

Robots That Take Instructions

Running vision-language-action models at the edge, with the safety controller still in charge

Fireball Industries · September 30, 2026 · 24 minute read

The kitting cell was justified on one part family. Eighteen months later it sees forty SKUs, three of them in nearly identical black housings, and the totes arrive from the AMR in whatever orientation the upstream operator left them. Every new SKU means a programmer, a vision retrain, new taught points, and a week of hand-feeding. The AMR next to it has the same problem in a different shape: it follows mapped routes perfectly until a pallet sits in the aisle, then it stops, raises a fault, and waits for someone to walk over.

Vision-language-action models are the first credible attempt to close that gap with learning instead of programming. A VLA takes camera images and a plain-language instruction ("put the blue connector housings in tote 3") and outputs robot actions directly. The published results are real, and so are the limits: success rates well short of what a production cell needs, sensitivity to lighting and camera placement, and compute that has to sit close to the robot. This paper covers what these models are, where they sit relative to the safety-rated controller, what the latency budget allows, how to collect the plant's own training data, what to do about Wi-Fi, and how to evaluate a cell before trusting an autonomy claim.

Start with your own cell

Answer these for one cell or AMR route.

1. How many interventions per shift does the cell need today (re-teaches, vision faults, manual picks, AMR "blocked path" stops), and who logs them?
2. How many SKUs or part variants does it handle, how many were added in the last year, and what did each one cost in programming time and downtime?
3. Which safety functions protect people around this cell today (protective stop, speed and separation monitoring, power and force limiting, zone limiting), and are they implemented in safety-rated hardware or in application code?

4. Where would a model run if you had one: on the robot, in a cabinet at the cell, in the server room, or in someone's cloud account? What is the network path, and what happens to the robot when that path drops for two seconds?
5. Do you have synchronized camera video, joint states, and outcome labels for this cell, even for one week? Could you collect a hundred demonstrations of the hard task without stopping production?
6. If a vendor tells you their model succeeds 90% of the time, how would you check that number on your parts, under your lighting, before you sign?
7. Does your fleet manager speak VDA 5050 or the MassRobotics interoperability standard, or does every brand of AMR have its own island?

What a VLA model is, and what it is not

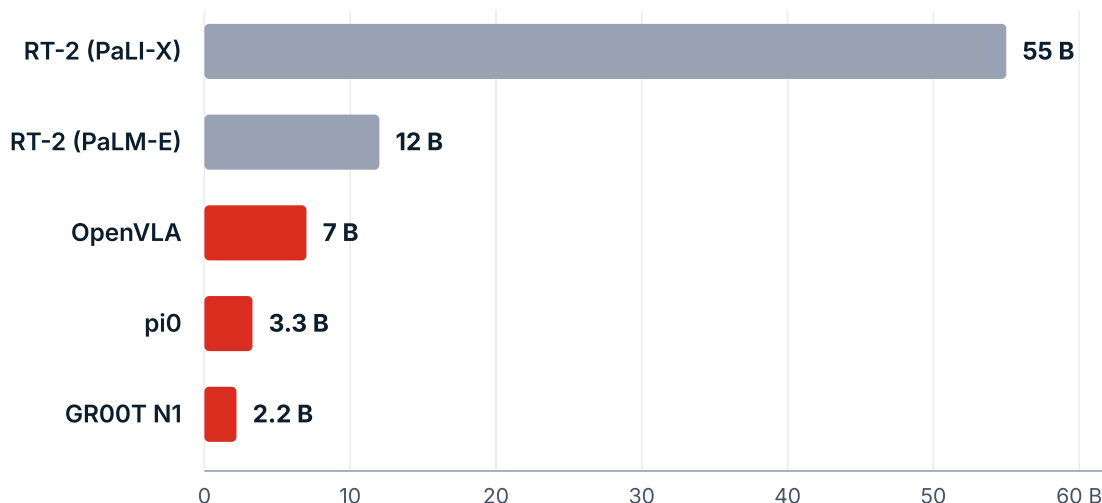
A vision-language-action model starts from a vision-language model trained on internet images and text, then is trained further on robot demonstrations so that it outputs motor commands. Google DeepMind's RT-2 paper (July 2023) introduced the term; across roughly 6,000 evaluation trials it handled objects and instructions absent from its robot data [1]. The project page lists versions built on 12-billion and 55-billion parameter backbones [2].

