Changelog

Contents

- 1.0.0 — 2026-10-08
- 0.6.0 — 2026-10-08
- 0.5.1 — 2026-10-08
- 0.5.0 — 2026-10-08
- 0.4.2 — 2026-10-08
- 0.4.1 — 2026-10-08
- 0.4.0 — 2026-10-08
- 0.3.0 — 2026-10-08
- 0.2.0 — 2026-10-05
- 0.1.0 — 2026-10-05

1.0.0 — 2026-10-08

The first release: a program of its own.

- **Its own Python.** The engine ran on the Python of another program beside it; now it has `runtime\` of its own: Python 3.12, PyTorch 2.8.0 - with CUDA 12.8 on an NVIDIA card whose driver is 570 or newer, for the processor otherwise - NumPy, OpenCV, Pillow and openpyxl, every version pinned exactly so that the count does not move when a library updates. The one thing put into Windows is the Microsoft Visual C++ Redistributable, when it is missing there. On the new runtime the check table of the counting net came out line for line as before.
- **SETUP in the window.** A new section: this machine (processor, memory, video cards, the NVIDIA driver and what follows from it), what is installed, and INSTALL WHAT IS NEEDED, which downloads and installs everything the count needs - 0.3 to 3 GB - and removes the downloads after. Three smaller buttons repair one part. A copy without its Python opens on this section by itself.
- **The builder and the cleanup.** `builder_main.cmd`: `check_machine`, `build_env`, `verify`, `install_all`, `torch_cuda`, `torch_cpu`, `clean_download`. `cleanup_project.cmd` brings

the folder down to the skeleton that is published: the code, the net the program counts with, the configs, the documents and the built window stay; everything that is downloaded goes.

- `doctor`, a tool of the engine: Python, PyTorch, the video card, the libraries and the net as one line; it needs no pictures.
- **Tests of the engine** on a drawn set, run with the checks: a new set is counted at once by the net of the program; a work folder is refused for the pictures of another set; every mark is in the layers once and at its place on the ground.
- Fixed: a count whose answers carried no ground coordinates (counted by an early version, or written by hand) gave empty layers for MapInfo without a word. The place of a mark is now worked out from the `.tab` of its picture.
- The user guide in Russian and English, the guide for an AI agent in both, and F1 opens the user guide in the language of the window.
- Checked the way it will be used: a copy cleaned to the skeleton, beside a drawn data folder, installed its engine by its own button in 2.4 minutes, said that everything is in place, and went through viewing, marking, counting, the report and the layers. Not checked: the folder dialogs themselves; the layers were never opened in MapInfo itself.
- **Nothing is downloaded without a question.** INSTALL WHAT IS NEEDED and the repair buttons ask first: what will come for this machine, how many gigabytes (about 3.4 GB with CUDA, about 0.7 GB for the processor), what goes into Windows, and that without the engine the program shows the pictures but does not count. A copy without its engine asks it by itself at the first start. The scenario `offer` of `--shots` presses the button and answers no.
- **The Microsoft Visual C++ runtime, first.** PyTorch cannot start without the Microsoft Visual C++ Redistributable (x64), and a freshly installed Windows may not have it. The first step of the installation looks for it and installs it with winget when it is missing (`install\Ensure-VCRedist.ps1`); Windows asks for the administrator's consent. It is the one thing the program installs into Windows. SETUP says so when the runtime is missing, and when winget is not there either it names the Microsoft download before anything starts.
- **The window is built without Python.** The translation table is checked by `Engine\Check-Texts.ps1` (PowerShell) instead of a Python script: `build_app` needs the .NET SDK and nothing else. The checks of the engine run where a Python is and are passed over with a word where none is.
- **An old NVIDIA card counts on the processor.** The PyTorch with CUDA carries code for NVIDIA cards from the GTX 10 series up; an older card was given that build all the same and the count would have stopped with "no kernel image is available". Now the machine check reads the generation of the card and installs PyTorch for the processor for it, SETUP says so, and the engine

itself passes such a card over. Checked with the generation of the card assumed and by tests of the rule, not on a card of that age.

