Edge AI Capability Has to Be Proven in the Application

By: Nick Weir PhD, VP of Mission Engineering

The useful question about an edge AI model is whether it can complete the work an operator needs at the tactical edge, on the hardware and with the data actually available. Parameter counts and leaderboard scores help narrow the candidates, but they don't show whether the model will serve the mission need.

Disconnected Is the Baseline, an earlier Command Paper on tactical edge AI, argued that disconnected operation has to be the design baseline. That creates a practical model selection problem, because a body-worn system, a laptop, and a transportable server have very different compute budgets. What work can each support, and how does a program know when a smaller model is sufficient?

The answer depends on how much the application asks the model to figure out. A model following a well-defined maintenance procedure faces a different problem from one deconflicting supply requests across several subordinate units. Benchmarks serve neither problem well. AI applications now have to call the right agent plugin and reason over data from several sources, and a leaderboard that scores one problem in isolation can't show whether a model handles that combination. A model becomes suitable for an edge mission when the complete application repeatedly demonstrates that it can finish the intended work within the deployment constraints.

A Correct Tool Call Is Only One Step

An assistant that formats every call correctly can still apply the right procedure to the wrong equipment.

A tool call, also called a function call, is how a model uses software. It lets an AI application do more than output text. The model might search a technical manual, read equipment status, or prepare a work order. Supporting function calls is useful, but a workflow requires the model to do more than select a function and format its arguments.

Consider a maintenance assistant. It needs to identify the equipment variant, find the applicable procedure, interpret the reported symptoms, and determine which check

comes next. After receiving a result, it has to update its understanding. If the reading contradicts the diagnosis, it needs to investigate further or ask for help. Retrieving a plausible procedure is only the beginning.

Errors also have different consequences. An unnecessary search costs time. Applying an action to the wrong equipment can invalidate the rest of the workflow. Evaluation needs to distinguish between them, including cases where the system reports success after changing something it should have left alone. That is why the evaluation has to run on the real workflow from the mission application, and eventually on real mission data, before anyone can judge how the system will perform.

Application Design Changes What a Small Model Can Do

A small model looks inadequate when the application leaves every decision to it.

Application design sets how difficult the model's job is, and the team building the application controls most of that difficulty. An application can expose a handful of clearly differentiated tools, supply the relevant equipment record, and maintain task state outside the conversation. Code can enforce prerequisites and check whether an identifier refers to the equipment being serviced. The model can then focus on interpreting information and choosing among valid next steps.

TinyAgent, a 2024 research project from UC Berkeley, demonstrates the value of this approach. Researchers combined task-specific training with tool retrieval to make 1.1B- and 7B-parameter models competitive with GPT-4-Turbo, a much larger model, on their specialized function-calling evaluation. That result establishes useful capability within the tested application. Extending it to unfamiliar missions requires new evidence.

Why Retrieval Agents Fail found the same pattern: model capability, retrieval tools, and instructions can each constrain performance. Buying a larger model won't make an inaccessible maintenance record appear in the search results.

The same discipline supports human authority. Agentic AI Security Is an Architectural Decision describes bounded workflow steps and validated tool inputs. Those boundaries also make the task easier to evaluate. Correctly formatted arguments can still contain the wrong values, though. Independent checks and approval requirements remain necessary where actions carry consequences.

There is a limit to how much structure can compensate for model capability. A procedure can enumerate known faults; it cannot anticipate every combination of

symptoms. Giving a small model more turns to reason and retry also consumes time and power without ensuring that it recognizes the underlying problem. The evaluation needs to reveal when additional reasoning helps and when the system is simply repeating an unsuccessful approach.

Model Size Buys Different Amounts of Flexibility

Each step up in model size puts more hardware on the operator, so the job has to justify it.

Model size buys flexibility in steps. Models of roughly 4B parameters, often quantized to reduce memory requirements, are candidates for body-worn or laptop-scale applications with narrowly defined jobs. Useful starting points include extracting fields, retrieving a specific procedure, and selecting branches in a known troubleshooting process. The application needs to carry much of the procedural knowledge. Unexpected observations and unfamiliar tool combinations should trigger a defined fallback, including returning the decision to the operator.

The 20B to 30B range provides more room for applications that combine documents with equipment status, use several tools, and adapt an investigation as evidence arrives. It is a reasonable starting point for local assistants whose tasks require some flexibility within a bounded domain. Whether the additional capability justifies the hardware depends on the actual workflow. Google's Gemma 4 results show a substantial improvement from its small edge models to larger variants on Tau2, a stateful tool-use benchmark (42.2% for the E4B edge model against 76.9% for the 31B model), although those are developer-reported results rather than mission acceptance tests.