The open models that followed are smaller and more practical to host:

1. **OpenVLA** (Kim et al., June 2024): 7 billion parameters, trained on 970,000 robot demonstrations from the Open X-Embodiment collection, and reported 16.5 percentage points higher task success than the 55-billion parameter RT-2-X across 29 tasks [3].
2. **pi0** (Physical Intelligence, October 2024): 3.3 billion parameters, a 3-billion parameter vision-language backbone plus a 300-million parameter "action expert" that uses flow matching to produce chunks of 50 actions, at up to 50 Hz for dexterous tasks. Pre-trained on more than 10,000 hours of robot data [4]; the follow-on pi0.5 cleaned rooms in homes it had never seen [5].
3. **GROOT N1** (NVIDIA research, March 2025): 2.2 billion parameters in a dual-system design, a vision-language module running at 10 Hz and a diffusion transformer producing motor actions at 120 Hz. Sampling a chunk of 16 actions took 63.9 ms on an L40 GPU [6].
4. **SmoVLA** (Hugging Face, June 2025): runs on consumer GPUs or even CPUs [7].
5. **Gemini Robotics On-Device** (Google DeepMind, June 2025): a VLA that runs entirely on the robot, adapts to new tasks with 50 to 100 demonstrations, with low-level safety-critical controllers executing its actions [8].

The training data is public too: Open X-Embodiment pooled 22 robot types from 21 institutions [9], and DROID added 76,000 trajectories collected in 564 scenes [10].

Parameter count of published VLA models



Source: Brohan et al. 2023 (RT-2 project page); Kim et al. 2024; Black et al. 2024; NVIDIA 2025.
Red: open-weight models small enough for a single local GPU.

Figure 1. Published parameter counts for five VLA models. Source: RT-2 project page [2], Kim et al. [3], Black et al. [4], NVIDIA [6]. Red marks open-weight models in the size range a single local GPU can serve.

What the papers say about their own limits

- 1. Reliability.** OpenVLA's authors state the model typically achieves under 90% success on the tasks they tested [11]. A kitting cell that fails one pick in ten needs a person standing beside it.
- 2. New motions require new data.** RT-2's authors note that web knowledge improved semantic understanding but did not give the robot any physical skill it had not seen in robot demonstrations [12]. A VLA can recognize a part it has never been shown; it will not invent a grasp strategy for a part geometry outside its training data.
- 3. Fine-tuning is expected.** The OpenVLA-OFT work states that VLAs struggle with novel robot setups and require fine-tuning to perform well [13]. OpenVLA's own LoRA fine-tuning used 10 to 150 demonstrations per task and one A100 GPU for 10 to 15 hours per task [11].
- 4. Compute and speed.** RT-2's authors flagged real-time inference as a likely bottleneck [12]; OpenVLA cannot support 50 Hz bimanual setups [11].
- 5. Adversarial inputs.** Small printed patches in the camera view cut VLA task success by up to 100% in one 2024 study [14]; another jailbroke language-model-controlled robots, often with 100% attack success [15].

The practical reading: a VLA is a perception-and-motion-proposal layer that copes with variety better than taught points. It is not a safety function, and without plant-specific data and evaluation it does not

run a cell unattended at production reliability.

What needs to be measured and connected

A VLA cell has more inputs than a taught cell, and they must share one clock for the training data to be usable.

SIGNAL OR SOURCE	WHERE IT COMES FROM	WHY IT MATTERS
Wrist and scene camera frames	GigE or USB3 industrial cameras at the cell	Primary policy input and training record; fixed mounts matter more than resolution.
Joint positions, velocities, gripper state	Robot controller (vendor API or fieldbus)	Policy input and demonstration label.
Commanded vs. actual trajectory	Robot controller	Shows proposals clipped by limits.
Safety function status	Safety PLC or robot safety controller (protective stop, speed reduction, zone, monitored standstill)	Each trip is evaluation data and a hazard log entry.
Cell PLC handshakes	Cell PLC tags (part present, fixture clamped, door closed, mode)	The gate that permits the policy to run.
Task instruction and operator ID	HMI or tablet at the cell	Language input and who issued it.
Outcome label	Downstream check: scale, vision verify, scanner, operator button	Per-attempt result; no label, no success rate.
AMR pose, state, battery, errors	Fleet manager via VDA 5050 or MassRobotics messages	Coordinates arrivals; shows where units stall.
Network quality at the cell	Access point and client statistics (RSSI, retries, roam events)	Explains stops before anyone blames the model.
Model version and configuration	Edge node deployment record	Ties outcomes to exact weights.

The robot controller and safety PLC stay the systems of record; the edge node reads them through integration on their existing interfaces and tags. The outcome label is the row most plants skip, and the one that decides whether the project can prove anything.

The split between the learned policy and the safety-rated controller

The rule: the policy proposes motion, the robot controller executes it, and the safety functions bound it regardless.

