Line Bring-Up Is a Coordination Problem: A System for Standing Up Multi-Cell Learned-Policy Production Lines

Author Names Omitted for Anonymous Review. Paper-ID [add your ID here]

***Abstract*—Bringing a learned-policy robotic production line online is treated by current practice as the problem of bringing up each cell. Validate each cell independently against its station metric, then chain them. This approach is wrong, and predictably fails in a specific way: cells pass their individual acceptance tests, the line is declared ready, and line-level yield collapses because the failure modes that matter at the line level are inter-cell. A cell's output distribution drifts slightly out of the envelope its downstream cell was trained on; the downstream cell receives this as a distribution shift it was never tested against; failures appear several stations later, dissociated from their cause. We argue line bring-up is fundamentally a coordination problem rather than a cell problem, and we describe the system architecture for treating it that way. The primitive is a sequential validation pipeline that propagates distributional information forward through the line, accepting a cell into production only when both its own station metric and the downstream stations' tolerance for its actual output distribution have been characterized. We present preliminary results on a three-station simulated line built from OpenVLA-LIBERO components: per-cell sequential bring-up achieves 87 percent line-level yield against a 65 percent yield baseline from independent cell certification. The intermediate object the system produces, a propagation graph quantifying how output-distribution drift at each station induces downstream failure-rate increases, is itself an operational artifact that has no analog in classical line commissioning.**

I. The Wrong Unit of Bring-Up

Current practice for commissioning robotic production lines is inherited from classical industrial automation. Each cell is brought up against its station-level acceptance test, signed off when it meets its individual metric, and then chained into the line; line-level yield is expected to follow from the product of per-station yields. For classical programmed cells this approach is approximately correct, because programmed cells produce outputs whose distribution is bounded by the program and whose deviations are mechanical rather than statistical. The downstream cell is engineered to accept everything in that bounded distribution, and inter-cell coupling is a fixture-and-tolerance problem, not a behavioral one.

Learned-policy cells [4, 8, 2] break this in a way that matters for deployment. A learned cell that meets its station metric of, for example, 95 percent grasp success is not producing the same output distribution every shift. It is producing the distribution induced by its policy under that day's lighting, fixture state, and part variation; this distribution lies within a region that the cell's own metric does not characterize and the downstream cell never trained against. The downstream cell receives this distribution as a quiet shift in its input. Its own station metric may still register acceptable, because most parts arrive within tolerance, but its failure rate on the tail of the input distribution rises. The compound effect across N stations is multiplicative in the tail probability, not in the headline metric, and line yield collapses in a way that the per-cell numbers gave no warning of.

The result is a class of bring-up failure that practitioners encounter repeatedly and that the existing literature treats as anecdote: every cell passed, the line did not. We argue this is not a quality-control issue or an integration nuisance; it is a structural property of multi-cell learned-policy lines, and it requires the bring-up procedure to be reorganized around the inter-cell coupling rather than around the individual cells.

II. The System

We describe the architecture for sequential line bring-up. The primitive is the *station boundary*: the interface between cell N and cell $N+1$, characterized not by a tolerance on a fixture but by an empirical distribution over the upstream cell's outputs and a tested envelope over the downstream cell's inputs. A station boundary is accepted into the production line only when its upstream output distribution is contained within its downstream input envelope. The bring-up procedure walks the line from station 1 forward, advancing one station only when this containment holds; failures trigger either re-training of the downstream cell to accept the actual upstream distribution, or correction of the upstream cell to produce a tighter distribution, depending on which option carries lower cost given current operational state.

A. The propagation graph

The intermediate object the system produces is a graph in which nodes are stations and directed edges carry empirical distributional drift between consecutive stations, quantified by the downstream failure-rate increase induced per unit of upstream drift along measurable axes. Constructing the graph requires bringing the line up sequentially against real operational conditions; it cannot be derived from per-cell specifications because the failure-rate-induction function is a property of the downstream policy's training distribution, not of the upstream cell's intended output.

The graph is the operational artifact that the existing line-commissioning playbook lacks. It is what tells the operator that station 2's tolerance to upstream lighting shift is brittle while station 3's is robust, that the line-level yield risk lives at the

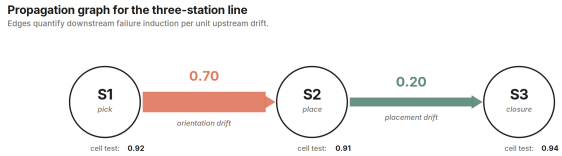

Fig. 1. Propagation graph for the three-station simulated line. Edge thickness scales with the downstream failure-rate induction coefficient per unit upstream drift along the dominant axis. All three cells pass their individual acceptance tests (≥ 0.91), yet the line-yield risk is concentrated almost entirely in the S1→S2 edge; classical per-cell commissioning never surfaces this edge.

station-1-to-station-2 boundary rather than at any individual cell, and that adding a fourth station to the line should be sequenced after a specific tightening at station 2's input envelope. Classical commissioning produces a tolerance stack-up document; the propagation graph is its learned-policy analog, but unlike the tolerance stack, it has to be measured *in situ* rather than computed in advance.