Systems able to host roughly 60B to 150B parameters offer candidates for less structured work, including ambiguous diagnosis and synthesis across conflicting sources. The reason to carry that additional capacity is the margin it may provide when conditions depart from the examples anticipated by the application designers. Buyers should require evidence of that margin. A larger model still needs accurate data, appropriate authority, and verification of its actions.

These ranges are starting points for experiments. Public evidence is concentrated in software, customer service, and simulated tool environments; it does not establish universal thresholds for defense workflows.

Even the parameter number needs qualification. Gemma 4's 26B A4B mixture-of-experts model has about 25.2B total parameters but activates about 3.8B per token. That reduces active computation while retaining a much larger weight storage requirement. Hardware planning must account for resident weights, working memory, and the rest of the application. An active parameter count alone cannot establish whether a system fits on a body-worn device.

Evaluate the Job on the System That Will Ship

A demo on a connected workstation says little about a quantized model running on a battery.

Evaluating an edge system starts with the same foundation as any agent evaluation. AI Agent Evaluation: Building an Evaluation Platform That Scales makes the case for reproducible datasets and metrics tied to operational work. For edge model selection, the test also needs to include the starting conditions, available tools, and authority under which the application operates.

Define success in terms that can be checked. For infrastructure remediation, inspect service health and configuration after the run. Check that unrelated services were preserved and required approval was obtained. For maintenance assistance, assess whether the procedure matched the equipment and whether missing information caused an appropriate stop. The agent's statement that it completed the task is insufficient evidence.

Then run the scenarios repeatedly and introduce realistic failures. Return stale readings, deny a permission, or interrupt connectivity after an action was submitted but before its result arrived. Check whether the system distinguishes an unconfirmed action from a failed one before trying again. A duplicate write can be a more serious failure than an unfinished task.

There are two useful comparisons here. Hold the application fixed when asking whether a new model can replace the current one. Allow the tools, prompts, and workflow structure to be adapted when asking what capability each model can support. Keep the mission scenarios and acceptance criteria fixed in both cases. Otherwise, an application designed around a large model can make a small model look inadequate simply because its interface was never designed for it.

The deployment configuration belongs in the test too. Quantization changes, runtime updates, and context handling can alter behavior. Measure completion time and energy

for the whole job, including retries and verification. A reduction in energy per generated token is useful only if the application still completes the work within its battery and time budget.

Disconnection must be tested separately. Ben Van Roo's The Denied Column argues that average connected performance can conceal weakness under denial. A local model still needs local documents and working tools. It also needs to preserve task state and reconcile changes after reconnection. Additional model capacity cannot supply a missing service.

Keep the Acceptance Standard as Models Improve

Better small models are coming, and a program has to decide how much to plan around them.

The research points toward better task-specific training and more efficient model architectures as ways to extend what edge hardware can support. Over the next year, those are reasons to re-run the comparisons. They are a poor basis for promising that a particular size class will become sufficient for an untested mission.

The useful evidence would be smaller systems completing unfamiliar combinations of tasks, recovering from misleading tool results, and preserving authorization boundaries across repeated runs. Those improvements could let an organization carry less hardware or support more flexible work with the hardware it already has. Each needs to be measured in the application.

For Centurion by Legion Intelligence, a deployable edge system for disconnected environments, that means selecting models against the local system's intended jobs. Legion Packs organize agents and tools around those jobs, with people reviewing and approving the work. A model change should be tested against those requirements before it reaches the operator.

Leaders buying edge AI should require a demonstration of the intended workflow on the proposed hardware, including the conditions under which it fails or returns a decision to the operator. Select the smallest system that repeatedly meets those requirements. Carry additional capacity when the evidence shows that the mission needs it.

References

- Erdogan, L. E., Lee, N., Jha, S., et al. TinyAgent: Function Calling at the Edge. UC Berkeley and ICSI, EMNLP 2024 (System Demonstrations).
- Google DeepMind. Gemma 4 model card. Parameter counts for 26B A4B and Tau2 tool-use results.

Continue the series

Follow Legion Intelligence on LinkedIn: linkedin.com/company/legion-intel

Explore all Command Papers: legionintel.com/command-papers

Ready to talk: legionintel.com/contact