The safety standards fit this structure. ISO 10218-1:2025 covers the industrial robot itself as partly completed machinery, and ISO 10218-2:2025 covers the robot application and cell through design, integration, commissioning, operation, and maintenance [16][17]. The 2025 revision made functional safety requirements explicit, folded the collaborative-robot content of ISO/TS 15066 into both parts, replaced “collaborative robot” with “collaborative application,” renamed “safety-rated monitored stop” to “monitored standstill,” and added cybersecurity requirements [18]. ANSI and A3 adopted the revision as R15.06-2025 in September 2025; A3 reports that the number of defined safety functions grew to roughly 36 [19]. ISO/TS 15066:2016 remains current, with a revision under development [20].

For mobile robots, ANSI/A3 R15.08-1 defines industrial mobile robots, R15.08-2 (October 2023) covers integration of IMR systems into applications and requires integrators to reduce risks to an acceptable level through risk assessment [21], and R15.08-3-2026 (September 2026) puts requirements on end users: conduct and document a risk assessment, maintain the risk reduction measures it identifies, train people, and document changes to the robot, application, or environment after deployment [22].

None of these documents treats a neural network as a safety function, and the VLA literature does not claim one should [8]. In practice:

1. Protective stop, emergency stop, speed and separation monitoring, power and force limiting, and zone limits stay in safety-rated hardware and the robot’s safety controller, configured and validated as they would be for a taught program.
2. The policy’s output enters the robot controller as a trajectory request, inside joint, Cartesian, and speed limits the controller enforces.
3. A software check on the edge node (workspace bounds, velocity, confidence, a cell-state gate from the PLC) rejects bad chunks. It is useful and not safety-rated; the risk assessment cannot take credit for it.
4. New model weights are a change to the application, documented like a new robot program [22].

DIVISION OF LABOR

What each layer is trusted to do

Learned policy (VLA)

- Find the part among mixed SKUs
- Turn an instruction into a task
- Propose grasp and motion chunks
- Recover from small variations
- Improve with the cell's own data

Safety-rated controller

- Protective stop and e-stop
- Speed and separation monitoring
- Force and power limiting
- Zone and axis limiting
- Monitored standstill

Figure 2. The division of labor. Left: what a VLA is good at. Right: the safety functions in ISO 10218-1/-2:2025 and R15.06-2025 that remain in safety-rated hardware regardless of the policy [18][19].

Latency: why inference sits at the cell

The robot controller closes its own servo loop; nothing here touches that. The VLA sets the rate at which new motion intentions arrive, which decides whether the robot reacts to where the part is or where it was a quarter-second ago. RT-2 at 55 billion parameters ran at 1 to 3 Hz from a multi-TPU cloud service queried over the network [12]. OpenVLA runs at about 6 Hz on one RTX 4090 [11]. OpenVLA-OFT, using parallel decoding and action chunking, raised throughput 26 to 43 times and ran a bimanual robot at 25 Hz [13].

Action chunking is what makes these models workable: the model predicts a short sequence of future actions, the controller executes them, and the next chunk is computed meanwhile. That tolerates inference time only up to the length of the chunk. If a 16-action chunk at 120 Hz covers about 133 ms of motion, then a round trip that adds 100 ms of network delay and jitter eats most of the margin, and a single Wi-Fi roam or WAN hiccup exhausts it. The robot then either stops (safe, but the cell is down) or executes a stale chunk (safe within its bounds, but wrong).

For comparison, 3GPP's industrial requirements set a 40 to 500 ms transfer interval for mobile robots, with communication service availability above 99.9999% [23], a bar a shared corporate network or an internet path to a cloud region does not meet.

Published timing numbers that set the budget

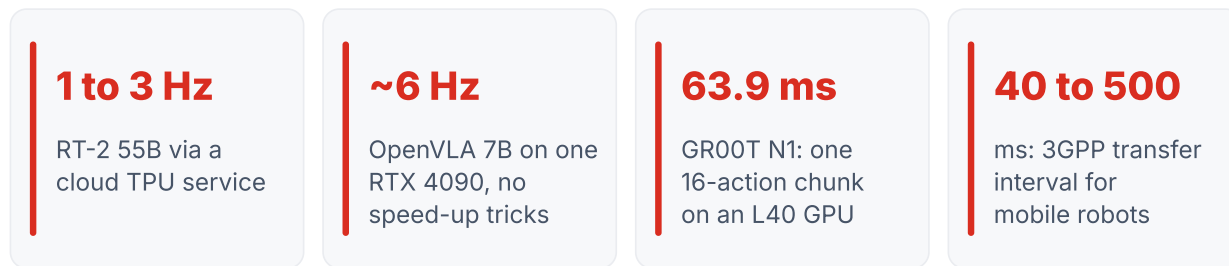
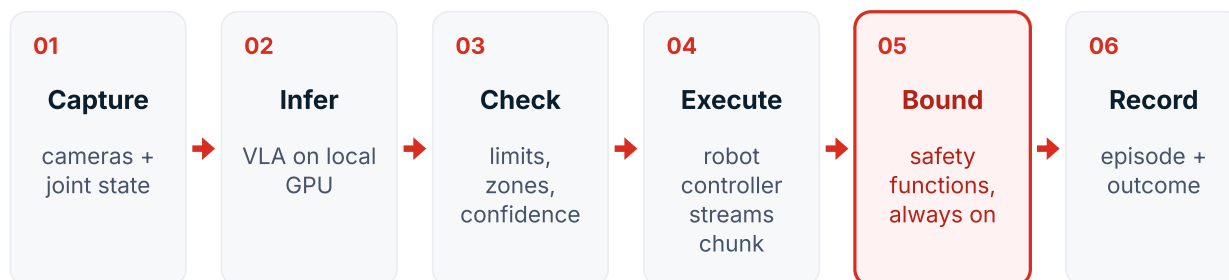


