

AI That Knows When Not to Act: Trustworthy Perception at the Edge

A deployed perception model can stay confident even when the scene contains something it has never seen, or when its own camera input degrades. Capsa adds a risk signal alongside the prediction, so an edge system can tell a familiar input from an unfamiliar one — and decide whether to proceed, verify, reroute, or escalate rather than acting blindly.

Running live on the i.MX 8M Plus NPU · full INT8 · TensorFlow Lite 2.19 with the NXP VX delegate (eIQ) · Linux BSP LF6.18.2
· Probe scoring on two Cortex-A53 cores

The result at a glance

97%

of segmentation failures caught (Demo 2)

81% fewer missed than the model's own confidence

5% false-alarm threshold · PIDNet-S
i.MX 8M Plus · INT8

77%

fewer missed unfamiliar objects than the model's own confidence (Demo 1)

MUAD-OOD synthetic driving scenes
threshold flags 5% of clear-scene pixels
i.MX 8M Plus · INT8

< 2%

added NPU time

risk computed in a single pass, on-device

Demo 2 · plus ~4 ms on two Arm cores

These are live demonstrations on public driving data (A2D2) and a scene we built and captured ourselves; the measured figures also use the public and simulated datasets listed under Methodology. They illustrate the behavior of the risk signal; they are not production benchmarks, yield figures, or safety certifications.

Why perception fails in a vehicle

In a vehicle, a perception model that is confidently wrong is a safety liability, and the hardest failures are the ones the model never saw coming. Out-of-distribution conditions are among the most common causes of perception degradation in the field: an unfamiliar object in the road, or a camera degraded by fog, rain, glare, or a fouled lens.

The usual answer is redundancy: more sensors, sensor fusion, a fallback path. That is necessary, but it has a known gap: fusion can only recombine what the sensors provide, and against a common-cause failure — the kind where the model is confidently wrong about something it was never trained on — duplicating sensors or models often fails the same way at the same moment, and the system cannot tell which input to trust.

A risk signal that flags from within the model addresses exactly that blind spot. It recognizes the unfamiliar object and the failing camera before the system acts on them, and turns that into a decision:

- **Proceed:** act on the prediction (familiar input)
- **Verify:** flag for review before acting
- **Reroute:** fall back to another model or sensor
- **Escalate:** hand control to the driver

Because Capsa runs live on the edge, that judgment happens on the device, inside the control loop, in the fraction of a second it has.

Two demonstrations

We show two demos. Each targets a different way perception goes wrong, and — as the later section explains — each is driven by a different kind of uncertainty.

Demo 1: Foreign object detection

A camera watches a road scene, and the model flags anything it does not recognize. Place a car on the road and the model is comfortable: it has seen plenty of cars, so the scene stays calm. Place an animal in the road — something it was never trained to expect — and the region lights up red. The model is signaling, in real time, that this input is out of distribution and its read of it should not be trusted.

Figure 1 — out-of-distribution detection



Camera view — familiar vehicles, one unfamiliar animal



Capsa risk map — the animal lights up

Out-of-distribution detection. The model flags an unfamiliar object on the road in real time; familiar objects such as cars do not raise the signal.

The live flag. The risk signal is computed for every pixel and shown as a heatmap over the camera feed, so an unfamiliar object lights up where it sits. The flag is computed over a region of the image. When the region's average risk passes set multiples of its clear-scene level ($1.25\times$ for elevated, $1.5\times$ for high risk), the region is flagged and the verdict becomes escalate to a human. In this demo the region is the full frame. In a real deployment, the region, the thresholds and the resulting actions are user-defined operating points.

Measured performance. Beyond the live binary flag, we evaluated the underlying signal on labeled data:

77%

fewer missed unfamiliar objects

than the model's own confidence: 13 vs 56 of 59 objects missed, with the heatmap threshold set so 5% of clear-scene pixels are flagged.

+5.5 ms

added NPU inference time

+10%.

PIDNet-S driving segmentation with Neo • NXP i.MX 8M Plus • INT8 • MUAD-OOD (synthetic driving scenes), 59 unfamiliar objects.

