applied sciences

MDPI

Article

Recovery Policy Optimization for Online Quality Monitoring and Hybrid Force/Position Control in Robotic Precision Assembly

Fengwei Zhang [1], Rui Xu [2] and Qiang Tang [3,*]

[1] College of Mechanical and Electronic Engineering, Northwest A&F University, Yangling 712100, China; zhangfengwei2026@yeah.net
[2] Changsha University of Science and Technology, Changsha 410114, China; 202425740104@csust.edu.cn
[3] School of Artificial Intelligence, Anhui University of Science and Technology, Hefei 231131, China
* Correspondence: tangqiang@whu.edu.cn

Abstract

Contact-rich precision assembly fails in ways visible in the force/torque and proprioceptive streams before they are visible in the finished part, which makes online quality monitoring attractive. We ask whether monitoring accuracy or the recovery policy is the binding constraint on closed-loop performance, and answer it by improving both. For monitoring we propose AdaStack, which mixes a random forest on summary statistics, a fully convolutional network and a CNN-LSTM with simplex weights fitted on pooled out-of-fold posteriors, so the combination adapts to whichever learner suits a given task. On the UCI Robot Execution Failures benchmark, evaluated over repeated stratified cross-validation against eight baselines under one protocol, AdaStack attains the best mean rank (3.20 against 4.00) and the best mean macro F_1 (0.787 against 0.784), and is the top model on three of five tasks; it significantly outperforms dynamic time warping on two tasks after Holm correction, while differences against the stronger baselines remain within the resolution of a 47-to-164-trial benchmark. We then close the loop in simulation, on a physics-based MuJoCo peg-in-hole cell. There the limiting factor turns out not to be perception: with a recovery policy written by hand, a spiral search that ignores the estimated state seats 78.5% of insertions while the learned monitor reaches 33.0% and even an oracle given the true fault state reaches 52.0%. Optimizing the twelve free policy parameters against a cost over seating, contact force, cycle time and damage, with disjoint optimization and evaluation seeds, yields in this simulated cell a controller that matches the search's seating rate (76.5% against 78.5%) while cutting mean peak contact force by 46% (17.2 N against 31.6 N) using the learned monitor rather than an oracle. The same gain, however, is available without the monitor: a spiral search whose own three parameters are optimized with the same budget seats 79.0% at 18.3 N, and it matches or beats the monitor-driven policy under four cost weightings and five of six shifted operating conditions. What paid was optimizing the reaction, not conditioning it on the estimated state. Effort spent on the recovery policy repaid far more than effort spent on monitoring architecture; whether this transfers to a physical cell remains to be tested.

Keywords: robotic assembly; quality monitoring; multimodal fusion; spatiotemporal Transformer; hybrid force/position control; peg-in-hole; policy optimization; negative results; reproducible evaluation

Check for updates

Academic Editor: Yutaka Ishibashi

Received: 24 September 2026
Revised: 23 October 2026
Accepted: 24 October 2026
Published: 29 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Precision assembly is the point at which a robot stops moving through free space and starts negotiating contact. Inserting a connector, seating a bearing or mating a shaft to a bore all involve clearances on the order of tens of micrometres, which is finer than the absolute positioning accuracy of most industrial manipulators. The consequence is that assembly is not solved by accurate motion alone: the robot must sense what is happening at the contact and react to it. When it does not, the failure modes are familiar and expensive: the part jams at the mouth of the hole, the peg wedges at an angle, galling scores a mating surface, or debris blocks the seat and the controller keeps pressing until something yields.

Two research communities have converged on this problem from different directions. The control community has long treated it as a question of regulating interaction explicitly, beginning with the hybrid position/force decomposition of Raibert and Craig [1], in which a selection matrix partitions the task frame into directions that are position controlled and directions that are force controlled. That decomposition remains the backbone of industrial contact tasks and of recent work on friction stir welding [2], robotic polishing [3] and compliant batch assembly [4]. Its weakness is that the partition and the gains are usually fixed in advance, so a controller tuned for nominal insertion behaves badly when the actual failure is lateral misalignment or an obstructed seat. The learning community has instead attacked the problem by making the policy adaptive, combining vision and force feedback through reinforcement learning [5–8]. These methods are powerful but data hungry, and they typically fold perception and control into a single opaque policy, which makes it hard to say what the system believes about the contact state or to audit why it acted.

Between these lies a more modest target that is attractive for deployment: *online quality monitoring*. Rather than learning the control policy end-to-end, one learns a monitor that reads the sensor streams and infers the current quality state of the assembly, and then uses that state to reconfigure a conventional, familiar controller. This keeps the control law interpretable and certifiable while still adapting to what is actually happening. It also matches the sensing reality of an assembly cell, where a six-axis force/torque sensor and joint encoders are standard and cheap, and where the informative signal is a multivariate time series rather than an image.

Time-series modeling has moved quickly in this space: convolutional and Inception-style networks set strong baselines [9–11], Transformers have been adapted to long multivariate series through patching [12] and applied to multivariate anomaly detection [13–15], and multimodal fusion has become a subject in its own right [16–20]. Deformable attention [21] supplies a mechanism for attending at learned continuous offsets, which is directly suggestive for signals related by a physical delay. Section 2 reviews these strands in detail.

That observation shapes our monitoring design, but not in the direction we first expected. Contact mechanics does suggest specialized structure: the force response lags the commanded motion, and the modalities live on different timescales. What the data says, however, is that no single learner is best everywhere. On the public benchmark a random forest on summary statistics wins the smallest tasks, where its feature prior compensates for a few dozen trials, while convolutional and recurrent networks win where sequences carry more structure. We therefore propose **AdaStack**, which combines a random forest, a fully convolutional network and a CNN-LSTM with nonnegative simplex weights fitted on pooled out-of-fold posteriors, so the stack adapts to whichever member suits the task instead of committing to one in advance.

Our second contribution is the evaluation, which is deliberately two-sided. It is easy to propose an architecture and demonstrate it where the differences are smaller than the noise. The public benchmark for this problem, the UCI Robot Execution Failures dataset of Lopes and Camarinha-Matos [22,23], has five tasks of 47 to 164 trials, each only 15 samples

long. We evaluate under repeated stratified crossvalidation and test significance with the Nadeau–Bengio corrected resampled t-test, which accounts for folds sharing training data. Under that protocol we report a negative result: no deep model, including ours, separates from a random forest on summary statistics.

Because that benchmark is small and short, a negative result on it alone would be weak evidence, so we also built a regime with long streams: a physics-based peg-in-hole cell simulated in MuJoCo [24] with long, high rate multimodal streams and overlapping, near-threshold failure conditions, measuring *early* detection against observation horizon. There a single fully convolutional network [9] leads at nearly every horizon, and stacking adds nothing. Closing the loop then shows why more monitoring did not pay: the recovery policy, not the diagnosis, is the binding constraint, and optimizing it delivers what better perception did not.

This coupling between what the monitor reports and what the controller does with the report is the central subject of the paper. Adaptive assembly control usually adapts in one of two places: inside the control law, by scheduling gains or impedance online, or inside an end-to-end learned policy. In both cases the quality of the state estimate and the quality of the reaction are entangled, so a weak result cannot be attributed to either. We keep them apart on purpose. The monitor is evaluated on its own, the reaction is a small explicit table from states to controller settings, and an oracle that receives the true state separates the two in the loop. This decomposition is what lets us say that the reaction, not the estimate, was the binding constraint, and it is what makes the reaction cheap to optimize once that is known.

In summary, this paper contributes:

1. **AdaStack**, a monitoring architecture that mixes a random forest, a convolutional network and a CNN-LSTM with weights fitted on pooled out-of-fold posteriors. Under one protocol against eight baselines it attains the best mean rank and the best mean macro F_1 on the public benchmark, and we show that how the weights are fitted matters more than which members are in the stack.
2. A **correction-aware evaluation** showing that this benchmark cannot separate these methods: 22 of 40 paired comparisons reach $p < 0.05$ under an uncorrected Wilcoxon test over folds, and only 2 survive the Nadeau–Bengio corrected resampled t-test with Holm adjustment, both against 1-NN DTW, the weakest baseline. AdaStack attains the best mean rank and the best mean macro F_1, and we state plainly that the margin over the stronger baselines is not significant.
3. A **physics-based assembly benchmark** with an early-detection protocol, built specifically to give the proposed inductive biases the long, high rate multimodal streams they assume, on which a 2017 convolutional network still leads.
4. The identification of the **recovery policy as the binding constraint**: with a policy written by hand, a spiral search that ignores the state outperforms both learned monitors and an oracle given the true state, which separates perception quality from benefit in the loop.
5. An **optimized recovery policy** that follows from that diagnosis and is the paper's positive result: optimizing twelve free policy parameters against a cost over seating, contact force, cycle time and damage, with disjoint optimization and evaluation seeds, matches the search's seating rate while reducing mean peak contact force by 46% in simulation. We then test this result under alternative cost weights, under shifted contact stiffness, friction, clearance, sensor noise and misalignment, and against a spiral search optimized with the same budget. The last comparison shows that most of the gain comes from optimizing the reaction's parameters rather than from conditioning them on the monitor's state estimate.

6. An observation of independent methodological interest: two of the five tasks in the public benchmark (LP2 and LP3) contain *byte-identical sensor data* under two different labelings, so treating them as independent datasets overstates the diversity of the benchmark.

The rest of the paper is organized as follows. Section 2 reviews force control, time-series classification and multimodal fusion. Section 3 defines the problem, AdaStack, the hybrid controller, the recovery-policy optimization, the online implementation, the two datasets and the evaluation protocol. Section 4 reports the public benchmark, the early-detection study and the closed-loop study, including the cost-weight sensitivity and robustness tests. Section 5 discusses what the results mean for practice and lists the threats to validity, and Section 6 concludes.