Figure 3. Published timing numbers. Sources: Brohan et al. [12], Kim et al. [11], NVIDIA [6], 3GPP TS 22.104 [23].

So put the GPU at the cell, on the same switch as the cameras and the robot controller, and keep the WAN out of the action loop. Training and analytics can live anywhere. Physical Intelligence's open release ships a policy server that streams actions to the robot over a websocket [24]; in a plant, that server belongs a few meters of copper from the robot.

CONTROL SPLIT

One cycle of a learned manipulation skill



Steps 1 to 3 and 6 run on the edge node; step 4 on the robot controller; step 5 in safety-rated hardware that does not depend on steps 1 to 3.

Figure 4. One cycle of a learned manipulation skill. Capture, inference, checking, and recording run on the edge node; execution runs on the robot controller; safety functions bound the motion independently of the first three steps.

On EmberNet. An EmberNode at the cell runs the policy server as a container beside the recorder, an MQTT broker, and the cell's HMI, orchestrated by k3s, so a crashed inference process does not take the node or the other apps down. EmberRTOS can switch on a real-time kernel; on a Raspberry Pi 5 it measured a 23 μ s worst-case wake-up latency over a 24-hour soak, with the stress load on the housekeeping core, against the roughly 100 to 200 μ s that industrial control loops need. The robot controller keeps its own servo loop and safety functions unchanged.

The site Wi-Fi problem

AMRs live on Wi-Fi, and many AMR “robot problems” are roaming problems. A robot moving between access points must reassociate and, on 802.1X, reauthenticate. With access points tuned for laptops, it sees gaps during roams, retries near racking, and dead spots behind the stretch-wrapper.

The fleet standards assume this. VDA 5050 expects wireless links with lost messages, uses best-effort QoS 0 on most topics, and publishes robot state on events and at least every 30 seconds [25]; MassRobotics status goes at most once a second [26]. Both are built for supervisory coordination at those rates. Three rules follow:

- 1. Inference rides on the robot.** A mobile manipulator running a VLA carries its own compute. Google DeepMind’s on-device model was released for exactly this reason: latency-sensitive work in places with intermittent or zero connectivity [8].
- 2. Wi-Fi carries orders, state, logs, and model updates.** Missions, VDA 5050 orders, telemetry, recorded episodes, and new model versions move over the wireless link, tolerant of loss and delay.
- 3. Survey at robot height.** Log RSSI, retries, and roam events along the route and fix coverage before blaming navigation or the model.

Fleet interoperability

Plants that buy AMRs usually end up with several brands and fleet managers. Two open specifications address this:

1. **VDA 5050** (VDA and VDMA, current version 3.0.0) defines the interface between a master fleet control and mobile robots over MQTT with JSON messages (order, instantActions, state, connection, factsheet, and others), and explicitly leaves out safety, traffic logic, and cybersecurity [25].
2. **MassRobotics AMR Interoperability Standard** (version 1.0, 2021) lets robots from different vendors share identity, location, speed, health, and availability over WebSockets with JSON. It is not to be relied on for safety [26].

For a VLA cell, the fleet layer reports when an AMR has arrived with a tote, which becomes the PLC handshake that lets the arm's policy start. An instruction such as "kit order 4471" becomes a VDA 5050 order to the AMR and a language prompt to the arm, both logged.