On real photos (RoadAnomaly, SMIYC), Neo ranks unfamiliar-object pixels better than the model's own confidence (AUROC 0.71 vs 0.59 and 0.72 vs 0.53).

Demo 2: Degraded-input detection

Here the question is not what is in the scene, but whether the camera itself can be trusted. A failing sensor, a fogged or wet lens, or glare all degrade the image the model sees, and this demo flags that degraded input before the system acts on it. The demo runs a real driving clip, and the user controls the image quality with a blur slider. At first the picture is clean and the risk signal stays low — the verdict reads trusted. As the input is blurred, the risk signal climbs: once it crosses the warning threshold the verdict flips to degraded. The model recognizes that its own input can no longer be relied on.

Figure 2 — degraded-input detection



Clean — trusted

Blurred — degraded

Degraded-input detection. As the camera input is blurred, the epistemic uncertainty rises; once it crosses the threshold the verdict flips from trusted to degraded. Driving frames from A2D2 (Geyer et al., 2020, arXiv:2004.06320), used under CC BY-ND 4.0.

Measured performance.

97%

of segmentation failures caught

81% fewer missed than the model's own confidence (3 vs 16 of 105), at a 5% false-alarm threshold.

< 2%

added NPU inference time

plus ~4 ms of scoring on two Arm cores.

Across 9 degraded conditions, Probe missed 57% fewer degraded frames than the model's own confidence (28 vs 65 of 315). With heavy blur it missed 69% fewer (4 vs 13 of 40). On real fog, the model's own confidence missed 4 frames and Probe missed 16.

PIDNet-S driving perception · NXP i.MX 8M Plus NPU · INT8. 380 frames (A2D2 clean and blurred, CARLA and MUAD simulated weather), 105 failures. Failure = frame mIoU below half the clear-weather average.

On the threshold. The degraded warning triggers at the 95th percentile of clean-frame scores (5% false alarms on clean frames, by design); high risk at 1.5× the clean average. In a real deployment these are user-defined operating points chosen to match the application's tolerance; the brief does not claim a single correct value.

Two kinds of uncertainty

The two demos look similar on the surface — both flag an input the model should not act on — but they catch fundamentally different problems, and Capsa uses a different signal for each.

Vacuitic uncertainty — what the model cannot understand due to latent feature imbalance: inputs that fall into regions the training data barely covered are poorly represented and therefore uncertain. An animal in the road, essentially never seen in training, is such an input. This is the signal behind **Demo 1 (foreign object detection)**.

Epistemic uncertainty — what the model does not know due to limited knowledge or data. It is high for inputs that drift away from what the model was trained on — which is exactly what blur does. This is the signal behind **Demo 2 (degraded-input detection)**.

Note: Capsa also estimates aleatoric uncertainty (irreducible noise or genuine ambiguity in the data itself), although not shown in these demos.

The pairing is deliberate: vacuitic uncertainty for the unfamiliar object, epistemic uncertainty for the degraded camera input. Matching the right signal to each failure mode is part of what Capsa does.

How Capsa works

Capsa wraps an existing trained model and adds uncertainty estimation without retraining and without changing the training pipeline. The conversion is a single operation applied to the model you already have; the model then outputs a risk estimate alongside every prediction. Because the signal is produced in the same forward pass, it is cheap enough to run live on the edge — no ensembles, no multiple training runs, no cloud round-trip.

The two demos use two different Capsa wrappers, matched to the two uncertainty types:

- **Neo**: produces the vacuitic uncertainty signal used in Demo 1 (foreign object detection). Neo learns a signature of the training data and measures how closely each new input aligns with it; the degree of divergence from that signature is the signal, flagging novel, out-of-distribution inputs. It is designed to run efficiently enough for real-time inference on resource-constrained devices.
- **Probe**: produces the epistemic uncertainty signal used in Demo 2 (degraded-input detection).