2. Related Work

2.1. Force Control and Contact-Rich Assembly

The hybrid position/force formulation [1] partitions the task frame with a selection matrix S, regulating position along some directions and force along the complementary ones. It remains the standard decomposition for industrial contact tasks and continues to be extended: force/position hybrid schemes have recently been applied to eight-axis friction stir welding [2] and to force–position decoupling in robotic bonnet polishing [3], while compliant control strategies have been developed for robust batch assembly [4]. The limitation relevant here is that S, the force setpoint and the stiffness are normally fixed offline. A controller tuned for a nominal insertion continues to press when the true condition is a blocked seat, which converts a recoverable error into a damaged part.

Learning-based approaches make the policy itself adaptive. Recent work combines admittance control with residual reinforcement learning using visual and force feedback [6], fuses multiple modalities to drive reinforcement learning for peg-in-hole assembly [5], and applies two-stage deep reinforcement learning with hybrid vision and force perception to robotic assembly for lunar construction [7]; multimodal reinforcement learning has also been used to push precision assembly into large workspaces [8]. These systems are capable, but they require substantial interaction data and they couple perception to action, so the contact state is not separately observable. Our position is intermediate: we learn only the perception component and let it reconfigure a classical controller, which preserves interpretability and lets us measure the monitor and the controller independently. It also differs from adaptive force control that tunes gains or impedance continuously from the measured wrench: there the adaptation law is the reaction, and it is designed once. In our setting the reaction is a separate, low-dimensional object, a lookup from quality state to force limit, search radius and stiffness, so it can be audited, replaced, and optimized against an explicit cost without touching the monitor.

2.2. Time-Series Classification and Industrial Monitoring

Deep classification of time series is a mature field with baselines that are well characterized. Fully convolutional networks established that a simple convolutional stack is a strong learner for this problem [9], InceptionTime extended this with multiscale convolution [10], and a broad review catalogued the design space [11]. Temporal convolutional networks provide dilated causal receptive fields [25], and multiscale feature learning has been applied specifically to industrial process monitoring [26]. An important and frequently under-weighted lesson from this literature is that on small datasets, distance and feature classifiers remain highly competitive, which is precisely the regime of the public assembly benchmark and precisely what our first experiment confirms. The same pattern appears

in other small-sample domains. Data-mining pipelines built on curated features remain the workhorse for biomarker evaluation on cohorts of modest size [27]. Where convolutional networks are used on non-image data, the features are often first arranged into an image so that the network can exploit their structure [28]. Both point to the same lesson: a strong feature prior or an appropriate representation is often worth more than added model capacity.

Transformer variants have since been adapted to multivariate series. PatchTST showed that patching over independent channels is an effective tokenization for long series [12]. For anomaly detection on multivariate streams, recent work in this journal's neighbouring literature includes neighbour relation-guided Transformers [13], contrastive adversarial autoencoders [14], self-supervised contrastive representations [15], and interpretable masked latent generative modeling [29]. Our setting differs in that the label is a discrete quality state with a known physical cause, and in that the monitor's output is consumed by a controller rather than by a human analyst.

2.3. Multimodal and Spatiotemporal Fusion

How to fuse heterogeneous streams is an active question. Recent work addresses the robustness of multimodal fusion via causal mixture-of-experts [16], cross modal attention between appearance and skeleton streams for anomaly detection [17], and text-enhanced Transformer fusion [18]. Closest in spirit to our aims are uncertainty-aware multimodal time series anomaly detection via reduced-order evidence fusion [19] and representation decoupling to boost sensitivity in multimodal anomaly detection [20]; both recognize that modalities contribute unequally and that naive concatenation is suboptimal. On the spatiotemporal side, decoupled spatiotemporal hybrid Transformers [30], dynamic predictive Transformers with temporal relevance regression [31], and Transformers with temporal depthwise convolution [32] all exploit structure in the temporal axis rather than treating it as a bag of tokens.

What these methods generally assume is that the modalities are temporally aligned. In a mechanical contact task that assumption is wrong in a specific and exploitable way: the force response is causally downstream of the commanded motion and therefore delayed. Deformable attention [21] supplies a mechanism to handle this, attention at learned continuous offsets. In preliminary experiments we adapted it to the lag between proprioception and contact force in a Transformer monitor (the "Transformer variant" of Section 4.2); it did not separate from simpler learners at the data scales available, which is what led us to the stacking design of Section 3.2.

3. Materials and Methods

3.1. Problem Statement

Let $\mathbf{x}_{1:T} \in \mathbb{R}^{T \times C}$ be a synchronized multivariate stream from an assembly cell, sampled at a fixed control rate. The channels partition into M modalities,

$$\mathbf{x}_{1:T} = [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}], \qquad \mathbf{x}^{(m)} \in \mathbb{R}^{T \times C_m}, \qquad \sum_{m=1}^{M} C_m = C. \tag{1}$$

In the simulated cell the modalities are contact force ($C_1 = 3$), contact torque ($C_2 = 3$), pose ($C_3 = 6$) and velocity ($C_4 = 6$); on the public benchmark only force and torque are recorded, giving $M = 2$ and $C = 6$. We treat force and torque as distinct modalities because they carry different units and different information about the contact configuration.

A monitor is a map from a prefix of the stream to a quality state $y \in \mathcal{Y} = \{1, \ldots, K\}$,

$$\hat{y}_H = \arg\max_{k \in \mathcal{Y}} \; [\mathbf{p}(\mathbf{x}_{1:H})]_k, \qquad H \le T, \tag{2}$$

where $\mathbf{p}$ returns a posterior over $\mathcal{Y}$. For *online* use the relevant quantity is the accuracy at small H, not at $H = T$: a monitor that becomes accurate only once the outcome is mechanically determined cannot change it. We therefore report accuracy as a function of H.

In the closed-loop setting, the monitor additionally parameterizes the controller. Along the insertion axis the wrist runs a force-limited velocity loop; the lateral and orientation axes are impedance controlled toward the commanded pose. Writing v^* for the commanded insertion speed, f^* for the force limit and K_{lat} for the lateral stiffness, the monitor selects a controller configuration at each decision instant through a policy π_θ defined in Section 3.4. Figure 1 shows the complete system.

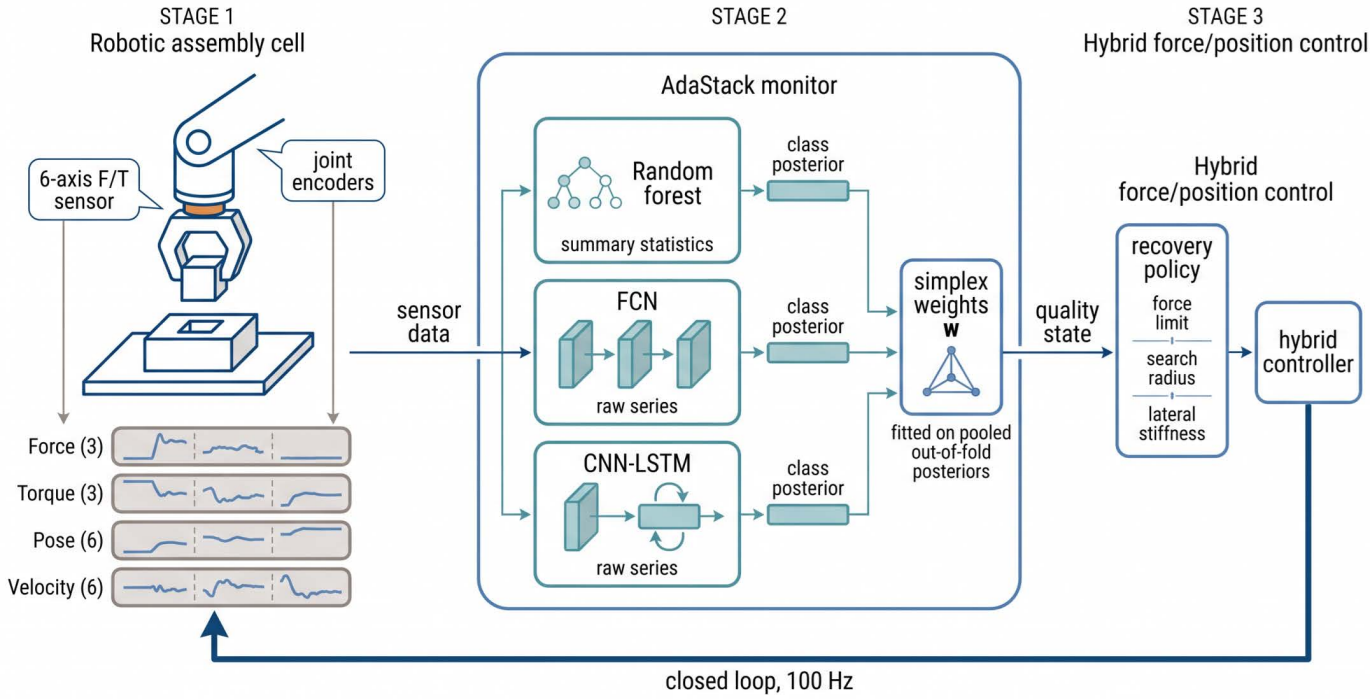


Figure 1. The monitoring and control pipeline. The assembly cell supplies four sensor modalities. AdaStack runs three learners with different inductive biases over them-a random forest on summary statistics, a fully convolutional network and a CNN-LSTM on the raw series—and mixes their class posteriors with simplex weights fitted on pooled out-of-fold predictions from the training fold (Equations (5)–(8)). The resulting quality state selects the force limit, search radius and lateral stiffness of a hybrid force/position controller through the policy of Equation (12). The controller runs at 100 Hz; the monitor classifies the trailing 0.5 s window and reconfigures the controller every 10 control steps (10 Hz). The figure shows the general architecture. In the closed-loop study of Section 4.4 the monitor slot is filled by the fully convolutional network alone, which was the strongest monitor on the simulated cell (Section 4.3); a windowed AdaStack in the same slot is reported as a check in Table 6.

3.2. AdaStack: The Monitoring Architecture

Which learner is best for assembly monitoring changes between tasks. On the public benchmark a random forest on summary statistics is strongest on the smallest tasks, where its feature prior compensates for having a few dozen trials, while a convolutional or recurrent network is strongest where sequences carry more structure. A fixed choice is therefore wrong somewhere by construction, which argues for combining learners rather than searching for a single best one.