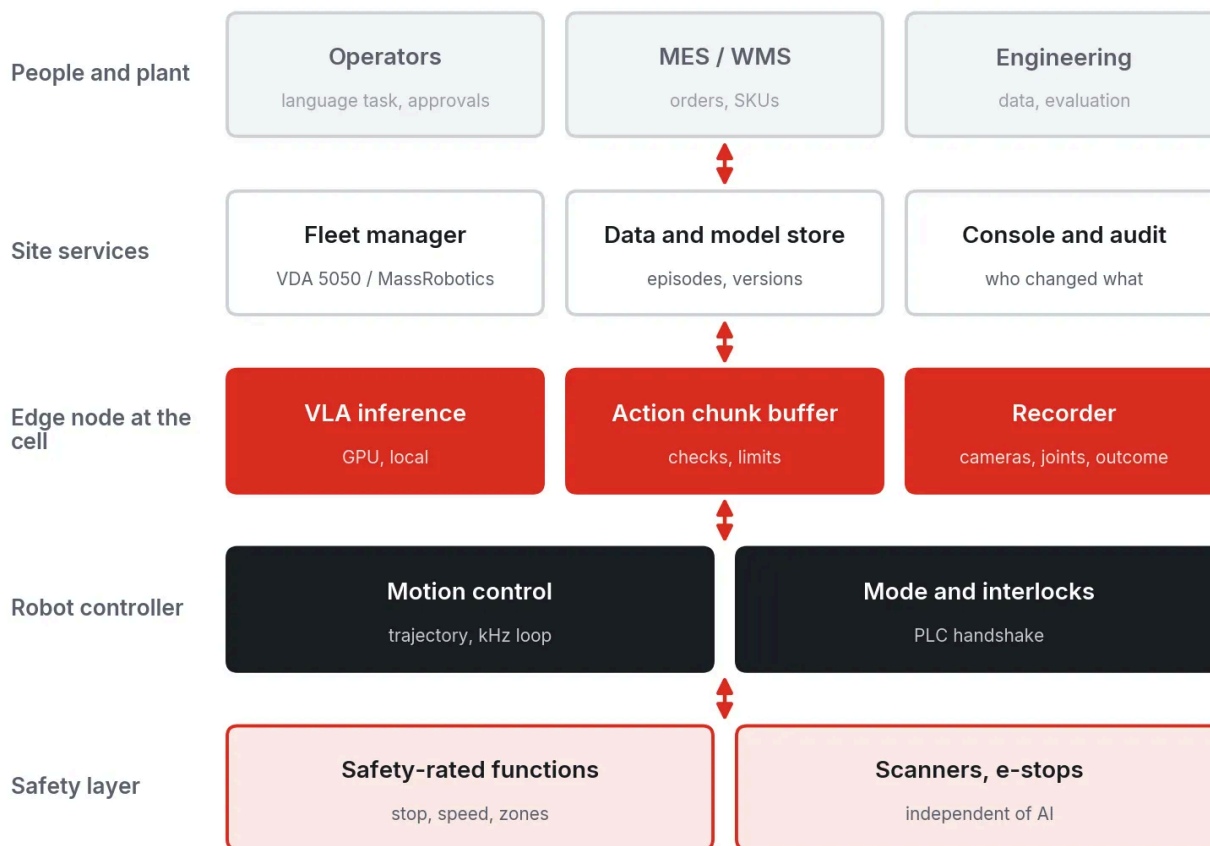
On EmberNet. EmberNet is a platform; a VDA 5050 master control, a MassRobotics receiver, an MQTT broker, and the plant's own adapters all run as containers on the same nodes, side by side with Ignition Edge or Node-RED for the cell HMI. Connectors are not limited, so a mixed fleet is integration work on each vendor's interface rather than a reason to pick one brand.

Reference architecture

Five layers, and the boundaries matter more than the products. The safety layer (scanners, light curtains, e-stops, the robot's safety controller) has no dependency on the edge node or the network. The robot controller and cell PLC keep trajectory execution, limits, and interlocks as they run today. The edge node at the cell holds inference, the chunk checker, the episode recorder, and the approved model version. Site services hold the fleet manager, dataset and model store, training compute, and the deployment record. People and plant systems issue instructions and orders and own evaluation.

REFERENCE ARCHITECTURE

Where the learned policy sits in a robot cell



The policy proposes motion; the robot controller executes it; the safety functions bound it regardless of what the policy proposes.

Figure 5. Reference architecture for a VLA-driven cell, from the safety layer up to people and plant systems.

Network segmentation follows the same lines. ISA/IEC 62443 organizes control systems into zones and conduits [27]. A sensible zoning puts the safety network in its own zone with read-only status to the edge node at most, the robot controller and cell PLC in a cell zone, and the edge node as the single conduit to site services.

On EmberNet. An EmberNode goes in front of the robot controller and cell PLC and microsegments them with a firewall built into EmberRTOS, so the edge node becomes the cell's conduit without changes to the existing controllers. Remote support reaches the node through EmberFLUX: outbound connections on 443 only, no inbound ports at the plant, no VPN, encrypted traffic, and devices reached by identity.

Walking the work, cheapest steps first

1. Baseline the cell before any model

Log every intervention for three weeks: who, what, how long, which SKU. Count re-teach hours per new SKU. Add a cheap outcome check at the cell exit (a scale, a barcode verify, an operator button). This shows whether the problem is variety, presentation, or network, and some cells get fixed here with a better tote insert.