Numerics

Table 1 — Cost

Measurement	Unwrapped	With Capsa	Difference
Demo 1 (Neo) NPU inference time	52.8 ms	58.4 ms	+5.5 ms, +10%
Demo 1 deployed artifact size	8.07 MB	8.26 MB	+0.19 MB, +2%
Demo 2 (Probe) NPU inference time	23.7 ms	24.0 ms	+0.3 ms, +1.3%
Demo 2 Arm-core scoring	—	4.2 ms	2 × Cortex-A53
Demo 2 deployed artifact size	8.06 MB	8.70 MB	+0.64 MB, +8%

Latency measured on the i.MX 8M Plus NPU, full INT8 (TensorFlow Lite 2.19, NXP VX delegate), as the mean of 100 runs after warm-up, in 3 alternating rounds, against each demo's own unwrapped export.

Table 2 — Quality (AUROC, with 95% CI where shown)

Measurement	Model's own conf.	With Capsa	Difference
Demo 2 (Probe), heavy blur vs. clean (A2D2)	0.73 [0.60, 0.86]	0.96 [0.91, 1.00]	+0.23 [+0.10, +0.37]
Demo 1 (Neo), RoadAnomaly pixels	0.59	0.71	+0.13 [+0.004, +0.25]
Demo 1 (Neo), SMIYC RoadAnomaly21	0.53	0.72	+0.18 [+0.04, +0.35]
Demo 1 (Neo), MUAD-OOD	0.80	0.89	+0.09 [+0.06, +0.13]
Demo 1 (Neo), obstacle sets*	0.95	0.95—0.97	not significant

*RoadObstacle21 and street obstacles. Demo 1 RoadAnomaly row is scored on unfamiliar-object pixels. Differences in both tables are computed before rounding.

Methodology

Datasets. Demo 2 uses the public A2D2 autonomous driving dataset. Demo 1 uses a physical road scene (a model diorama) that we built and captured ourselves with a live webcam; there is no third-party dataset behind the demo footage. The measured figures for Demo 1 come from MUAD-OOD, RoadAnomaly, SMIYC and the obstacle sets, and those for Demo 2 from A2D2 (clean and blurred) and CARLA and MUAD simulated weather, as labelled where they appear.

Models and signals. Both demos run PIDNet-S. Demo 1 computes vacuitic uncertainty via the Neo wrapper; Demo 2 computes epistemic uncertainty via the Probe wrapper.

Thresholds. Demo 1 flags a region (here, the full frame) when its average risk exceeds 1.25× (elevated) or 1.5× (high risk) its clear-scene level. Demo 2 warns at the 95th percentile of clean frames and escalates at 1.5× the clean average. In both cases, a real deployment would set these as user-defined operating points.

Stack and measurement. Full INT8, TensorFlow Lite 2.19 with the NXP VX delegate (eIQ), Linux BSP LF6.18.2, i.MX 8M Plus; Probe scoring on two Cortex-A53 cores. Latency is the mean of 100 runs after warm-up, in 3 alternating rounds, against a matched unwrapped export.

Limitations

- **Live demonstrations, not benchmarks.** These demos show the risk signal behaving correctly on specific inputs. They are not evaluations on a production dataset and do not establish field accuracy, false-alarm rates, or PPM figures.
- **Demo 1 is a staged scene.** The road diorama is something we built; it is not real driving data, and results on it do not transfer unmeasured to real roads.
- **Demo 1's flag thresholds are illustrative.** They are set relative to clear scenes, not calibrated against labeled examples; the measured figures above are for the per-pixel signal.
- **Demo 2's thresholds are calibrated on clean frames from the evaluated sets.** Other cameras need their own calibration; the operating point is a user choice, not a validated setting.
- **Not a safety certification.** Capsa decides when an input should not be trusted or should be escalated; it does not authorize any action or certify a safety function.

Come see it live, or bring us your own case

We are at NXP Technology Days in Detroit, October 28—29, 2026, at the Tech Lab. Stop by to try both demos and talk through how this could apply to your own models and hardware.

Contact info@themisai.io • themisai.io/nxp-tech-day

NXP, i.MX and eIQ are trademarks of NXP Semiconductors. This brief describes independent work by Themis AI and does not imply endorsement by NXP.