3.2.1. Members

AdaStack combines $R = 3$ members with deliberately different inductive biases. The first is a random forest on a fixed map of summary statistics. Writing $\Delta\mathbf{x}_t = \mathbf{x}_{t+1} - \mathbf{x}_t$ and taking each operator channel-wise over time,

$$\varphi(\mathbf{x}_{1:T}) = \Big[\, \bar{\mathbf{x}},\ \mathrm{sd}(\mathbf{x}),\ \min_t \mathbf{x}_t,\ \max_t \mathbf{x}_t,\ \overline{\Delta\mathbf{x}},\ \max_t |\Delta\mathbf{x}_t| \,\Big] \in \mathbb{R}^{6C}. \tag{3}$$

The second is a fully convolutional network [9] and the third a CNN-LSTM, both applied to the raw series. Each member returns a posterior over $\mathcal{Y}$,

$$\mathbf{p}^{(1)} = \mathrm{RF}\big(\varphi(\mathbf{x})\big), \qquad \mathbf{p}^{(2)} = \sigma\big(g_{\mathrm{FCN}}(\mathbf{x})\big), \qquad \mathbf{p}^{(3)} = \sigma\big(g_{\mathrm{LSTM}}(\mathbf{x})\big), \tag{4}$$

with σ the softmax. The stacked posterior is a convex combination,

$$\mathbf{p}_{\mathbf{w}}(\mathbf{x}) = \sum_{r=1}^{R} w_r\, \mathbf{p}^{(r)}(\mathbf{x}), \qquad \mathbf{w} \in \Delta^{R-1} = \Big\{ \mathbf{w} \in \mathbb{R}^R : w_r \geq 0,\ \textstyle\sum_r w_r = 1 \Big\}. \tag{5}$$

Because the vertices of Δ^{R-1} are the individual members, any weight search that attains its optimum can only place the stack below its best member through selection noise, never by construction.

3.2.2. Fitting the Weights

How $\mathbf{w}$ is fitted matters more than which members are in the stack. Let $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ be the training fold and let $\kappa(i) \in \{1, \ldots, k\}$ index the inner cross-validation fold containing sample i. Writing $\mathbf{p}_{-\kappa}^{(r)}$ for the r-th member fitted with fold κ held out, the *out-of-fold* posterior of sample i is

$$\tilde{\mathbf{p}}_i^{(r)} = \mathbf{p}_{-\kappa(i)}^{(r)}(\mathbf{x}_i), \qquad i = 1, \ldots, n, \tag{6}$$

so every training sample contributes a prediction made by a model that never saw it. The weights are then chosen once, on all n pooled predictions, over a grid $\mathcal{G}$ of the simplex in steps of 0.1:

$$\mathcal{W}^\star = \arg\max_{\mathbf{w} \in \mathcal{G}} \mathrm{F}_1^{\mathrm{macro}}\Big(\big\{ \arg\max_k \big[\textstyle\sum_r w_r \tilde{\mathbf{p}}_i^{(r)} \big]_k \big\}_{i=1}^n,\ \{y_i\}_{i=1}^n \Big). \tag{7}$$

The alternative, scoring $\mathcal{G}$ separately inside each inner fold and summing, fits a combiner on n/k samples at a time; on a 47-trial task that is about a dozen, and the search then tracks noise in the inner validation split. Finally we average any ties in (7) and shrink towards the uniform weight to damp residual selection noise,

$$\hat{\mathbf{w}} = (1 - \lambda)\, \overline{\mathcal{W}^\star} + \lambda\, \tfrac{1}{R}\mathbf{1}, \qquad \lambda = 0.25, \tag{8}$$

after which all three members are refitted on the whole training fold and (5) is evaluated with $\hat{\mathbf{w}}$. No test data enters the weight choice at any point. Algorithm 1 states the procedure.

3.3. Hybrid Force/Position Control Driven by the Monitor

Control follows the hybrid position/force decomposition [1]. A diagonal selection matrix $S \in \{0, 1\}^{6\times 6}$ splits the task frame into force-regulated and position-regulated directions, and the applied wrench is

$$\boldsymbol{\tau} = S\, \boldsymbol{\tau}_{\mathrm{force}} + (I - S)\, \boldsymbol{\tau}_{\mathrm{pos}}, \tag{9}$$

with $S = \mathrm{diag}(0, 0, 1, 0, 0, 0)$ in nominal insertion, so the insertion axis z is force-regulated and the lateral and orientation axes are position-regulated by an impedance law $\boldsymbol{\tau}_{\mathrm{pos}} = K(\mathbf{q}^* - \mathbf{q}) - D\dot{\mathbf{q}}$.

The force-regulated axis runs a velocity loop under a force limit rather than a constant force command, which is how force-controlled insertion is run on a real cell. With $e_v = -v^* - \dot{q}_z$,

$$\tau_z = \mathrm{clip}\Big(K_v e_v + K_i \int_0^t e_v \, \mathrm{d}t', \; -f^*, \; f^*\Big), \tag{10}$$

with anti-windup: the integral advances only while the output is unsaturated or the error would unwind it. The wrist therefore descends at v^* in free space and saturates at f^* once motion is blocked. Discrete stability of (10) requires $\Delta t \, K_v / m_{\mathrm{eff}} < 2$; with $m_{\mathrm{eff}} \approx 0.51$ kg and $\Delta t = 5 \times 10^{-4}$ s this bounds $K_v \ll 1020$, and we use $K_v = 250$ N s/m and $K_i = 2.5 \times 10^4$ N/m.

Algorithm 1 AdaStack: fitting and inference

Input: training fold $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$; members $r = 1, \ldots, R$; inner folds k; grid $\mathcal{G} \subset \Delta^{R-1}$; shrinkage λ
Output: fitted members $\{\mathcal{M}^{(r)}\}$ and weights $\hat{\mathbf{w}}$

$k \leftarrow \min(k, \min_c |\{i : y_i = c\}|)$ ▷ stratification limit
partition $\mathcal{D}$ into stratified folds $\mathcal{F}_1, \ldots, \mathcal{F}_k$
for $\kappa = 1$ **to** k **do**
  if some class is absent from $\mathcal{D} \setminus \mathcal{F}_\kappa$ **then**
    continue ▷ cannot fit all classes on this split
  end if
  for $r = 1$ **to** R **do**
    fit member r on $\mathcal{D} \setminus \mathcal{F}_\kappa$
    $\tilde{\mathbf{p}}_i^{(r)} \leftarrow$ its posterior for all $i \in \mathcal{F}_\kappa$ ▷ Equation (6)
  end for
end for
$\mathcal{W}^\star \leftarrow \arg\max_{\mathbf{w} \in \mathcal{G}} \mathrm{F}_1^{\mathrm{macro}}$ on the pooled $\tilde{\mathbf{p}}$ ▷ Equation (7)
$\hat{\mathbf{w}} \leftarrow (1 - \lambda)\overline{\mathcal{W}^\star} + \lambda \mathbf{1}/R$ ▷ Equation (8)
for $r = 1$ **to** R **do**
  refit member $\mathcal{M}^{(r)}$ on all of $\mathcal{D}$
end for

function PREDICT($\mathbf{x}$)
  return $\arg\max_k \left[\sum_r \hat{w}_r \, \mathcal{M}^{(r)}(\mathbf{x}) \right]_k$ ▷ Equations (2) and (5)
end function

Recovery superimposes an Archimedean spiral on the lateral setpoint. Letting t_s be the time spent searching, $\dot{\rho}$ the radial growth rate and ρ_s the radius cap for state s,

$$\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \min\left(\dot{\rho} \, t_s, \; \rho_s\right) \begin{bmatrix} \cos \omega t_s \\ \sin \omega t_s \end{bmatrix}, \qquad \dot{\rho} = 1.6 \text{ mm/s}, \quad \omega = 2\pi(1.2) \text{ rad/s}. \tag{11}$$

A circle of fixed radius, the obvious alternative, only finds the aperture when its radius happens to match the localization error; a growing spiral sweeps the whole disc of plausible errors.

Every $n_d = 10$ control steps (every 0.1 s at the 100 Hz control rate) the monitor classifies the trailing window of length $W = 50$ (0.5 s) and the controller is reconfigured by a policy $\pi_\theta : \mathcal{Y} \to \mathbb{R}^3 \times \{0, 1\}$,

$$\pi_\theta(s) = \left(\phi_s f^*, \;\; \rho_s, \;\; \kappa_s K_{\mathrm{lat}}, \;\; h_s\right), \tag{12}$$

where ϕ_s scales the force limit, ρ_s caps the search radius, κ_s scales lateral stiffness, and h_s is a hold flag that switches the insertion axis to position control at its current depth. We set $h_s = 1$ only for the debris state, which is mechanically unrecoverable; holding rather than releasing the force command matters, because releasing it leaves only the gravity compensation term and the wrist drifts upward out of the hole.

3.4. Optimizing the Recovery Policy

The closed-loop study in Section 4.4 identifies the recovery policy, not the monitor, as the binding constraint, so we treat the policy itself as an object to be designed rather than assumed. We keep the same interface (a quality state selects a controller configuration) and optimize the configuration directly.

Each state s carries three continuous knobs: a multiplier ϕ_s on the commanded force limit, a cap ρ_s on the spiral search radius, and a multiplier κ_s on lateral stiffness. The debris response remains an action that holds position and is not optimized, because that state is mechanically unrecoverable and no setting of the knobs should be allowed to turn it into pressing harder. The parameter vector θ therefore has $4 \times 3 = 12$ free entries, one triple for each recoverable state; the original submission counted 15 by including the three debris entries, which the hold action ignores. The knobs are bounded to $\phi_s \in [0.2, 1.2]$, $\rho_s \in [0, 10]$ mm and $\kappa_s \in [0.3, 1.2]$.

