

Audion Cars Counter

[Русский](#) · [User guide](#)

Counts passenger cars on satellite pictures that are tied to the ground: how many stand on each picture and on the whole set, where each one is, and how dense they are by district. One button counts the set; the result is a number with its bounds, a Word document, an Excel sheet and layers for MapInfo. The count runs on your machine: the pictures are not sent anywhere.

Start **Start.exe**, open the SETUP section and press **INSTALL WHAT IS NEEDED**: the program says what it will download for your machine and asks, then puts its own Python and PyTorch into its own folder - 0.7 to 3.4 GB of downloads, about 7 GB installed on a machine with an NVIDIA card. The one thing that goes into Windows is the Microsoft Visual C++ Redistributable PyTorch needs, when it is not there (by winget, with the administrator's consent).

What is installed, and what the gigabytes buy. The button puts the same things on any machine - the program's own Python, PyTorch and a few libraries - but PyTorch comes in two builds, and the build decides both the size and the speed.

	No NVIDIA card	NVIDIA card, driver 570 or newer
PyTorch build	for the processor	with CUDA
Downloaded	about 0.7 GB	about 3.4 GB
Takes in the program folder	about 3.5 GB	about 7 GB
Counting a picture of 165 megapixels	about 20 minutes	about a minute
Counting a set of 31 pictures, about 500 km ²	about 11 hours (reckoned)	20 minutes
Teaching one net, 8000 steps	about 2 h 40 min (reckoned: 2000 steps took 40 minutes)	6 minutes

	No NVIDIA card	NVIDIA card, driver 570 or newer
Teaching the four nets that are joined into the one that counts	about 11 hours (reckoned)	10 minutes, the four side by side

The times are the author's: a laptop processor with six threads on the left, an RTX 5070 on the right. An NVIDIA card older than the GTX 10 series is treated as no NVIDIA card: this PyTorch does not count on it.

Where the size comes from. Nearly all of it is PyTorch. The build with CUDA carries inside it NVIDIA's libraries for computing on a video card: about 7 GB, almost the whole folder. The build for the processor is about 3 GB of the 3.5. The rest is Python itself, OpenCV (0.1 GB) and small libraries. There is no TensorRT in this program. The downloaded installation files are removed once everything is installed.

What it is for. Counting and teaching are the same arithmetic of the net, and a video card does it 25 to 30 times faster. The count itself does not depend on the build: on a check of several hundred cars the two differed by a car or two.

The program counts with a taught net. The public release carries one, `open-03`: it was taught on free aerial photographs of three open state services and on nothing else, and on pictures it had not seen it counted within a few percent of marks set by eye. Your pictures are another source than the ones it has seen: before a count is believed, examine the net on a few patches you have marked. A net of your own is taught on your own pictures together with an AI agent, or a net file you were given is put into the program - see [A net for your own pictures](#). Once a net is there, the set is counted by one button; a net that lies in the program itself counts every new set at once.

Section	What it does
PICTURES	The map of the set: every picture in its place, with what is marked and counted on it
VIEW	One picture: a piece of it large, the whole of it small beside, the marks of the program over it; points are set by a click to check or to teach a net
COUNT	Counts every picture of the set: the marks of the program, the bounds of the number of cars, the area, the density; the report as Word and Excel files and the layers for MapInfo
SETUP	The machine, what is installed, and a button that installs exactly what it lacks
SETTINGS	A field for every kind of data: the pictures, the work folder, the reports

Section	What it does
ABOUT	The version, the keys, the folders

It takes a folder of `.jpg` pictures, each with its MapInfo raster registration `.tab` beside it; the name of a file ends with the column and the row of the picture in the set (`..._2-3.jpg`). About 0.3 m to a pixel is what the author's net was taught on. Windows 10 or 11, 64-bit.

A **mark of the program** is a place a small net took for a car: for every pixel the net answers how likely the centre of a car is there. Every mark has its place on the ground, from the picture's `.tab`. A mark is not yet a car - at this resolution the eye cannot tell many dark blobs either - so the number is always told with its bounds: cars from "the marks less a quarter" to "the marks plus a half". Trucks and buses are not counted. The pictures are only read; what is made from them lies in a **work folder**, one for each set of pictures.

What the number is worth. As much as the net it is counted with. The numbers here are those of the author's net, which is not part of the public release: what a net taught this way gave in the author's work, to expect of a net you teach - not a property of the copy you download. That net was taught and checked on satellite pictures of one town from one source. On three pictures it had never seen, 511 cars are marked by eye for certain and it put 509 marks: -2 % on a plant with a large open car park, +12 % on residential ground, 0 % in an old town under long shadows - where a quarter of the cars is missed and as many marks are extra, and they cancel. A picture that only looks lighter or softer moved the count by up to 4 % on open car parks and by up to 16 % in the shadowed old town. On another imagery source of the same town, of a lower effective resolution, that net found about a third of the cars, and ground of a kind a net never saw can give false marks by the hundred. A new source or a new kind of ground has to be examined first and is usually taught for.

The layers for MapInfo are the marks as points, a grid of squares (100, 250, 500 or 1000 m) with marks and density, and the outlines of the pictures - as MIF/MID in the coordinate system of the pictures' `.tab` - plus a density map: as a picture with a raster `.tab` of its own, and with a scale and a caption, light and dark, for reports. The files were checked by reading them back; they have never been opened in MapInfo itself.

The program is made to be worked on together with an AI agent, Claude Code or Codex: open its folder in the agent and ask it to read `Docs\AI_AGENT_GUIDE_EN.md` ([Russian](#)). The agent marks patches on your pictures, teaches a net on them, judges it and counts; when a net is already there, it first examines that net on your pictures - see [Working with Claude and Codex](#). Teaching a net has no button in the window: it is a command of the engine.

The interface follows the language of the system (Russian or English; RU | EN at the top right). The window is C# on Avalonia 12 (.NET 10); the counting engine is Python 3.12: PyTorch, NumPy, OpenCV, Pillow, openpyxl. Third-party libraries are downloaded by your button and keep their own licences. Icons: Lucide. Font: JetBrains Mono.

The screenshot of the window shows a free aerial photograph, not the author's working pictures: Beeldmateriaal Nederland via PDOK, CC BY 4.0. The marks on it were set by eye.

The net that comes with the program, `open-03`, was taught on aerial photographs of open state services: Beeldmateriaal Nederland via PDOK (CC BY 4.0); orthophotos of the Republic of Estonia Land and Spatial Development Board, taken from the service 2026-10-09 (licence of open data of the Estonian Land Board, <https://geoportaal.maaamet.ee/opendata-licence>); Geobasis NRW (Datenlizenz Deutschland - Zero 2.0). The photographs themselves are not part of the program; what the net was taught on and how it was examined is in `system_core\nets\README.md`.