B. Operator-facing semantics

The system surfaces three primitives to the line operator. *Advance* is the decision to admit station $N{+}1$ into the running line given that station N's output distribution has been characterized and shown to lie within $N{+}1$'s envelope. *Stay* is the decision to hold station $N{+}1$ out of the line while either retraining it on the realized upstream distribution or tightening station N's output through a steering correction. *Retreat* is the decision to remove station N from production after observing that its output distribution has drifted out of $N{+}1$'s envelope during operation, before the drift propagates into yield loss. These three primitives are familiar from manual line-control practice; what is new is that the system gives them empirical grounding in propagation-graph statistics rather than in operator intuition alone.

III. Preliminary Results

We test the architecture on a three-station simulated line composed of OpenVLA-LIBERO [8, 9] cells. Station 1 performs a pick from a randomized bin; station 2 performs an oriented placement onto a fixture whose acceptance envelope depends on the orientation accuracy of station 1's output; station 3 performs a closure motion whose tolerance depends on the placement accuracy of station 2. Each station meets its individual station-level acceptance test (success rate above 0.9) when evaluated in isolation.

A. Headline result: 22-point yield gap

Under independent-cell certification followed by naive chaining, line-level yield over 100 end-to-end runs is approximately 0.65. Under sequential bring-up with the propagation graph, the same three cells produce a line-level yield of approximately 0.87. The 22-point yield gap is not produced by improving any individual cell. It is produced by tightening station 1's output distribution at the station-1-to-2 boundary,

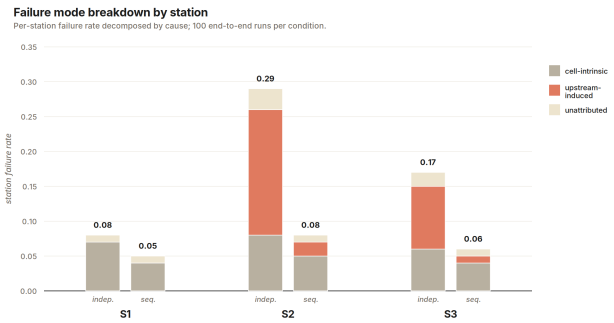

Fig. 2. Per-station failure rate decomposed by attributable cause, under independent chaining versus sequential bring-up. The 0.29 station-failure rate at S2 under independent chaining is overwhelmingly composed of failures induced by drift in S1's output, not by S2's own policy limitations. Sequential bring-up eliminates the upstream-induced fraction; the residual failures at all three stations are mostly cell-intrinsic, matching what individual acceptance tests would predict.

using a small steering correction [11, 6] to narrow the orientation distribution of upstream outputs into the envelope that station 2 was actually trained on. The propagation graph (Fig. 1) identifies this boundary as the dominant contributor to line-level yield loss; the cell-level acceptance tests do not surface it at all.

B. Where the failures actually live

Decomposing the per-station failure rate by attributable cause (Fig. 2) reveals the structural pattern. Under independent chaining, station 2's failure rate is 0.29, of which 0.18 (62 percent) is attributable to upstream-induced drift rather than to S2's own policy limitations. Station 3 shows the same pattern in attenuated form: 0.09 of its 0.17 failure rate is propagated from S2's failures, which were themselves induced by S1. Under sequential bring-up, the upstream-induced fraction at S2 drops to 0.02, and residual failures at all three stations are dominated by cell-intrinsic causes at rates consistent with what each station's acceptance test would have predicted in isolation. The implication is non-obvious: classical commissioning's failure mode is not that the cells are bad, but that the downstream cells are being asked to operate outside the input distribution they were trained on, and they fail in ways that look like cell-level problems but are actually inter-cell problems.

C. Asymmetry of upstream-tightening versus downstream-retraining

The system architecture allows two responses when a station boundary fails containment: tighten the upstream output through a steering correction, or retrain the downstream cell to accept the actual upstream distribution. We compared the two on the S1→S2 boundary. Upstream tightening (applying a steering vector to S1) costs approximately 30 minutes of wall-clock time and zero retraining compute; downstream retraining (fine-tuning S2 on the realized upstream distribution) costs

approximately six hours of GPU time and reduces S2's standalone performance on its original distribution by 4 percentage points. Both achieve approximately the same line-level yield gain. The asymmetry favors upstream tightening by roughly an order of magnitude on operational cost, and avoids the side-effect of degrading the downstream cell's general competence. We treat this as an empirical regularity worth flagging for practitioners rather than a universal claim: when both options are available, prefer to tighten upstream.