We minimize a scalar cost that reflects what an assembly cell actually cares about,

$$J(\theta) = \big(1 - \text{seat}\big) + 2.0\,\text{damage} + 0.30\,\frac{\bar{F}_{\text{peak}}}{100\ \text{N}} + 0.20\,\frac{\bar{t}}{1200}, \tag{13}$$

where seating and damage rates are fractions of trials, $\bar{F}_{\text{peak}}$ is the mean peak contact force and $\bar{t}$ the mean cycle length in control steps. Damage carries the largest per-unit weight because a scrapped part costs more than a retry; force and time are normalized to their plausible ranges so the weights are interpretable. Read as exchange rates, the weights say that one damaged part costs as much as two failed insertions, that 3.3 N of mean peak force is worth one percentage point of seating, and that so is 60 control steps (0.6 s) of cycle time. Seating therefore dominates; force comes second because contact force is what wears tooling and parts in a cell of this kind, and cycle time comes last because a retry is cheap compared with a scrapped part. These are judgements, not measurements, so Section 4.5 repeats the whole optimization under three other weightings: force-dominant ($w_F = 1.00$, i.e., 1 N is worth one point of seating), cycle-dominant ($w_t = 0.80$) and seating-dominant ($w_F = 0.10$, $w_t = 0.05$), with the damage weight kept at 2.0 throughout.

Optimization uses a simple $(1 + \lambda)$ evolution strategy with $\lambda = 6$, 25 iterations and a decaying step size, started from the policy we wrote by hand in Section 4.4 so that the search can only improve on the design a practitioner would write by hand. The procedure is deliberately plain: the question is whether the policy has headroom, not whether a sophisticated optimizer can find it.

A fair reference needs the same treatment. The spiral search that ignores the state has its own three knobs (force-limit multiplier, radius cap and stiffness multiplier, applied to every trial from the start), and in the original submission they were set by hand while the state-conditioned policy was optimized. We therefore also optimize the state-independent search with the same optimizer, budget, seeds and cost, started from its hand setting ($\phi = 1$, $\rho = 8$ mm, $\kappa = 1$). Comparing the two optimized controllers isolates what the state estimate adds beyond tuning.

Two precautions keep the result honest. First, optimization and reporting use *disjoint trial seeds*, 24 per class for the search and 40 per class held out for the numbers we report, so the policy cannot be tuned to the trials it is scored on. Second, we optimize twice: once with the true state, which bounds what any state-conditioned policy can achieve, and once with the learned monitor in the loop, which is what could actually be deployed. Reporting only the first would overstate the method; reporting only the second would confound policy quality with perception quality.

3.5. Online Implementation

The closed loop runs as follows. The controller of Equations (9)–(11) executes at 100 Hz. Each control step appends the 18-channel observation (force, torque, pose, velocity) to a ring buffer of the last $W = 50$ samples. Until the buffer is full (the first 0.5 s), the controller runs the nominal configuration. From then on, at every tenth step, the buffer is standardized with the training-set mean and standard deviation stored with the monitor, passed through the monitor as one 50×18 window, and the arg-max state selects a row of the policy table. The new configuration takes effect at the next control step and is held for the following ten steps. Two latches keep the reaction well defined. The spiral clock t_s of Equation (11) keeps running while consecutive decisions request a search and resets to zero when a decision requests none, so the radius grows only while the monitor keeps asking for a search. The hold target of the debris response is latched when that state is first entered, because re-reading it from the live position would let any drift become the new setpoint. No smoothing or voting is applied across decisions.

The time budget is 10 ms per control step and 100 ms per monitoring decision. We measured inference latency for a single window (batch size one) on one core of the server CPU (aarch64), with PyTorch and scikit-learn in their default builds and no compilation or quantization (Table 1). The convolutional network that drives the loop needs 1.13 ms per decision (p99 1.15 ms), about 1% of the decision budget and well inside a single 10 ms control step, so it could even run at every step. Timed inside the closed-loop simulation, the same call averaged 1.28 ms. The full three-member AdaStack needs 31.7 ms (p99 32.1 ms, worst case 37.9 ms), almost all of it in the 500-tree forest; this fits the 100 ms decision budget but not a 10 ms step, so it must run asynchronously from the control thread, with the controller keeping the previous configuration until the new decision arrives. Shrinking the forest to 100 trees brings it to 5.9 ms. All figures are for unoptimized Python on a general-purpose core; an embedded controller would need its own measurement, but the margins are wide.

Table 1. Inference latency per monitoring decision: one 50×18 window, batch size one, one CPU thread, 2000 calls for the networks and 500 for the forests after warm-up. The budget is 100 ms per decision (every 10 control steps at 100 Hz) and 10 ms per control step.

Monitor	Median (ms)	p99 (ms)	Max (ms)
FCN (closed-loop monitor)	1.13	1.15	4.94
CNN-LSTM	1.95	2.10	2.29
Random forest, 500 trees	28.14	28.43	28.67
Random forest, 100 trees	5.91	6.10	6.15
AdaStack, all three members	31.73	32.11	37.90

3.6. Datasets

3.6.1. Public Benchmark

The UCI Robot Execution Failures dataset [22,23] contains five tasks recorded on a real manipulator, each trial a 15×6 series of force and torque samples following a detected event. LP1 (88 trials, 4 classes) concerns failures in approach to grasp; LP2 (47, 5) and LP3 (47, 4) concern transfer failures and the resulting part position; LP4 (117, 3) concerns approach to ungrasp; LP5 (164, 5) concerns motion with a part.

We note a property of this benchmark that affects how it should be used: the sensor arrays of LP2 and LP3 are *identical*, element for element (maximum absolute difference 0.0 over all $47 \times 15 \times 6$ entries). They are the same 47 trials annotated under two different taxonomies. Results on LP2 and LP3 are therefore not independent evidence, and we report them separately rather than averaging them into a single benchmark score.

3.6.2. Simulated Assembly Cell

To obtain long, high-rate multimodal streams, we built a square peg-in-hole cell in MuJoCo [24]. A 20 mm peg hangs from a Cartesian wrist with six degrees of freedom actuated by joint torques; the hole is formed by four walls with a clearance per side drawn from $\mathcal{U}(0.8, 1.8)$ mm. The insertion axis runs a force-limited PI velocity loop with anti-windup. The wrist descends at $v^* \sim \mathcal{U}(5, 12)$ mm/s until contact, after which the load saturates at $f^* \sim \mathcal{U}(15, 28)$ N, while the lateral and orientation axes are impedance controlled. This matters for realism: an unregulated force command inserts the peg in tens of milliseconds, which makes every observation horizon effectively post hoc, whereas a force-limited velocity loop takes seconds, as a real cell does.

Five quality states are induced by *physical* conditions rather than by label assignment: *normal*; *lateral misalignment*, an error in locating the hole of 1.2–3.5 clearances on the commanded target; *angular jam*, a commanded tilt of 25–60 mrad; *high friction*, a coefficient of 0.9–1.5 with enough tilt to guarantee wall contact; and *debris*, a small obstruction in the seat. The misalignment is applied to the *commanded* target rather than the initial state, because an offset in the initial state is simply corrected in free space before contact and every trial then succeeds. Condition ranges are deliberately near-threshold and overlapping, which is what makes the problem non-trivial: angular jams recover in 14% of trials and high-friction trials seat in 33%, so the label cannot be read off the outcome. We generate 200 trials per class, 6 s each at 100 Hz, giving 1000 × 600 × 18 samples, and audit the label–physics correspondence in Table 2. The audit uses two depth criteria on purpose. Reaching 75% of the hole depth within the 6 s recording shows that the peg has entered the bore, which is what separates the fault classes physically. Full seating, at least 95% of the depth, is the success criterion of the closed-loop study (Table 6), where trials run for 12 s; within 6 s of nominal open-loop insertion only 72% of normal trials seat fully, because a force-limited descent at 5–12 mm/s takes several seconds. The debris rows illustrate the difference: 94% of those pegs pass 75% of the depth, and none seats fully.

The 1000 trials of the dataset are used only to train and test the monitors. The closed-loop work draws fresh trials from the same generator with seed ranges that do not overlap the dataset or each other. Table 3 lists every use.

Table 2. Label–physics audit of the simulated cell. Each quality state is induced by a randomized physical condition; the resulting outcome statistics confirm the classes are mechanically distinct but overlapping, so the label cannot be read off the outcome. Outcomes are for the nominal controller over the 6 s recording. The 75% criterion marks entry into the bore and is used only for this audit; the closed-loop study counts success as full seating (≥95%), shown in the last column.

Quality State	Trials	Final Depth (mm)	Peak $\lvert F_z \rvert$ (N)	Inserted (≥75%)	Seated (≥95%)
Normal	200	37.87	15.0	0.94	0.72
Lateral misalignment	200	1.91	22.3	0.04	0.04
Angular jam	200	5.94	20.8	0.14	0.09
High friction	200	13.91	19.5	0.33	0.25
Debris in seat	200	33.61	19.0	0.94	0.00

The 60/15/25 split is stratified and repeated over three seeds for the early-detection study; the window monitor used in the loop is the one trained on the first of those splits. No trial used for policy optimization or closed-loop evaluation was seen by any monitor, and the closed-loop numbers in every table come from the same 200 evaluation trials.

Table 3. How simulated trials are used. Seeds are per class; the generator draws all random conditions of a trial from its seed, so disjoint seed ranges give disjoint trials.

Use	Seed Range	Trials	Length
Monitor training (60%)	0–199	600	6 s
Monitor validation (15%)	same pool	150	6 s
Monitor test (25%)	same pool	250	6 s
Policy optimization	700,000+	120 (24/class)	12 s
Closed-loop evaluation	800,000+	200 (40/class)	12 s

3.7. Evaluation Protocol

On the public benchmark we use repeated stratified k-fold cross-validation with three repeats, using $k = 5$ except on LP3, where the rarest class has three members and $k = 3$ is the maximum possible. Channels are standardized using statistics from the training fold only. All neural models are trained for 120 epochs with AdamW, cosine decay, batch size 16, learning rate 10^{-3}, weight decay 10^{-4} and label smoothing 0.05, with gradient clipping at 1.0. We deliberately apply *identical, untuned* hyperparameters to every neural model, so that the comparison is between architectures rather than between tuning budgets; we return to this choice in the discussion.