2. Instrument and record

Mount cameras rigidly with fixed exposure and focus. Record video, joint and gripper state, PLC handshakes, and safety events on one clock. Hugging Face's open LeRobot library stores synchronized MP4 video plus Parquet state and action files and supports pi0, SmoVLA, and GROOT policies [28]; an open format keeps the plant's data portable between models and vendors.

3. Collect demonstrations on your own cell

Research datasets were built by teleoperation, and DROID's authors note the logistical, safety, and labor cost [10]. Plan for it:

1. Pick one task with clear success criteria, such as picking a mixed-SKU tote into a kit tray.
2. Collect 50 to 150 demonstrations covering the real variation: all the SKUs, the bad orientations, the lighting at 3 a.m. That range matches what OpenVLA and Gemini Robotics On-Device reported for fine-tuning [11][8].
3. Hold out a set of SKUs and conditions the model never trains on. That set is the honest test.
4. Teleoperate inside the cell's normal safeguarding, in a mode the risk assessment covers.

4. Fine-tune and bench-test

OpenVLA's LoRA recipe needed one A100 for 10 to 15 hours per task; openpi lists more than 22.5 GB of GPU memory for pi0 LoRA fine-tuning and more than 8 GB for inference [11][24]. Score the held-out set: attempts, successes, failures by type, safety trips, cycle time. Simulation can screen candidates (SIMPLER tracked real performance closely [29]); acceptance happens on the real cell.

5. Shadow, then supervise

In shadow, the policy proposes chunks, the checker scores them, and the existing program keeps running. Supervised, the policy drives while an operator approves each cycle and every intervention is logged. Go unattended only when held-out success meets the number the business case needs.

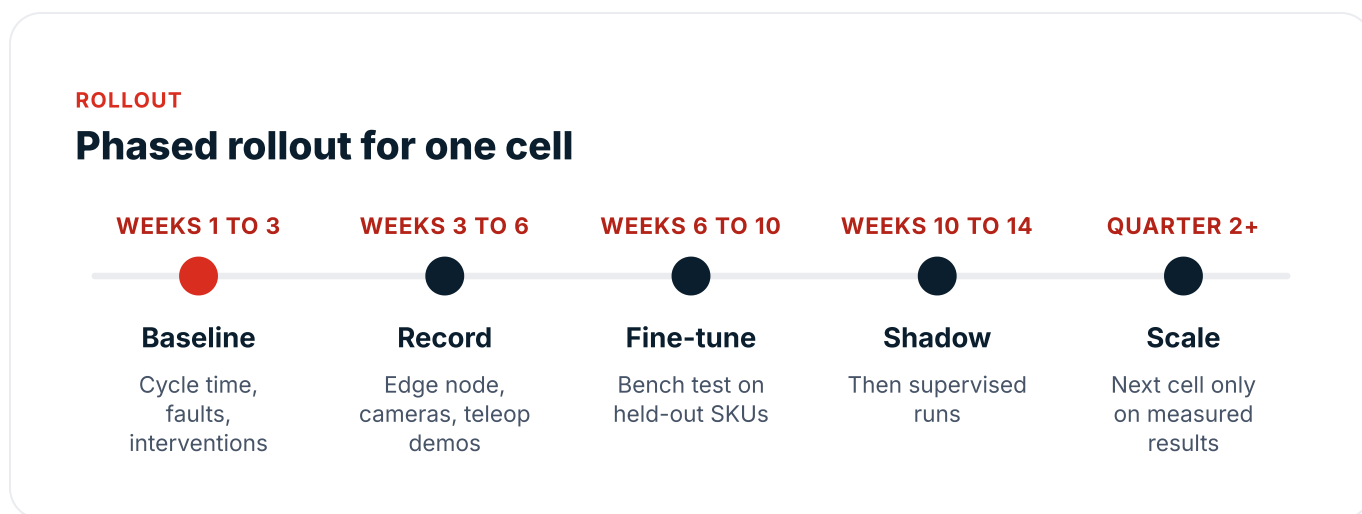


Figure 6. Phased rollout for a single cell, from baseline to scaling by measured results.

On EmberNet. Each model version deploys to the cell node as a container from EmberCORE, which keeps the audit trail of who deployed what and when, and a version that misbehaves rolls back from the same console. That record is the change log R15.08-3 expects end users to keep when the application changes.

Where these projects go wrong

- 1. Trusting the demo video.** Published success rates come from the authors' setups [11]. Ask for success on a held-out set from your cell, with the trial count.