- The menu of `builder_main.cmd`: every line fits a console line.
- **The installation for a machine without an NVIDIA card** was run for the first time: PyTorch for the processor in 1.2 minutes, 3.4 GB installed, the drawn picture counted on the processor by the tests.
- **The public release carries a net of its own.** `open-03` was taught from a random start on marks set by eye on free aerial photographs of three open state services - 2334 cars on 22 photographs - and on nothing else; it lies in `system_core\nets`. On photographs of those services it was not taught on it counted +2 % against 296 cars marked by eye, on satellite pictures of a fourth source it had never seen +1 % against 511. `config\counting_default.json` names several nets in the order they are tried: the first whose file is there counts.
- **A net that must not leave is kept as a key.** A net that must not leave with a public copy is kept the way a key is kept - `config\nets\<name>.credential.pt`: release tools that clear secrets out of `config` leave it out of a public archive, and a private copy carries it and counts a new set at once. A net anybody puts in lies in `system_core\nets\<name>.pt`; the program looks in the work folder, then in `config\nets`, then there. A copy without a net views and marks; the engine installs and answers (`doctor`: an empty `net`, not a fault), SETUP and COUNT say that there is no taught net, `verify` gives a warning, and the tests that count are skipped. It counts once a net is taught in the work folder or a net file is put into `system_core\nets`. `CARS_COUNTER_NETS` names one folder instead of the two of the program.
- **The program names no source of pictures.** With nothing chosen in SETTINGS the set that is counted is the one the work folder `Work` was made on, and in a new data folder the first folder of `Pictures` by name; before, a folder of a fixed name was the default.
- **The icon**: one car in the ring of the mark of the program, in the frame of a picture. `builder_main.cmd build_icon` draws it for every size from `Assets\app.svg` (the 16 px frame from `Assets\app-16.svg`, drawn on the pixel grid); `Start.exe`, the window and the title of the window carry it.
- **A net has one name, whatever its file is called.** A count by a net kept in `config\nets` (`<name>.credential.pt`) went into a run named after the file, and the window, which looks for the run of the net (`count_<name>`), showed the set as not counted. Seen on the first set that had no net of its own in its work folder. Now the run, the summary, the report and the list of nets all go by the bare name of the net; a test holds it.
- **A net is taught on several sets of pictures at once**: `net train --also PICTURES WORK ["pictures kept out"]` adds the marks of another set - another city, another source - to the ones of the set at hand. Nothing is copied or mixed: every set keeps its pictures and its marks in its own folders, a work folder is still refused for pictures it was not made on, and the net remembers

which patches of which set it has seen, so `stability` and `shaken` count only what is fresh for the set they are run for. A net taught on one set comes out as before. A count is still of one set.

- **Quieter outlines.** The outline of a button, a switch and an input field - the plain grey one and the coloured ones (pressed, under the pointer, the green, amber and orange buttons) - stands halfway between its former colour and the surface, in both themes. Cards, text and icons keep their colours.
- **Lessons of another source**: `net train --lessons 3` shows the squares also turned by any angle, a little larger or smaller, through haze, paler or more vivid and coarser - what differs from one source of pictures to another. The marks are moved with the picture and their map is drawn anew. The default stays `--lessons 2`: on the two checks of 2026-10-09 the new lessons neither helped nor harmed a net taught on three sources.
- **The screenshot of the window is made on free material**: an aerial photograph under CC BY 4.0, marked by eye, with its credit on the picture and in the README. The window takes it itself: with `AUDION_SHOT_ONLY` the one pass is in the language and the theme named by `AUDION_SHOT_LANG` and `AUDION_SHOT_THEME`, the viewer shows the piece named by `AUDION_SHOT_SPAN` and `AUDION_SHOT_AT`, and the language, the theme and the size the user had chosen are put back.

0.6.0 — 2026-10-08

Minor: what the program has learned comes with the program; every kind of data has its field.

- The owner's principle: the pictures and the reports are not the main thing - they can be copied again and the counts reproduced; the main things are the program, what it has learned, and its stability. The net the set is counted with lay in the data folder; a program moved without that folder could not count. `cars-22` now lies in the program (`system_core\nets`, named by `config\counting_default.json`). A net of the same name in the work folder is taken first, and `models\counting.json` of a work folder still names the net of that set when one was taught and accepted there.
- SETTINGS has a field for each kind of data and for the reports: PICTURES (the set that is counted - by default a folder of `Pictures` of the data folder, or any folder), WORK FOLDER (the marks, the counts, the tiles of the viewer, the nets taught there - by default `Work` of the data folder, or any folder), REPORTS (where the reports and the layers go; the export buttons open the choice of a folder there). The engine reads the first two from the same settings (`"Pictures"`, `"Work"`); `CARS_COUNTER_PICTURES` and `CARS_COUNTER_WORK` name them for one command.
- A work folder belongs to one set of pictures. It remembers the pictures it was made on (`set.json`), and every tool but `where` and `info` refuses it for other ones - the marks, the tiles

and the runs in it are found by the end of a picture's name alone, and another source of the same ground or another city has pictures that end the same. By default the first set has `Work` and any other set `Work-<set>` beside it, so a second set gets a work folder of its own without anyone choosing it. With this the run of a set is `count_<net>` again, in its own work folder: the count of the second set moved from the runs of the first set's work folder into its own, `Work-<set>`.