Significance is assessed with the Nadeau–Bengio corrected resampled t-test. The correction is necessary because cross-validation folds share training data, so the ordinary paired t-test over folds is too liberal; the corrected statistic inflates the variance by $(1/n + n_{\text{test}}/n_{\text{train}})$ to compensate. We additionally report an uncorrected Wilcoxon signed-rank test for reference, and we apply Holm–Bonferroni correction across the eight baselines compared within each task. The same corrected test is applied to the ablation of Section 4.1, with Holm correction across the five tasks.

Baselines span the classical and deep families: 1-NN with dependent multivariate DTW; a random forest on summary statistics (per-channel mean, standard deviation, minimum, maximum, mean and peak absolute first difference); FCN [9]; a CNN-LSTM; a TCN [25]; InceptionTime [10]; a vanilla Transformer encoder [33]; and a patch Transformer over independent channels [12].

On the simulated cell we use stratified 60/15/25 train/validation/test splits over three seeds, select the epoch by validation accuracy, and report test accuracy and macro F_1. For the closed-loop study, each strategy is run on *identical* trial conditions (same seeds, hence same clearances, offsets and friction), so comparisons are paired.

3.8. Implementation and Compute

All models are implemented in PyTorch 2.9 and trained on Ascend 910C NPUs via `torch_npu` under CANN 8.5, using at most four accelerators concurrently. Physics simulation runs on CPU. Generating the 1000-trial dataset takes 25 s on 48 worker processes. The full public benchmark grid (five tasks × ten model variants × 15 folds, 9 on LP3) and the horizon sweep are the dominant costs; total compute for every experiment reported here is a few NPU-hours. The experiments added in revision (cost-weight sensitivity, the optimized search, robustness and the windowed AdaStack) ran on CPU only and took about 30 minutes on roughly 500 cores of the same server.

In accordance with MDPI policy on generative AI, we disclose that a large language model (Claude) was used as a coding and drafting assistant: it implemented the experiment and figure code under the authors' direction, produced the first draft of the manuscript text, and generated the schematic in Figure 1. All experimental design decisions, all reported numbers, and the final text are the authors' responsibility; every number in this paper is

produced by the code described in the Data Availability Statement, and every reference was verified against Crossref metadata.

4. Results

4.1. AdaStack Leads the Public Benchmark

Table 4 reports macro F_1 for ten methods on the five public tasks, all under one protocol and on identical folds. Figure 2 visualizes the per-task comparison.

Table 4. Macro F_1 on the UCI Robot Execution Failures tasks. All methods share one protocol: three repeats of stratified five-fold cross-validation on identical folds (three-fold on LP3, whose rarest class has three members). Best per column in bold; rank is best when lowest. Baselines are ordered by mean rank. After Nadeau–Bengio correction and Holm adjustment, AdaStack differs significantly only from 1-NN DTW, on LP1 and LP5; no difference against any other baseline, nor against its two-member variant, is significant (Table 5 and Section 4.1).

Model	LP1	LP2	LP3	LP4	LP5	Mean	Rank
AdaStack (this work)	**0.981**	0.509	0.747	**0.960**	**0.737**	0.787	**3.20**
w/o recurrent member	0.969	0.542	0.739	0.955	0.735	**0.788**	3.40
RF (summary stats)	0.966	0.578	0.627	0.955	0.692	0.764	4.00
CNN-LSTM	0.953	0.529	0.788	0.946	0.706	0.784	4.00
PatchTST	0.890	**0.589**	0.669	0.888	0.707	0.748	5.00
InceptionTime	0.903	0.477	**0.809**	0.886	0.679	0.751	6.00
TCN	0.897	0.537	0.773	0.846	0.603	0.731	6.60
FCN	0.907	0.485	0.748	0.850	0.662	0.730	6.60
Transformer	0.884	0.557	0.567	0.902	0.661	0.714	6.80
1-NN DTW	0.797	0.529	0.438	0.817	0.563	0.629	9.40

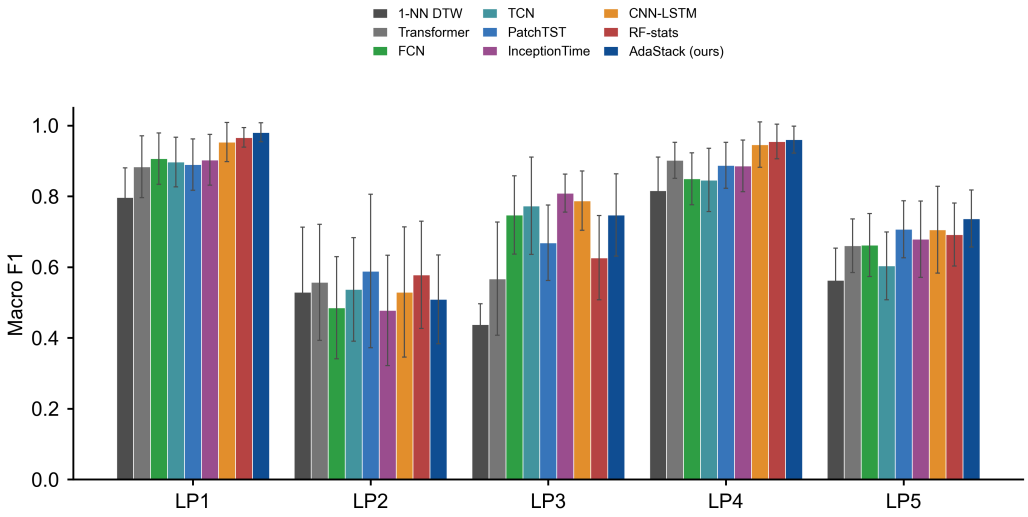


Figure 2. Macro F_1 per task on the public benchmark. Bars are means over repeated stratified cross-validation folds and error bars are one standard deviation across folds. AdaStack (dark blue) leads on LP1, LP4 and LP5. The error bars on LP2 and LP3, each 47 trials, are wider than the spread between methods, which is why no difference on those tasks is significant.

AdaStack attains the best mean rank, 3.20, against 4.00 for both the random forest and the CNN-LSTM, and the best mean macro F_1 among the baselines, 0.787 against 0.784 for the CNN-LSTM. It is the top model on three of the five tasks: LP1 (0.981), LP4 (0.960) and LP5 (0.737). The gain over the strongest single member is largest where the members disagree most, which is the behavior the construction is designed for.

The two tasks it loses are the two smallest. On LP2 PatchTST leads (0.589 against 0.509) and the random forest is also ahead; on LP3 InceptionTime leads (0.809 against 0.747). Both

are 47-trial problems, and as noted in Section 3.6.1, LP2 and LP3 are the same 47 trials under two labelings, so these are not independent losses. With a dozen test trials per fold, the standard deviation between folds on those tasks runs to 0.12–0.21, wider than any gap in the column.

Because LP2 and LP3 share their signals, the five-task mean and rank count one signal set twice. We therefore recomputed both over four independent signal sets, treating LP2 and LP3 as one set by averaging their scores. AdaStack stays first on both: mean rank 2.25 against 3.25 for the CNN-LSTM and 4.50 for the random forest, and mean macro F_1 0.827 against 0.816 and 0.804. Dropping either duplicate instead also leaves AdaStack first or tied for first on rank (2.75, tied with its two-member variant, without LP3; 2.00 without LP2). On mean macro F_1 without LP3, however, the random forest (0.798) and the two-member variant (0.800) are marginally ahead of AdaStack (0.797). The aggregate ordering is thus sensitive to how the duplicated signals are counted, which is a further reason to read the per-task columns rather than the summary row.

The ablation isolates the recurrent member. Removing the CNN-LSTM costs accuracy on four of the five tasks, most on LP1 (-0.012), but gains $+0.033$ on LP2, which leaves its mean over the five tasks marginally higher (0.788 against 0.787) and its rank clearly worse (3.40 against 3.20). Under the same corrected test as Table 5, none of the five per-task differences are significant: the smallest uncorrected p is 0.41 (LP1), all Holm-adjusted values across the five tasks are 1.00, and even the uncorrected Wilcoxon test finds nothing ($p \geq 0.10$). The two variants are statistically indistinguishable. We prefer the version with three members on the rank, rather than reading anything into 0.001 of macro F_1, and we do not claim that the recurrent member helps.

Table 5. AdaStack minus baseline macro F_1, paired over cross-validation folds; positive favors AdaStack. The last column is the smallest Holm-adjusted p across the five tasks, from the Nadeau–Bengio corrected resampled t-test; entries below 0.05 are marked with an asterisk.

Baseline	Δ LP1	Δ LP2	Δ LP3	Δ LP4	Δ LP5	min p
RF (summary stats)	+0.015	−0.069	+0.121	+0.006	+0.045	0.487
CNN-LSTM	+0.028	−0.020	−0.040	+0.014	+0.031	0.487
InceptionTime	+0.078	+0.031	−0.062	+0.074	+0.058	0.328
PatchTST	+0.091	−0.080	+0.078	+0.073	+0.030	0.079
TCN	+0.084	−0.028	−0.026	+0.114	+0.134	0.067
FCN	+0.074	+0.024	−0.000	+0.111	+0.075	0.061
Transformer	+0.097	−0.048	+0.180	+0.058	+0.077	0.246
1-NN DTW	+0.184	−0.020	+0.309	+0.144	+0.174	**0.035** *

How the weights are fitted mattered more than which members were in the stack. Scoring a weight grid inside each inner fold and summing the scores returned 0.671 macro F_1 on LP3, below the 0.815 of its own best member, because an inner fold there holds about a dozen trials. Pooling out-of-fold posteriors and fitting the weights once on the whole training fold raised the same configuration to 0.806. An ensemble is only as good as the data its combiner sees, which is easy to overlook when the combiner has three parameters.