- 2. Putting the model in the safety case.** A confidence threshold or learned obstacle detector is not a safety function under ISO 10218 or R15.08.
- 3. Cloud in the loop.** A WAN with variable latency feeds stale chunks to the robot or stops it [12].
- 4. No outcome label.** Without per-attempt results, nobody can show improvement or catch a regression after a model update.
- 5. Camera drift.** A bumped wrist camera or afternoon skylight drops success. Fixed mounts and a daily reference image catch it.
- 6. Unmanaged model versions.** A checkpoint goes to the cell from a laptop, and nobody knows which weights produced Tuesday's scrap.
- 7. Ignoring the input surface.** Patches and crafted instructions have defeated published systems [14] [15]. Restrict who can instruct, to a task vocabulary, and log every prompt with operator ID.
- 8. Wi-Fi and model blamed for each other.** Log network statistics with every episode.

Security and compliance

The site's safety and security programs own compliance. The architecture's job is to make the required controls straightforward to implement and to evidence.

Machine safety

- 1. Risk assessment.** ISO 10218-2:2025 and R15.08-2 require risk assessment of the application, including foreseeable misuse [17][21]. Assume the policy can command any motion the controller limits allow.
- 2. Safety functions in safety-rated hardware.** Protective stop, monitored standstill, speed and separation monitoring, power and force limiting, and zone limits stay where they are validated today [18][19].
- 3. Non-routine work.** OSHA notes many robot accidents happen during programming, maintenance, testing, and setup, and has no robot-specific standard [30]. Demonstration collection and supervised runs are that kind of work; write procedures for them.
- 4. Change management.** Model versions, camera moves, and vocabulary changes go through the change record R15.08-3-2026 expects [22].

Cybersecurity

- 1. Robot standard.** ISO 10218:2025 added cybersecurity requirements [18][19]; VDA 5050 leaves cybersecurity to the implementer [25].

- 2. Zones and conduits.** Apply ISA/IEC 62443-3-2 risk assessment (asset-owner duties in 62443-2-1, service providers in 62443-2-4) and keep the edge node as the one conduit [27].
- 3. No inbound access.** Remote support should not open ports into the cell zone.
- 4. Model integrity.** Treat weights like PLC programs: versioned, signed off, deployed from one place, with rollback.
- 5. AI risk governance.** NIST's voluntary AI Risk Management Framework (Govern, Map, Measure, Manage) is a workable structure for evaluation and change control [31].

Phased rollout and what to do Monday

The rollout in Figure 6 takes one cell from baseline to supervised operation in roughly a quarter and gates the second cell on the first cell's measured results. Teams that skip the held-out test set find out on second shift.

What to do Monday.

- 1.** Pick the cell or AMR route with the most interventions per shift. Start logging every intervention with SKU, cause, and duration.
- 2.** Add an outcome check at the exit of the cell, even if it is a button an operator presses.
- 3.** Pull the risk assessment for the cell and list which safety functions are in safety-rated hardware. If any protective measure lives only in application code, fix that first, independent of any AI project.
- 4.** Walk the AMR route with a laptop logging RSSI and roam events at robot height. Mark every spot where the robot has stopped in the last month.
- 5.** Ask every robot or AMR vendor in your pipeline three questions: where does inference run, what is the success rate on a held-out set from a customer cell and over how many trials, and what happens to the robot when the network drops for two seconds.

None of this requires a model or a capital request: one cell, three weeks of honest logging, and the safety case on the table first.

Fireball Industries is EmberNet's master integrator. Fireball's engineers design, build, and support this work end to end: surveying the cell and the network, keeping the existing robot controllers and safety systems in place, standing up EmberNodes at the cell for inference and data collection, integrating mixed AMR fleets, and running the evaluation that decides whether a learned policy has earned its place on the line.