- A new set of pictures with an empty work folder is counted at once, by the net that came with the program. Tried on the second set in the window: 31 pictures, 13 402 marks, no mark and no tile of the first set shown.
- The window: the buttons of ABOUT open the pictures and the work folder; the dialogs of the report and of the layers open where REPORTS points.

0.5.1 — 2026-10-08

Patch: the data folder says what is brought and what is made, and a count belongs to one set of pictures.

- The data folder is laid out as the owner asked for both counters: `Pictures\` holds the sets as they were brought, a folder a source, and is only read; `Work\` holds everything made from them (`labels`, `models`, `runs`, `probe`). Before, the sets lay straight in the data folder beside `work`; a folder laid out so is still read as it is. After the move the engine's answer about the set is the same but for the paths (31 pictures, 317 patches, 688 cars), and the report and the layers of the run came out with the same numbers: 18 186 marks, 8 326 squares.
- Two sources of the same ground are two shootings of different days under the same file names, and the owner's rule is "one or the other is counted; a net may be taught on all". A count run now says which set it is of (`"source"` of `summary.json`). `net count` refuses to continue a run with the pictures of another set - they would be taken for counted ones, and the answers of one shooting given out as those of the other - and `report` refuses to draw a run on the pictures of another set. A run that does not say was counted before runs said it: the first set. The run of the set is named by the engine: `count_<net>` for the first set, `count_<net>_<set>` for any other, and the window asks the engine for the name.
- `CARS_COUNTER_PICTURES` names the set for one command. The set was changed until now by editing `config\settings.json`, which the window and every other command read: a measurement that ran beside such an edit took the wrong pictures once.
- Not done, and said in the documents: the tiles of the viewer and the marks belong to the first set. A second set is counted and reported from the command line; it is not to be opened in the window until tiles and marks are kept by set.

0.5.0 — 2026-10-08

Minor: the count as layers for MapInfo.

- The transport department asked for it: every picture has a MapInfo `.tab` that ties it to the local coordinate system, so the count can be laid on the map and studied by district. `report RUN --out FOLDER --formats mif [--cell 250]` writes three tables in the coordinates of the pictures (the `CoordSys` clause of their `.tab` is copied word for word): every mark of the program as a point (the picture, the sureness, the place in the picture); a grid of squares on the ground, 250 metres by default, with the marks of each, the cars from and to, the share of the square the pictures cover and the marks per square kilometre of that share
  - empty squares too, "no cars here" is an answer; and the outlines of the counted pictures with their totals. A text beside them says what the layers and the columns are.
- The format is MIF/MID, the interchange format of MapInfo itself: Table > Import makes an ordinary table of each. A native `.TAB` table is four binary files that only MapInfo or the GDAL library writes; the program carries neither, and a text file can be checked figure by figure. Read back on the set of 31 pictures: 18 186 points, the marks of the squares add up to the same, the covered area is 509.0 km^2 as the pictures', a mark lies 0.8 m from where the corners of its picture's `.tab` put it. Not opened in MapInfo itself: it is not on the machine the program is made on.
- Squares are laid by the ground and not by the pictures (a square is `floor(x / cell)`, `floor(y / cell)`), so the squares of another count - another day, another source of pictures - are the same squares and compare one to one.
- The density map is written too, as a picture - the owner, seeing it: "can this map be exported as well?" Twice: `..._растр_плотности.png` with a `.tab` beside it, which MapInfo opens as it is, without an import (the registration of a raster is plain text, the same kind as the pictures' own, so this one the program can write), a square coloured by its marks on the same grid as the layer of squares; and `..._карта_плотности.png`, the same map with its scale of colours and a caption, for reports and letters - on a light ground and, as `..._карта_плотности_тёмная.png`, on a dark one ("is there a dark version of the map too?"). The MIF/MID tables stay ("we keep MID/MIF too").
- The side of a square is chosen in the window by a row of buttons, 100, 250, 500 or 1000 m (the owner asked whether it should be a field or a slider: neither - a choice of a few round sizes is a row of buttons in every Audion window, and round sizes keep the squares of two exports the same squares). The coordinates fix where a mark lies; how coarsely marks are gathered into a density is a choice.
- The window: COUNT, the card REPORT, a third button LAYERS FOR MAPINFO. The check of the window presses it by its own path (`AUDION_SHOT_SCENARIO=count`: "layers ok").