Table 5 gives the paired comparisons. AdaStack beats 1-NN DTW significantly on LP1 and LP5 after Holm correction ($p = 0.035$ and $p = 0.044$); these are the only 2 of the 40 comparisons that survive correction, against 22 of 40 under the uncorrected Wilcoxon test. Against the stronger baselines nothing reaches significance: the smallest adjusted p anywhere else is 0.061, for FCN on LP4. This is a property of the benchmark rather than of the method. Fifteen folds of a 47-to-164-trial task cannot resolve differences of two or three points of macro F_1, and a study that reports such differences without correcting for fold dependence and multiple testing will find significance that is not there. We return to this in Section 5.2.

4.2. The Benchmark Limits the Comparison, Not the Architectures

It is worth asking why a benchmark of this kind resolves so little, because the answer bears on how its results should be read. One possibility is that the methods genuinely differ by only a point or two of macro F_1; another is that the tasks are too small for any protocol to tell them apart. Figure 3 separates the two by varying the amount of training data on LP5, the largest task, and watching how the ordering moves.

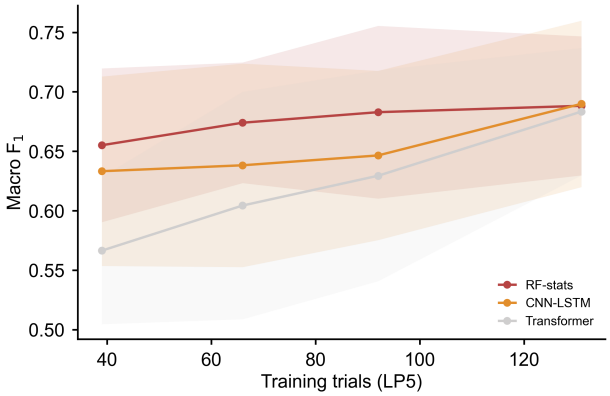


Figure 3. Learning curves on LP5. The learned models improve steadily with training data while the random forest saturates early, and the gap between them closes within the range the benchmark provides. Bands are one standard deviation over folds.

The learning curves are informative in two ways. First, they are still rising at the full training set for both learned models, so none of the methods has saturated: the CNN-LSTM gains 0.057 macro F_1 between 39 and 131 training trials, and the Transformer variant we also measured gains 0.117, against 0.033 for the random forest. The forest is flat almost from the start, which is what a strong feature prior buys and also what caps it. Second, the ordering changes within the range we can measure. At 39 training trials the random forest leads by 0.089; by 131 it leads by 0.005, and the CNN-LSTM has drawn level. Whatever ranking a study reports on these tasks is therefore a statement about a particular data scale rather than about the architectures in general.

This is the concrete reason we present AdaStack's lead as a lead in rank rather than as a demonstration of superiority. It also suggests where the field's effort would pay off: a public assembly monitoring dataset an order of magnitude larger would separate these methods, and until one exists, comparisons on the current one should be reported with paired, statistics that correct for fold dependence and read with the confidence intervals in view.

4.3. Early Detection on the Simulated Cell

The public tasks are 15 samples long, which says nothing about how a monitor behaves on the continuous streams a real cell produces. The simulated cell of Section 3.6.2 supplies those: 600-sample trials with four modalities and overlapping failure conditions. Because a monitor that only becomes accurate once the outcome is already mechanically determined cannot help a controller, we measure accuracy as a function of the observation horizon rather than at the end of the trial (Figure 4).

Accuracy rises from 0.50 at 0.25 s to 0.88 at 6 s against a chance level of 0.20, so the wrench and proprioceptive streams carry most of the information about the fault well before the trial ends, which is the premise the closed-loop study depends on. Among the learners, the fully convolutional network [9] leads from 0.5 s onward, reaching 0.880 with the full 6 s, with InceptionTime close behind at 0.869; the CNN-LSTM leads only at the shortest

horizon. We therefore use the convolutional network as the monitor in the closed-loop study. Stacking gives nothing here (0.881 against 0.880 at 6 s): with 600 training trials one member dominates, the out-of-fold weight fit concentrates on it, and the stack reduces to that member. The construction pays off where its members genuinely disagree, which on our evidence is the small real datasets rather than this one.

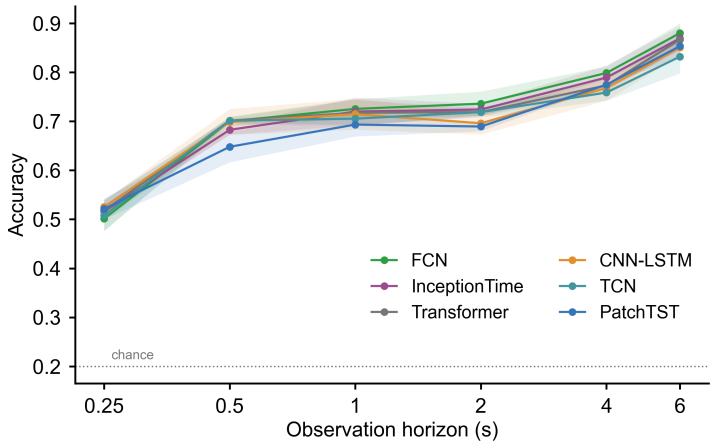


Figure 4. Early detection on the simulated cell: accuracy against observation horizon, means over three seeds with bands at one standard deviation. Chance is 0.20. The convolutional network leads from 0.5 s onward and is the monitor used in the closed-loop study.

4.4. Closing the Loop: The Policy, Not the Monitor, Is the Constraint

Monitoring accuracy says nothing about whether a monitor is useful once it drives a controller, which is the question a practitioner faces. Table 6 evaluates seven strategies on the same 200 held-out trials, none of which were used to fit any policy.

Table 6. Closed-loop insertion. Every strategy is evaluated on the same 200 held-out trials, none of which were used to fit any policy. Full seating requires 95% of hole depth; a damage event is any contact force above 120 N. Cost combines the four columns as in Equation (13). Brackets give 95% bootstrap intervals over trials (10,000 resamples, stratified by class). Rows marked † are new in this revision: the spiral search optimized with the same optimizer, budget and seeds as the monitor-driven policy, and the optimized policy driven by a windowed AdaStack monitor instead of the convolutional network. Best per column in bold; lower is better for force, cycle, damage and cost.

Strategy	Seating	Peak Force (N)	Cycle	Damage	Cost
Position control	0.780 [0.760, 0.795]	102.6 [99.0, 106.4]	**269**	0.325	1.223
Fixed-gain hybrid	0.295 [0.260, 0.330]	22.6 [22.0, 23.1]	998	**0.000**	0.939
Hybrid + always search	0.785 [0.765, 0.800]	31.6 [31.2, 32.0]	814	**0.000**	0.446
Hybrid + optimized search †	**0.790** [0.775, 0.800]	18.3 [18.1, 18.5]	784	**0.000**	**0.396**
Monitor-driven hybrid control					
hand-written, learned monitor	0.330 [0.290, 0.370]	19.5 [18.8, 20.1]	1000	**0.000**	0.895
hand-written, true state	0.520 [0.480, 0.560]	13.5 [13.2, 13.9]	897	**0.000**	0.670
optimized, learned monitor	0.765 [0.740, 0.785]	17.2 [16.8, 17.7]	809	**0.000**	0.421
optimized, AdaStack monitor †	0.735 [0.705, 0.765]	14.4 [13.9, 15.0]	829	**0.000**	0.446
optimized, true state	0.735 [0.705, 0.765]	**12.2** [12.0, 12.4]	792	**0.000**	0.434

Start with the two references that use no state estimate. Pure position control seats often (0.780) but is unusable: a stiff position loop drives the peg with a mean peak contact force of 102.6 N and exceeds our 120 N damage threshold in 32.5% of trials, including on nominal parts. Fixed-gain hybrid control is gentle (22.6 N, no damage) but seats only 0.295, because it presses straight down and never searches. Superimposing a spiral search that

ignores the estimated state raises seating to 0.785 at 31.6 N, which is the strongest reference in the table.

Now the monitor-driven rows, and the result that shaped the rest of this work. With the recovery policy we wrote by hand, the learned monitor seats only 0.330, barely above fixed-gain control, and giving the same policy the *true* fault state lifts it only to 0.520, still well below the search that uses no state at all. A perfect diagnosis was worth less than not diagnosing and simply searching. The constraint was therefore not perception but the reaction: our policy written by hand reduced the force limit to 0.40–0.45 of nominal while searching, which presses the peg against the face plate hard enough that lateral motion stalls against friction.

Optimizing the twelve free policy parameters of Equation (12) against the cost in Equation (13) removes that constraint. Driven by the same learned monitor, the optimized policy seats 0.765, matching the search's 0.785 while cutting mean peak contact force from 31.6 N to 17.2 N, a reduction of 46%, at a slightly shorter cycle and with no damage events. The paired bootstrap interval for the seating difference, -0.020 $[-0.050, 0.010]$, includes zero (McNemar $p = 0.34$, seven against three discordant trials), while the force reduction, -14.4 N $[-14.8, -14.0]$, is far outside the noise. Given the true state instead, the same procedure reaches 0.735 seating at 12.2 N, a 61% force reduction and the lowest force in the table, at a seating cost that is now significant (-0.050 $[-0.085, -0.015]$, McNemar $p = 0.021$).

The same gain without the monitor. The search that ignores the state was, in the original submission, run with hand-set parameters while the monitor-driven policy was optimized, which favours the latter. Optimizing the search's own three parameters with the same optimizer, budget, seeds and cost gives a light-touch spiral ($\phi = 0.44$, $\rho = 5.6$ mm, $\kappa = 0.95$) that seats 0.790 at 18.3 N with a cost of 0.396, the best in Table 6. Against it, the monitor-driven optimized policy seats 2.5 points less (-0.025 $[-0.045, -0.005]$; all 5 discordant trials favour the search, McNemar $p = 0.06$) and saves only 1.1 N of peak force ($[-1.5, -0.7]$ N, 6%). Per class, the monitor-driven policy loses its seating on high-friction trials (0.825 against 0.950), where misclassified windows lower the force limit too far, and gains force mainly on debris trials (14.7 N against 16.5 N), where it sometimes recognizes the obstruction and holds. The true state shows what a perfect monitor could add: on debris trials it holds at once and the peak force drops to 1.1 N, and over all trials it reaches the lowest force in the table. The learned monitor captures little of that. So the paper's positive result has to be stated more narrowly than we first did: in this cell the recovery policy was the binding constraint, and optimizing it cut contact force by 42–46% at matched seating, but almost all of that gain comes from the optimized light touch and wide search, which a controller without a monitor can apply to every trial.