Sources

1. Brohan, A. et al. (Google DeepMind), "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control," arXiv, July 28, 2023. <https://arxiv.org/abs/2307.15818>
2. Google DeepMind, "RT-2: Vision-Language-Action Models" project page, 2023. <https://robotics-transformer2.github.io/>
3. Kim, M. J. et al., "OpenVLA: An Open-Source Vision-Language-Action Model," arXiv, June 13, 2024 (rev. September 5, 2024). <https://arxiv.org/abs/2406.09246>
4. Black, K. et al. (Physical Intelligence), "pi0: A Vision-Language-Action Flow Model for General Robot Control," arXiv, October 31, 2024. <https://arxiv.org/html/2410.24164>
5. Physical Intelligence, "pi0.5: a Vision-Language-Action Model with Open-World Generalization," arXiv, April 22, 2025. <https://arxiv.org/abs/2504.16054>
6. NVIDIA, "GR00T N1: An Open Foundation Model for Generalist Humanoid Robots," arXiv, March 18, 2025. <https://arxiv.org/html/2503.14734>
7. Shukor, M. et al. (Hugging Face), "SmolVLA: A Vision-Language-Action Model for Affordable and Efficient Robotics," arXiv, June 2, 2025. <https://arxiv.org/abs/2506.01844>
8. Google DeepMind, "Gemini Robotics On-Device brings AI to local robotic devices," June 24, 2025. <https://deepmind.google/discover/blog/gemini-robotics-on-device-brings-ai-to-local-robotic-devices/>
9. Open X-Embodiment Collaboration, "Open X-Embodiment: Robotic Learning Datasets and RT-X Models," arXiv, October 13, 2023 (rev. May 14, 2025). <https://arxiv.org/abs/2310.08864>
10. DROID Dataset Team, "DROID: A Large-Scale In-The-Wild Robot Manipulation Dataset," arXiv, March 19, 2024. <https://arxiv.org/abs/2403.12945>
11. Kim, M. J. et al., "OpenVLA" full text (inference speed, memory, fine-tuning, limitations), 2024. <https://arxiv.org/html/2406.09246>
12. Brohan, A. et al., "RT-2" full text (deployment and limitations), 2023. <https://arxiv.org/pdf/2307.15818>
13. Kim, M. J., Finn, C., Liang, P., "Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success," arXiv, February 27, 2025. <https://arxiv.org/html/2502.19645>
14. Wang, T. et al., "Exploring the Adversarial Vulnerabilities of Vision-Language-Action Models in Robotics," arXiv, November 18, 2024 (rev. August 1, 2025). <https://arxiv.org/abs/2411.13587>
15. Robey, A. et al. (University of Pennsylvania), "Jailbreaking LLM-Controlled Robots," arXiv, October 17, 2024. <https://arxiv.org/abs/2410.13691>
16. ISO, "ISO 10218-1:2025 Robotics: Safety requirements, Part 1: Industrial robots," February 2025. <https://www.iso.org/standard/73933.html>
17. ISO, "ISO 10218-2:2025 Robotics: Safety requirements, Part 2: Industrial robot applications and robot cells," February 2025. <https://www.iso.org/standard/73934.html>
18. A3 (Association for Advancing Automation), "Updated ISO 10218 FAQ," March 20, 2025. <https://www.automate.org/robotics/blogs/updated-iso-10218-faq>

- 19.** A3, "Key Robot Safety Standard Gets A Major Revision" (ANSI/A3 R15.06-2025), September 2025.
<https://www.automate.org/industry-insights/ansi-a3-publish-revised-r15-06-industrial-robot-safety-standard>
- 20.** ISO, "ISO/TS 15066:2016 Robots and robotic devices: Collaborative robots," 2016 (confirmed 2022).
<https://www.iso.org/standard/62996.html>
- 21.** Oitzman, M., The Robot Report, "New AMR safety standard available with release of ANSI/A3 R15.08-2," October 26, 2023. <https://www.therobotreport.com/new-amr-safety-standard-available-with-release-of-ansi-a3-r15-08-2/>
- 22.** ANSI Blog, "ANSI/A3 R15.08-3-2026: Industrial Mobile Robot Applications," September 2026.
<https://blog.ansi.org/ansi/ansi-a3-r15-08-3-2026-industrial-mobile-robot/>
- 23.** ETSI / 3GPP, "TS 122 104 V16.5.0: Service requirements for cyber-physical control applications in vertical domains," Release 16.
https://www.etsi.org/deliver/etsi_ts/122100_122199/122104/16.05.00_60/ts_122104v160500p.pdf
- 24.** Physical Intelligence, "openpi" repository (models, hardware requirements, remote inference), 2025.
<https://github.com/Physical-Intelligence/openpi>
- 25.** VDA and VDMA, "VDA 5050 Version 3.0.0: Communication interface between fleet control and mobile robots." <https://github.com/VDA5050/VDA5050>
- 26.** MassRobotics, "MassRobotics Interoperability Standard, Version 1.0, for Industrial Mobile Robots," 2021.
https://github.com/MassRobotics-AMR/AMR_Interop_Standard
- 27.** ISA, "ISA/IEC 62443 Series of Standards." <https://www.isa.org/standards-and-publications/isa-standards/isa-iec-62443-series-of-standards>
- 28.** Hugging Face, "LeRobot" repository. <https://github.com/huggingface/lerobot>
- 29.** Li, X. et al., "Evaluating Real-World Robot Manipulation Policies in Simulation" (SIMPLER), arXiv, May 9, 2024. <https://arxiv.org/abs/2405.05941>
- 30.** OSHA, "Robotics: Overview." <https://www.osha.gov/robotics>
- 31.** NIST, "AI Risk Management Framework (AI RMF 1.0)," January 26, 2023. <https://www.nist.gov/itl/ai-risk-management-framework>