0.4.2 — 2026-10-08

Patch: a count of pictures that were not counted yet stopped after the first of them.

- Since 0.4.0 `net count` told its progress with the number of the picture in the count - and the same name was used a few lines above for the place of an answer on the ground. After the first picture with answers the count ended with an error; the picture itself was written and kept, so a second start went on to the next one and ended the same way. The whole set had been counted before 0.4.0, and the button COUNT THE PICTURES was checked only on a set already counted (it was named as not checked) - so nothing showed it until a second set of pictures was counted on 2026-10-08. Fixed; a count of 31 pictures now runs to its end in one go, as it did before 0.4.0.
- `system_core\tried\other_source.py`: the record of a measurement, not a command - the counting net on the pictures of another source of the same ground (another folder), against the marks set on ours. The same ground is taken out of the other picture by the two .tab files and laid on our pixel grid.

0.4.1 — 2026-10-08

Patch: the data folder may lie in a common data folder of several counters.

- The engine looks for the data in `Audion Counters Data\Cars Counter` beside the project folder first, and in `Cars Counter` beside the project folder when that one is not there (`system_core\paths.py`). The owner keeps the data of both his counters in one folder now; no drive is written down as before, the set of folders still moves to another machine as it is, and `config\settings.json` (`"Data"`) and `CARS_COUNTER_DATA` still win. `engine.cmd where` says which folder was taken.
- The folder of the pictures of the other source has an English name now: the names of the folders of the data are English. No command reads it.

0.4.0 — 2026-10-08

Minor: the count and the report from the window. The owner's word: "everything as in Traffic Counter: buttons, export, a count one can see", and of the wording - "for transport engineers, not geeks".

- The window has the section COUNT. COUNT THE PICTURES counts every picture of the set that is not counted yet (the engine's `net count`; the map and the table get the number of a picture as soon as it is counted, STOP stops it). The result stands as large numbers: the marks of the program, the bounds of the number of cars (the marks less a quarter to the marks plus a half - a number of

the set is never told without them), the pictures counted, the area, the marks per square kilometre. Beside them the table by picture; a click on a row opens the picture in VIEW with its marks.

- The report, two buttons: REPORT ON ALL PICTURES and REPORT ON THIS PICTURE ask for a folder and write a Word document and an Excel sheet there (`report`, a new tool of the engine). The document: the figures, what the number means, the layout of the pictures with the count on each, the table, then a page a picture - the picture with a dot on every mark and eighteen marks taken at random, enlarged, so that a reader sees what the program took for a car. The sheet: the table by picture, the marks at every threshold, every mark with its place in the picture and on the ground (to lay over a map), and a last sheet "Служебное" with the net, the threshold and the folder of the count. The whole set of 31 pictures: 46 pages, 15 MB, 20 seconds.
- The words of the window's count and of the report are a reader's, not a programmer's: a "mark of the program", "cars from ... to ...", "picture"; the net, its threshold and the folder of the run are named only on the service sheet.
- `models\counting.json` of the data folder names the net the set is counted with and its threshold (`{"net": "cars-22", "least": 0.20}`); `info` reports it, `net count --weights auto` and the window take it. It is written by whoever accepted the net - never the newest file of the folder, where single nets and nets of four lie side by side.
- `net count --json` tells its progress as JSON lines (a picture begun, how far down it, a picture counted or kept); without the key it prints as before.
- `docx.py` is the writer of .docx files of Audion Traffic Counter 1.4.0, copied: no library is added, the parts of the file are written directly.

0.3.0 — 2026-10-08

Minor: several nets answering as one, lessons that show other strips of the mosaic, and a tool that shakes the picture. Made on the machine with the video card, where the work is minutes now.

- `net join --name NAME --of "NET NET ..."`: a file that holds several nets whole and answers with the mean of them. It is a net like any other - `count`, `stability`, `seed`, `overlay` take it by its name. Why: four nets taught on the same marks from four random starts counted -4, -11, -8 and -17 % on the check pictures at one threshold; groups of three of them - -20, -21, -21, -20 %. The part of the error that changes from one teaching to the next goes away.
- `net train --seed N`: the random start and the choice of squares. The seed is written into the net; the default 1 teaches as before.
- `net train --lessons 2` (the default now): a square is also shown as another strip of the mosaic would show it - other tones, softer, sharper, pressed as a worse JPEG. `--lessons 1` teaches as

the nets up to `cars-11` were taught. The recipe is written into the net.