The configuration in the loop. Figure 1 shows AdaStack as the monitor, but the closed-loop rows above use the fully convolutional network alone, which was the strongest single monitor on the simulated cell. We also trained a windowed AdaStack on the same 0.5 s windows and split (random forest, the same convolutional network and a CNN-LSTM; weights fitted on the validation windows as in Equations (7) and (8)). Its window-level macro F_1 on the monitor test split was 0.765 against 0.769 for the convolutional network alone, with most of the weight on the CNN-LSTM (0.83). Behind the same optimized policy, not re-optimized for it, it seated 0.735 at 14.4 N (cost 0.446), three points below the convolutional monitor ($[-0.055, -0.005]$, McNemar $p = 0.07$) with 2.8 N less force. Stacking therefore does not help on the simulated cell, either for window accuracy or in the loop, which agrees with Section 4.3.

The parameters the optimizer found are mechanically interpretable and are not what our hand design produced (Table 7). Given the true state, the optimizer drives the force

limit *down* for all three recoverable fault states, to $\phi = 0.20$–0.30 of nominal against 0.40–0.60 by hand, and gives high friction a 6.1 mm search where the hand design had none. The sensible recovery is a light touch and a wide search.

Table 7. Recovery-policy parameters. *By hand*: the policy we wrote, which is also the starting point of both optimizations. *True state*: optimized with the true fault state in the loop (Table 6, "optimized, true state"). *Learned monitor*: optimized with the convolutional monitor in the loop, the deployable configuration (Table 6, "optimized, learned monitor"; the AdaStack row uses the same parameters). ϕ_s multiplies the force limit, ρ_s caps the spiral radius in mm and κ_s multiplies lateral stiffness (Equation (12)), within the bounds $[0.2, 1.2]$, $[0, 10]$ mm and $[0.3, 1.2]$. For comparison, the state-independent search optimized in the same way applies $\phi = 0.44$, $\rho = 5.59$ mm and $\kappa = 0.95$ to every trial. The hand-written policy pressed harder while searching and seated fewer than a quarter of angular jams even with the true state.

	By Hand			**True State**			**Learned Monitor**		
Quality State	ϕ_s	ρ_s	κ_s	ϕ_s	ρ_s	κ_s	ϕ_s	ρ_s	κ_s
Normal	1.00	0.00	1.00	0.99	3.81	1.18	1.10	6.60	0.95
Lateral misalignment	0.45	8.00	0.60	0.20	6.57	0.39	0.48	7.54	0.47
Angular jam	0.40	3.00	0.50	0.26	3.86	1.12	0.21	4.57	0.69
High friction	0.60	0.00	0.80	0.30	6.12	0.44	0.33	6.53	0.34
Debris in seat	hold position (not optimized)								

The deployable policy, optimized with the learned monitor in the loop (right-hand columns of Table 7), differs from the true-state policy in two entries that also hold in the repeat run described below. Its normal-state row searches farther (6.6 mm, and 4.5 mm in the repeat, against 3.8 mm), so a fault window that the monitor reads as normal still gets a wide search, and its lateral-misalignment row presses harder ($\phi = 0.48$, and 0.85 in the repeat, against 0.20). Every row of this policy searches, with caps of 4.6–7.5 mm, close to the 5.6 mm that the optimized state-independent search applies to every trial; what changes with the reported state is mainly the force limit and the stiffness. This fits the small gap between the two controllers in Table 6.

Two further details are worth noting. The policy optimized under the learned monitor seats *more* often (0.765) than the one optimized under the true state (0.735) when each is evaluated in its own setting, because the former is selected on trials where the state is sometimes misread, and the wider search in its normal row is one visible result. Given the true state, the same policy seats 0.725 at 13.7 N, close to the true-state policy, so this caution costs little when perception is perfect. A policy tuned against perfect perception is not the policy to deploy behind an imperfect monitor. And the optimized policy does not beat the search on raw seating; it matches it. The gain is on force and, marginally, on cycle time, which matters when the parts rather than the cycle are the expensive item. The optimizer itself is noisy: re-running the baseline search with a different random stream gave a policy that seats 0.785 at 21.7 N (cost 0.413 against 0.421), so differences in cost below about 0.01, or in force below about 4 N, between two optimized policies should not be over-read. Nor should single entries of Table 7. The repeat run reached nearly the same cost with some entries far from those listed (a 1.6 mm cap for high friction, for example), so the table shows one good setting, not a unique optimum.

Figure 5 shows the trade-off and the optimization trace, and Figure 6 shows two individual trials under the policy written by hand, where the two failure mechanisms are visible: on a lateral trial the monitor posterior never settles, so the search radius resets before it can grow far enough to reach the aperture, while on an angular trial the posterior is confidently correct but the policy written by hand grants that state only a 3 mm search. One of those is a perception problem and the other is not, which is the distinction the optimization acts on.

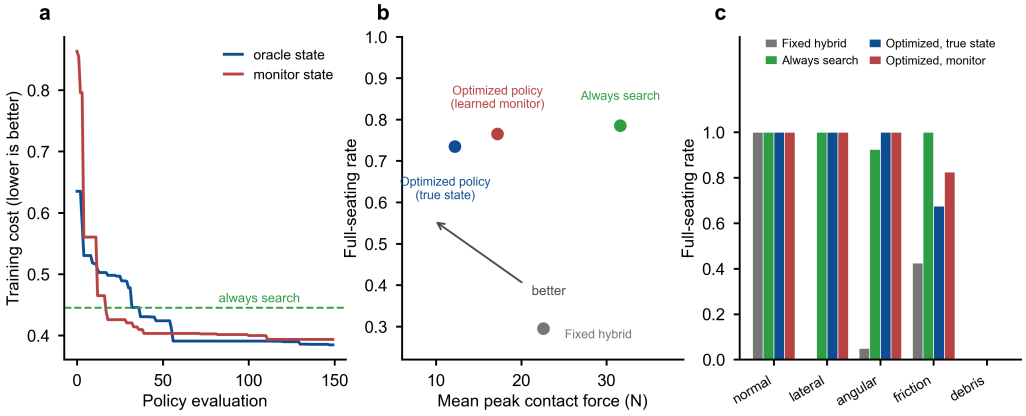


Figure 5. Optimizing the recovery policy. (**a**) Best training cost so far against policy evaluations, under the true state and under the learned monitor; the dashed line is the search that ignores the state. (**b**) Seating rate against mean peak contact force on held-out seeds; up and to the left is better. (**c**) Seating rate per class. The optimized policies recover lateral misalignment and angular jam, which fixed-gain control does not.

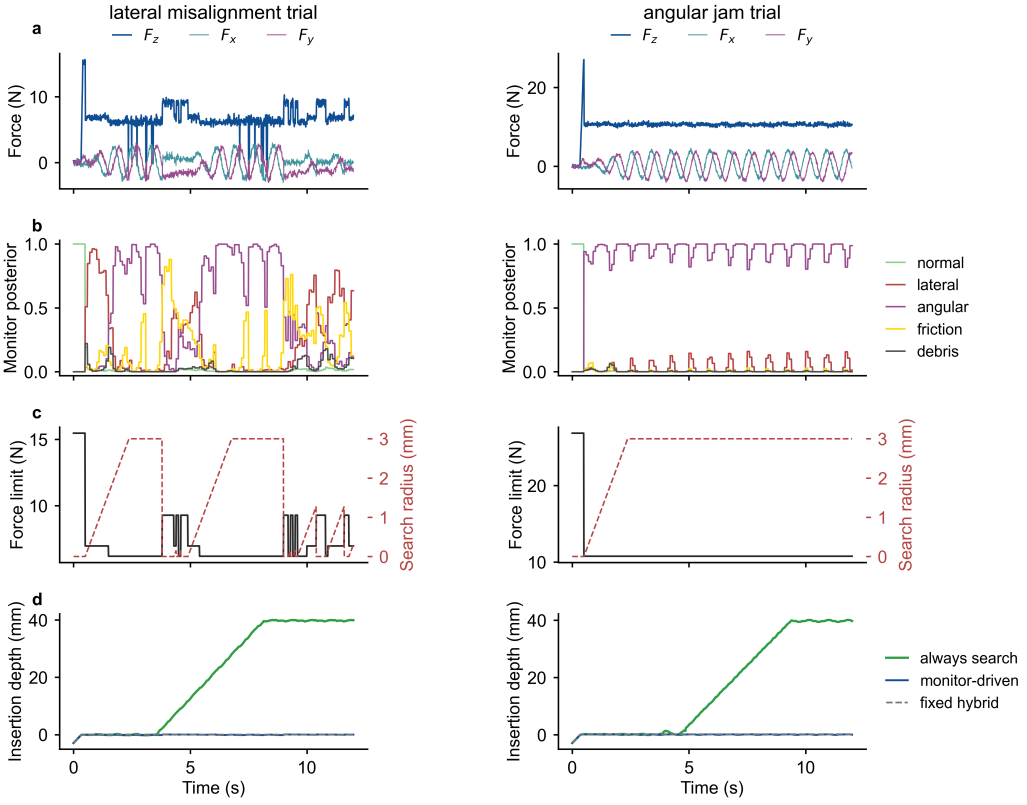


Figure 6. Two individual trials under the policy written by hand, showing why it underperformed. Left, lateral misalignment; right, angular jam. (**a**) contact wrench; (**b**) monitor posterior over quality states; (**c**) the resulting force limit and spiral radius; (**d**) insertion depth under three strategies. The search that ignores the state (green) seats in both cases.