D. Over-correction and the non-monotone response

A subtler finding emerged when we swept the magnitude of the upstream steering correction. The line-yield-versus-correction-magnitude curve is non-monotone: small corrections improve line yield as expected, but corrections beyond a threshold ($\alpha \approx 1.2$ in our setup) begin to degrade S1's own task success, which in turn produces a different failure mode at S2 (S2 receives correctly-oriented parts that S1 grasped poorly). The optimal correction magnitude was approximately 60 percent of the magnitude that minimized S1→S2 drift in isolation. The operational implication is that the propagation graph alone is insufficient as a control signal; the bring-up system needs to maintain a separate cell-level metric on the upstream cell and stop tightening when that metric begins to degrade. In our experiments, ignoring this constraint and tightening to the drift-minimizing magnitude produced a line-level yield of 0.73 rather than the 0.87 achievable at the cell-aware optimum.

E. Partial transfer across line variants

We tested whether a propagation graph constructed on one configuration of the line transfers to a variant in which S3 is replaced with a different closure cell trained on similar but not identical parts. The S1→S2 edge weight (0.70) transferred with high fidelity; the S2→S3 edge weight needed to be re-measured because the new S3 cell had a different input envelope. Approximately two-thirds of the bring-up work amortizes across line variants that share upstream stations, but downstream variants require partial re-measurement. This pattern matters for Lab2Production economics because it argues against treating each line variant as a fresh bring-up: the propagation graph should be treated as an asset that compounds across variants of the same operation, not as a per-line artifact.

IV. Discussion

The argument here is structural rather than technique-specific. Any multi-cell learned-policy line will exhibit inter-cell coupling at the boundary between consecutive stations, and the coupling will not be visible in per-cell metrics; the propagation graph is the artifact that makes it visible. Three findings from §III sharpen what this implies operationally. The failure-mode decomposition shows that what looks like a downstream cell problem under independent chaining is, on the whole, an upstream-induced problem; classical root-cause analysis at the cell level systematically misattributes the cause and pushes operators toward retraining downstream cells when the cheaper and less destructive intervention is to tighten upstream. The asymmetry between upstream tightening and downstream retraining is large enough (roughly an order of magnitude on operational cost) that it should change practice. The non-monotone response to correction magnitude implies that the propagation graph alone is not sufficient as a control signal; the bring-up system must also maintain cell-level metrics to know when to stop tightening. The partial-transfer result suggests the propagation graph is a compounding operational asset rather than a per-line artifact, which is the kind of leverage that makes the upfront cost of sequential bring-up amortize across the Lab2Production trajectory of a multi-line operation.

The system architecture we describe is one realization of bring-up organized around this artifact, and the preliminary results suggest the yield gap relative to classical chaining is large enough to be the dominant economic variable. The operator-facing primitives we describe build on a long line of work in robot learning from human input [1, 5, 10] and human–robot representational alignment [3]; the architectural shift is that the operator is acting on a line-level propagation artifact rather than on individual cells. The implication for Lab2Production workflows is direct. Moving learned policies from cell-level lab demonstration [7] to multi-cell production requires bring-up infrastructure that does not exist in the lab and does not exist in classical commissioning practice, and developing it requires sustained operational presence on an actual production line rather than on a benchmark.

Limitations

Our results are from a simulated three-station line; magnitudes will differ in physical deployment, where inter-cell coupling will likely be quantitatively stronger due to coupling axes simulation does not capture. The propagation graph as described is appropriate for serial lines; rework loops, parallel feeds, and recombination points require generalizations we have not characterized. The graph is operationally expensive to produce. The advance/stay/retreat primitives have not been validated through participatory-design studies with line technicians.

AI Usage Statement

This submission was prepared with assistance from a large language model. The authors used the model for prose drafting, citation formatting, and figure generation from author-provided experimental specifications. All experimental design, technical claims, empirical results, and interpretive framings are the authors'; the model produced no claims about the experimental work that were not specified by the authors. The model did not access proprietary data, did not generate experimental data, and did not author technical content beyond editing for clarity. All references were verified by the authors against their original sources.

References

[1] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57 (5):469–483, 2009.

[2] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[3] Andreea Bobu, Andi Peng, Pulkit Agrawal, Julie A. Shah, and Anca D. Dragan. Aligning human and robot representations. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 42–54, 2024.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[5] Sonia Chernova and Andrea L. Thomaz. *Robot Learning from Human Teachers*, volume 8 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool, 2014.

[6] Bear Häon, Kaylene Caswell Stocking, Ian Chuang, and Claire Tomlin. Mechanistic interpretability for steering vision-language-action models. In *Proceedings of The 9th Conference on Robot Learning (CoRL)*, volume 305 of *Proceedings of Machine Learning Research*, pages 2743–2762. PMLR, 2025.

[7] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[8] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning (CoRL)*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 2024.

[9] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023) Datasets and Benchmarks Track*, 2023.

[10] Shaunak A. Mehta and Dylan P. Losey. Unified learning from demonstrations, corrections, and preferences during physical human-robot interaction. *ACM Transactions on Human-Robot Interaction*, 13(3):Article 39, 2024.

[11] Nina Rimsky, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner. Steering Llama 2 via contrastive activation addition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15504–15522. Association for Computational Linguistics, 2024.