь размечает пользователь, когда агенту смотреть снимки нельзя.

- **ИЗОБРАЖЕНИЯ** — набор, как он лежит на местности. На каждом месте: размеченные участки и автомобили снимка, а для прогона и порога, выбранных справа (ОТВЕТЫ СЕТИ), — число ответов. Щелчок открывает снимок в ПРОСМОТРЕ.
- **ПРОСМОТР** — снимок целиком мелко справа, один кусок крупно посередине. Кусок двигается шагами: стрелками на клавиатуре или кнопками, щелчком по общему виду, Page Up / Page Down — по размеченным участкам. Размеры куска: КРУПНО (120 пикселей), УЧАСТОК (240), 2×2 (480), 4×4 (960).
- **Разметка в ПРОСМОТРЕ** — инструменты СМОТРЕТЬ, МАШИНА, ГРУЗОВИК, НЕ РАЗОБРАТЬ, СТЕРЕТЬ (клавиши 1–5). Щелчок ставит точку того вида, какой у инструмента; щелчок по точке меняет её вид, а инструментом её же вида — убирает; правая кнопка стирает при любом инструменте. Первая точка в квадрате сетки 240 делает этот квадрат участком; УЧАСТОК БЕЗ МАШИН записывает пустой; ВЗЯТЬ ОТВЕТЫ СЕТИ ставит ответы показанного прогона метками автомобилей — черновик, который нужно поправить; ОТМЕНИТЬ (Ctrl+Z) возвращает на шаг назад; УБРАТЬ УЧАСТОК переносит его файл в `<work>\labels\_removed`. Каждая правка пишется сразу.
- **Слои** — ОТВЕТЫ СЕТИ, РАЗМЕТКА, ОСВЕТЛИТЬ (клавиши A, M, L).
- **СЧЁТ** — ПОСЧИТАТЬ СНИМКИ считает каждый снимок, которого ещё нет в прогоне сети для счёта; ОТЧЁТ ПО ВСЕМ СНИМКАМ, ОТЧЁТ ПО ЭТОМУ СНИМКУ и СЛОИ ДЛЯ MAPINFO — это команда **report** из шага 9; ряд «Квадрат сетки» (100, 250, 500, 1000 м) — её **--cell**. Без сети кнопка приглушена, а раздел говорит «Обученной сети нет ни в рабочей папке, ни в программе: считать нечем.»

- **ПЕРЕЧИТАТЬ** (внизу справа) заново спрашивает движок о наборе: нажмите или попросите нажать после того, как разметку, сети или прогоны поменяли командами.

Файлы

Файл	Что это
<code><work>\set.json</code>	имя набора и имена файлов снимков, на которых сделана рабочая папка
<code><work>\labels\<снимок>\<x>_<y>.csv</code>	метки участка: <code>x,y,kind</code> , пиксели снимка
<code><work>\labels\<снимок>\ground.csv</code>	прямоугольники простой «земли»: <code>x,y,w,h</code>
<code><work>\labels_removed\<снимок>\</code>	участки, убранные в окне; их ничто не читает
<code><work>\models\<имя>.pt</code>	сеть: её веса и запись о том, на чём она училась, — места участков, отложенное, шаги, секунды, устройство
<code><work>\models\counting.json</code>	сеть, которой считается этот набор, и её порог
<code><work>\runs\count_<сеть>\<снимок>.csv</code>	ответы: <code>x,y,sure,ground_x,ground_y</code>
<code><work>\runs\count_<сеть>\summary.json</code>	<code>weights</code> , <code>device</code> , <code>source</code> (набор), <code>pictures</code> (счёт каждого по порогам), <code>total_by_least</code>
<code><work>\runs\look</code> , <code>sheets</code> , <code>overlay</code> , <code>overview</code>	картинки для просмотра; команды делают их заново
<code><work>\runs\thumbs</code> , <code><work>\runs\pyramid</code>	мелкие снимки и плитки для окна; делаются заново, когда их нет

Если что-то не так

Что напечатано	Что делать
<code>Python is not found</code>	движок не установлен: УСТАНОВКА, УСТАНОВИТЬ НУЖНОЕ или <code>builder_main.cmd install_all</code>

Что напечатано	Что делать
<code>where</code> говорит <code>no card, the processor</code> на машине с видеокартой NVIDIA	<code>builder_main.cmd check_machine</code> — что с драйвером, потом <code>builder_main.cmd torch_cuda</code>
<code>N pictures end with _2-3.jpg, one is needed</code>	папка снимков не та, что называется <code>engine.cmd where</code> , или в ней два набора
<code>the work folder ... was made on other pictures</code>	рабочая папка принадлежит одному набору: назовите для этого набора другую
<code>... is marked already</code>	<code>seed</code> поверх разметки не пишет: возьмите <code>fix</code> или, намеренно, <code>set</code> — он пишет участок заново
<code>... is already there</code> от <code>train</code> или <code>join</code>	возьмите для сети новое имя
<code>... is made of nets itself: name its members instead</code>	<code>join</code> берёт одиночные сети, а не склеенные
<code>... holds the count of ...: a run is the count of one net</code>	назовите другой прогон ключом <code>--out</code> или не задавайте <code>--out</code>
<code>... is the count of the pictures of ..., the set now is ...</code>	прогон другого набора: считайте и выгружайте его с тем набором
<code>... was taught on every marked patch: there is nothing fresh to count</code>	<code>shaken</code> и <code>stability</code> нужны участки, на которых сеть не училась: разметьте проверочные и отложите их
<code>no counted picture has its .tab beside it</code>	слоям нужны файлы <code>.tab</code> снимков
<code>no net to count with: none is named in ... \models \counting.json and this copy of the program carries none in ... \config \nets or ... \system_core \nets</code>	сеть ещё не обучена и не положена в программу, ничего не сломано: «Сети ещё нет: первая сеть»
окно показывает старые числа	ПЕРЕЧИТАТЬ