- A new tool, `shaken NET LEAST`: the patches a net was not taught on are made lighter, darker, flatter, harsher, softer, sharper, tinted, pressed - twelve ways - and counted at one threshold. It says how far the count moves when nothing on the ground has changed.
- The first run on the video card (RTX 5070): the path through CUDA works as it was written on the laptop, the stability tables of nets 03 and 04 came out within one car, a teaching of 8000 steps takes six minutes, a whole picture 48 seconds.
- Checked: `stability`, `shaken` and a count of the whole set on a joined net; `shaken` on two nets. Not checked: that `--seed 1 --lessons 1` gives `cars-08` again number for number (the random draws are the same by the code; it was not taught a second time). The window was built anew for the version; nothing of it was changed.

0.2.0 — 2026-10-05

The window: the pictures and the marks are looked at and set by a click.

- The owner asked when the window is made and how a picture of 165 megapixels is to be looked at; his answer to the second: step over tiles along two axes instead of dragging a heavy picture. So the window (Avalonia 12, the shell of Audion Traffic Counter) shows the whole picture small and one piece large, and the piece moves in steps.
- Section PICTURES: the set laid out as it lies on the ground, with the marked patches and cars of each picture and the answers of a chosen run at a chosen threshold.
- Section VIEW: the piece in four sizes (120, 240, 480, 960 pixels), moved by the arrow keys, the buttons and a click on the whole picture; Page Up / Page Down through the marked patches; layers for the marks, the answers of a run and lightening.
- Marking in VIEW: a car, a truck, "cannot tell" and erasing by a click, an empty patch, a draft from the answers of a run, undo. It writes the files `engine.cmd patch` writes; a patch taken away is moved to `Work\labels\_removed`, not erased.
- New engine tools for the window, each printing JSON lines: `info` (the set in one line), `thumbs` (small pictures for the map), `pyramid` (tiles of a picture in five sizes).
- `info` reads the count of the probe (`count_3-2_cars-03`, written before runs took several pictures) and gives its number at the threshold 0.20 from its table.
- `builder_main.cmd` with `Engine\Build-App.ps1`, `Build-Launcher.ps1`, `Run-Tests.ps1`: the window is published into `App\` (self-contained), `Start.exe` starts it. No icon yet: the owner chooses it from samples.

- Checked on the laptop from the built `App\`, without a mouse (`--shots` with the scenario `view,mark`): a picture opened from the map, steps along both axes, the four sizes, a click on the whole picture, lightening; marks set, changed, erased and undone on a picture with no marks, the file read back in the engine's format, an empty patch written and removed (moved to `_removed`), nothing left behind; the sections looked at in Russian and English, dark and light.
- Checked on a copy of the project in another parent folder, beside a drawn data folder (`Source\Tools\make_check_set.py`): `engine.cmd where` finds the folders there, the window opens the picture, and TAKE THE NET'S ANSWERS takes exactly the answers it should (not the one on a car that has its point, not the one under the threshold, not the neighbour's).
- Not checked: the choice of a data folder by the dialog, the question before a patch is removed, anything on the machine with the video card.

0.1.0 — 2026-10-05

The first version: the probe of one day made into a project.

- The owner's question: can a model on a video card count the cars of a set of several dozen satellite pictures. His measure: an error that holds - 5, 10, 20 %, the same everywhere - not the exact number.
- The engine, run as `engine.cmd <tool>`: `patch` (patches marked by eye: show, seed, set, fix, snap, list), `net` (the net of car centres: train, check, count), `stability` (is the error the same from place to place), `answers` and `overlay` (what a net answered), `overview` and `montage` (pictures to choose and to look through patches), `geo` and `layout` (where the pictures lie on the ground, by their MapInfo `.tab`), `where`.
- The project is code and documents; the pictures, the marks, the nets and the runs lie in the folder `Cars Counter` beside it, found from the parent folder - no drive is written anywhere, so the pair of folders moves to another machine as it is. The owner's word: a separate project with its documents in it, to switch to the machine with the video card without a seam.
- `net count` writes every answer with its place in the picture and on the ground, takes several pictures or `all`, and goes on after a stop with the pictures it has not written.
- The net takes the video card when the machine has one. That path was written on a laptop without a card and has not been run on one.
- No Python of its own yet: `engine.cmd` runs the one of Audion Traffic Counter in the same parent folder.
- Checked on the laptop: `where`, `patch list` (115 patches, 515 cars), `stability cars-03` gives the numbers of the probe (on picture 2-4: 41 of 49 and 60 of 60 at the threshold 0.20),

`layout`, `geo`, a count of a region with ground coordinates, `answers`, `overview`, `patch show`, `montage`, `overlay`, a teaching of 40 steps. Not checked: a count of a whole picture and of the whole set from the project (the probe counted one picture with the same code), anything on a video card.