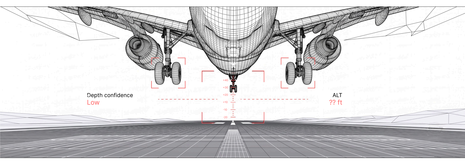
WHY A CAMERA-BASED LANDING SYSTEM PASSES EVERY TEST AND STILL CAN'T BE CERTIFIED TO FLY



A camera-based landing system uses a neural network to read a forward-facing camera feed, find the runway, and work out where the aircraft is on final approach. In testing it performs well, locating the runway accurately under the conditions it was trained on.

Then it has to be certified, and the accuracy figure that would settle the question in most industries settles very little. This is the wall machine learning runs into in aviation. It is worth understanding properly, because the wall is built from the same property that makes these systems risky in service.

THE QUESTION REGULATORS ARE REALLY ASKING

Airborne software has been certified for decades under a framework that assigns every function a level of scrutiny based on how bad its failure would be. DO-178C covers software, DO-254 covers hardware, and ARP4754B covers the system as a whole.

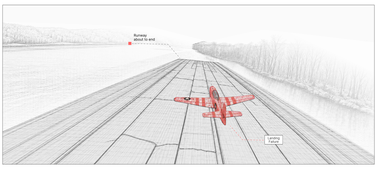
The principle underneath is simple. Every requirement has to trace to behaviour you can verify. You say what the software must do, you build exactly that, and you show line by line that it does that and nothing else. If you cannot prove it, you cannot fly it.

A neural network does not fit this. Its behaviour is learned from data rather than specified and then implemented. No line of code states the rule for recognising a runway. The behaviour sits in millions of weights fitted to examples, so the traditional question of whether the implementation matches the specification has no answer, because there is no specification in the form the framework expects.

An accuracy figure does not fill that gap. It reports how the system did on the cases it was shown, which is the kind of evidence the framework was built to distrust.

Regulators recognised this. Rather than forcing neural networks into a model that does not suit them, EASA developed a separate assurance approach for machine learning, worked out through the published CoDANN studies. What that process asks for is the interesting part. It does not ask for better accuracy. It asks an applicant to establish the bounds of the model's competence. That means showing how well the model generalises to unseen data and what its uncertainty looks like on inputs outside its training distribution. It also means identifying the edge and corner cases the application involves, and showing how robust the model stays when inputs drift away from what it learned.

The question regulators are asking is not how often the model is right. It is where its competence ends, and what happens when an input falls outside it.



THE ONE ANSWER THE MODEL CANNOT GIVE

The landing system was trained on a finite set of approaches, with particular runways, lighting, weather and camera conditions. On final approach in service it will meet conditions that set never covered. There might be an unusual runway marking, low sun straight into the lens, snow partly covering the threshold, or haze that flattens contrast. None of these are exotic. They are the ordinary variation of real flying, and they are precisely the edge and corner cases the assurance process wants characterised.

When the model meets one, it gives no sign that it has left familiar ground. It resolves the unfamiliar image to the nearest representation it knows and returns a runway position in the same form, often with the same score, as it returns on a clear day. A confident correct fix on a known approach and a confident wrong fix on an approach the model never properly learned look identical at the output. The model has no internal sense that one of them was a guess.

So the applicant is stuck. To certify the system they have to demonstrate where its reasoning holds and where it does not, and show that out-of-distribution conditions are detected and handled. Examined only through its outputs, the model offers no way to do that. Its competence boundary is real and invisible.

TURNING A COMPETENCE BOUNDARY INTO SUBMITTABLE EVIDENCE

This is the gap Vision Studio addresses, and the fit with what regulators ask for is unusually close. The assurance process wants an applicant to characterise the model's competence and detect operation outside it. That is a description of what the tool does.

Vision Studio modifies the model so it outputs the embeddings of a chosen layer alongside its predictions. Those embeddings are mapped into a view of the decision space, which separates regions where the model's reasoning is well supported from regions where inputs are sparse, unfamiliar or ambiguous and reliability falls away.

For the landing system, that map is more than a diagnostic. It answers the assurance questions item by item. The trusted regions describe how far the model generalises across approach conditions. The ambiguous regions are the edge and corner cases the process requires identified. The boundary between them defines the envelope against which out-of-distribution operation is measured. Where the applicant previously had an accuracy figure and no way to answer the real question, they now have evidence drawn from the model's reasoning rather than inferred from test scores.

Saliency analysis supports the same case. It shows what the model actually attended to when it located a runway. That distinguishes a system reading runway markings and geometry from one keying on surrounding terrain or lighting patterns that happened to correlate during training. For a regulator asking why a system behaves as it does, that is a more direct answer than a score.

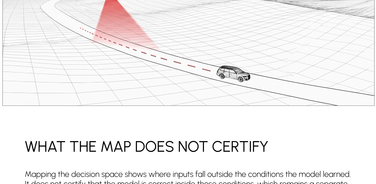
ONE CHARACTERISATION, USED TWICE

Vision Studio then exports a watchdog, a lightweight module that runs alongside the model in flight and checks where each frame falls relative to the map.

What matters for certification is that the watchdog is specified against the same boundary the evidence describes. The thing monitored in the air is the envelope documented in the submission. The offline evidence and the runtime monitor are one characterisation used twice, which is the kind of traceability between what was claimed and what is enforced that the framework is built around.

In service, when the system meets the low sun or the obscured threshold it cannot handle, the failure stops being silent. The watchdog flags that the guidance has left its characterised envelope, and control reverts to the pilot rather than to a confident wrong number. Reverting to the human when a model exits its valid conditions is a recognised means of compliance. A monitor grounded in the model's own decision space is an auditable way to implement it, because the condition that triggers reversion is the same boundary the regulator reviewed.

Uncertainty that keeps recurring under one class of condition is worth treating as a pattern rather than a series of separate events. A run of flags concentrated on low-sun approaches at a particular airfield says something a single flag does not. It points either at a condition the model needs more training on, or at an operating limitation worth documenting.



WHAT THE MAP DOES NOT CERTIFY

Mapping the decision space shows where inputs fall outside the conditions the model learned. It does not certify that the model is correct inside those conditions, which remains a separate question answered by testing and analysis.

It also does not remove the need for the rest of the assurance process. Requirements, traceability, verification and the full DO-178C apparatus still apply to the software around the model. What the map addresses is the specific problem of characterising a learned component's competence, which is the part the traditional framework has no vocabulary for.

Every reversion to the pilot carries an operational cost too. A boundary drawn conservatively hands back control more often than necessary, and one drawn loosely defeats the purpose. Where it sits is a safety case argument rather than a tuning decision.

FROM A GOOD TEST RESULT TO AN APPROVAL

Without this, an applicant is offering a regulator an accuracy score against a requirement that score does not address, and certification stalls because nobody can show where the model stops being trustworthy. The number proves the model is usually right on the conditions it was tested against. Certification is asking about the conditions it was not.

Seeing where the model's competence ends turns an unanswerable question into evidence. The applicant can describe the reliable envelope as a submitted artefact. They can show that departures from it are detected by a monitor built against that same envelope, and that the system reverts to safe control when its own reasoning no longer holds. That is the difference between a model with a good test result and a system that can be trusted to fly.

The scenario described here is representative, grounded in published aviation guidance on machine-learning assurance, including EASA's CoDANN work and the W-shaped learning assurance process, the certification frameworks, Design Assurance Levels, and assurance requirements reflect the state of the field. Squint Vision Studio's capabilities, decision-space mapping, and exportable watchdog modules are described as designed.