4.5. Sensitivity to the Cost Weights

Table 8 and Figure 7a repeat the optimization of both controllers under the three alternative weightings of Section 3.4. The qualitative result for the state-conditioned policy holds under every weighting: it seats 0.765–0.780, within 2 points of the hand-set search's 0.785, at 17.2–21.6 N of mean peak force against the search's 31.6 N, and it beats the hand-set search on cost under each set of weights (0.531 against 0.667 force-dominant, 0.832

against 0.853 cycle-dominant, 0.274 against 0.281 seating-dominant). The margin is smallest when seating dominates, because then force is nearly free and the hand-set search already seats well.

Table 8. Cost-weight sensitivity. For each weighting, the state-conditioned policy (12 free parameters, learned monitor) and the state-independent spiral search (3 free parameters) are optimized with the same optimizer, budget and optimization seeds, and evaluated on the same 200 held-out trials. Costs are computed under the row's own weights; lower is better, and the lower of each pair is in bold. w_F and w_t are the force and cycle-time weights of Equation (13); the damage weight is 2.0 throughout.

		State-Conditioned, Monitor			State-Independent Search		
Weighting	(w_F, w_t)	**Seat**	**Force (N)**	**Cost**	**Seat**	**Force (N)**	**Cost**
Baseline (Equation (13))	(0.30, 0.20)	0.765	17.2	0.421	0.790	18.3	**0.396**
Force-dominant	(1.00, 0.20)	0.780	17.8	0.531	0.740	12.1	**0.514**
Cycle-dominant	(0.30, 0.80)	0.765	21.6	0.832	0.775	17.6	**0.801**
Seating-dominant	(0.10, 0.05)	0.780	20.4	0.274	0.790	18.3	**0.261**

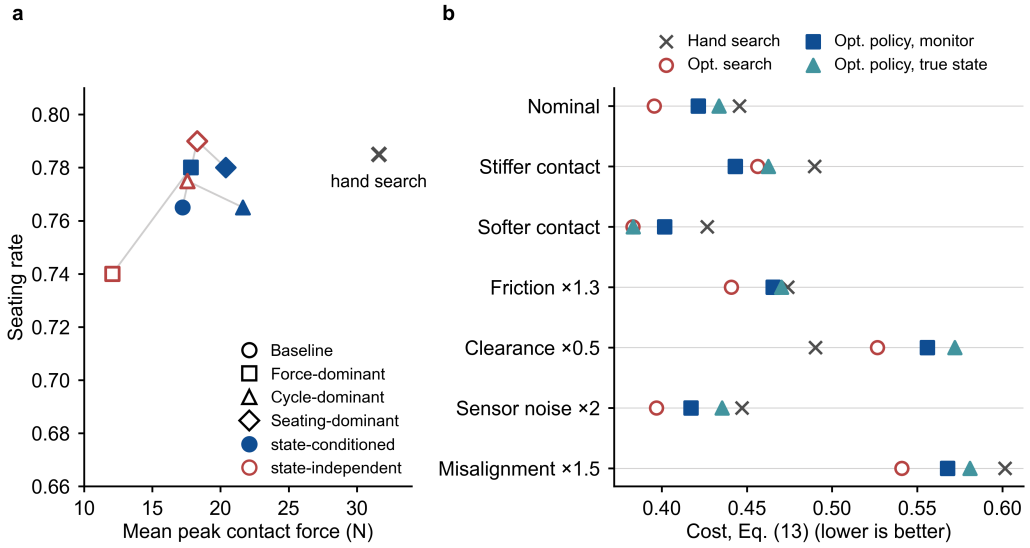


Figure 7. Cost-weight sensitivity and robustness. (**a**) Seating rate against mean peak contact force on the 200 held-out trials for the state-conditioned policy (filled, learned monitor) and the state-independent search (open), each optimized under four weightings; grey lines join the two controllers optimized under the same weights, and the cross is the hand-set search. Under the baseline and seating-dominant weightings the search converges to the same parameters, so its circle lies under the diamond. (**b**) Cost of Equation (13) under six shifted operating conditions, with no retraining or re-optimization; lower is better.

The comparison with the optimized state-independent search is also stable, and it goes the other way: the search has the lower cost under all four weightings, by 0.013 to 0.031. The search's own optimum barely moves. The baseline and seating-dominant weightings converge to the same parameters ($\phi = 0.44$, $\rho = 5.6$ mm, $\kappa = 0.95$), the cycle-dominant one to almost the same, and only the force-dominant weighting moves it to a lighter, softer search ($\phi = 0.30$, $\kappa = 0.52$) that trades 5 points of seating for 12.1 N. The conclusion that the optimized reaction is preferable to the hand-written one therefore does not depend on our weights; the conclusion that conditioning it on the learned monitor helps is not supported under any of them.

4.6. Robustness to Operating Conditions

Table 9 and Figure 7b evaluate the policies under six shifted operating conditions without retraining the monitor or re-optimizing any policy, so every shift is out of distribution for both. Three observations hold across them. First, the optimized policies keep their force advantage everywhere: mean peak force stays between 11.8 and 22.7 N against 31.1–34.9 N for the hand-set search, and no force-limited strategy records a damage event, while position control's damage rate rises to 0.555 with stiffer contact and 0.590 with larger misalignment. Second, the monitor is robust to sensor noise and contact stiffness: doubling the force/torque noise leaves the monitor-driven policy at 0.770 seating, and with stiffer contact it has the lowest cost of all strategies (0.443 against 0.456 for the optimized search). Third, the optimized light touch has a weak point. With the clearance halved, all three optimized policies lose 8–14 points of seating against the hand-set search (0.610–0.670 against 0.750), and the hand-set search has the lowest cost. A force limit and search radius tuned for one clearance range do not transfer to a tighter one; in a real cell the reaction would need re-tuning, or a clearance-dependent schedule, whenever the part tolerance changes. Larger misalignment lowers the seating of the four search-based strategies by 13–14 points without changing their order by cost.

Table 9. Robustness to shifted operating conditions. Each cell gives seating rate / mean peak force (N) on the same 200 held-out trial seeds with the condition shifted; the monitor is not retrained and no policy is re-optimized. Stiffer and softer contact halve and double the contact time constant (0.006 s nominal); the other shifts scale the sampled friction coefficient, clearance (with the localization error, which is defined in clearances), force/torque noise, and lateral and angular misalignment. The last column is position control's damage rate.

Condition	Hand Search	Opt. Search	Opt. Policy, Monitor	Opt. Policy, True State	Pos. Dmg.
Nominal	0.785/31.6	0.790/18.3	0.765/17.2	0.735/12.2	0.325
Stiffer contact	0.755/34.9	0.745/22.7	0.755/20.4	0.715/14.8	0.555
Softer contact	0.795/31.1	0.795/17.8	0.780/16.4	0.780/11.8	0.030
Friction ×1.3	0.760/31.5	0.750/18.4	0.725/17.3	0.705/12.3	0.350
Clearance ×0.5	0.750/31.8	0.670/18.2	0.645/17.6	0.610/12.2	0.410
Sensor noise ×2	0.785/32.1	0.790/18.8	0.770/17.5	0.735/12.8	0.325
Misalignment ×1.5	0.645/31.7	0.660/18.7	0.635/17.6	0.605/12.5	0.590

5. Discussion

5.1. Why Combining Learners Helped and Specialized Structure Did Not

The monitoring result has a shape worth naming. What improved accuracy was not a mechanism tailored to contact physics but a combiner that lets the data choose among general-purpose learners, and the improvement came mostly from fitting that combiner properly rather than from adding members to it. On LP3 the difference between a badly fitted stack and a well fitted one was 0.135 macro F_1, larger than the gap between our method and any baseline anywhere in Table 4.

We think the reason is that the public tasks are 15 samples long. Structure that represents a temporal lag, or separates modality timescales, needs sequences that resolve those effects; at this length a convolution with a wide enough receptive field already spans the informative structure, and the extra machinery buys nothing while adding parameters. A mechanism can be physically well motivated and still be the wrong thing to build, when the quantity it represents is already implicitly available to a simpler model. The corollary is that the sensible default for assembly monitoring at this data scale is a small convolutional network or a feature-based classifier, and that the burden of proof sits with anything more elaborate.

5.2. What the Public Benchmark Can and Cannot Decide

The public benchmark remains useful as a realism check, since the data come from a real manipulator with real failure modes, but it cannot arbitrate between modern architectures, because the confidence intervals it supports are wider than the differences between them. Our 22-versus-2 result (22 of 40 comparisons significant under an uncorrected test over folds, 2 under a corrected one, both against the weakest baseline) is a concrete measure of how easily that limitation can be missed, and the fact that LP2 and LP3 are the same signals under two labelings further reduces the benchmark's effective diversity from five tasks to four. Studies that report a single split, or that aggregate all five tasks into one number, are likely to report architecture differences that the data cannot support.

5.3. What This Means for Building an Assembly Cell

Read as engineering guidance, our results point in a specific direction. All of the numbers below come from the simulated cell, and each recommendation should be read as "in this simulated cell" until it has been checked on hardware (Section 5.4).

Prefer force-limited insertion to stiff position control. Position control seated 0.780 of trials but averaged 102.6 N of peak force and crossed our damage threshold in 32.5% of them, including on nominal parts; every force-limited strategy stayed below 32 N with no damage events. Where the parts are the expensive item, that trade is not close.

Add a continuous search before adding a monitor. A slowly growing spiral superimposed on force-limited insertion raised seating from 0.295 to 0.785 with no perception at all, nothing to train and behavior that is trivial to certify. The learned monitor, driving the policy we wrote by hand, reached 0.330.

Then optimize the reaction, not the monitor. Tuning the twelve free policy parameters against an explicit cost recovered the search's seating rate at 17.2 N instead of 31.6 N in simulation. If contact force matters, that is where the return is, and the optimizer is a few dozen lines. Optimize the search on its own first: in our cell a state-independent spiral with three optimized parameters did as well as the twelve-parameter state-conditioned policy, and it needs no monitor. Re-tune it when the clearance changes, because the optimized light touch lost seating when we halved the clearance.

If a monitor is wanted, start small, and know what it is for. A fully convolutional network was the best monitor on the simulated cell at almost every horizon, and a random forest on summary statistics was strongest on the real benchmark; the ensemble of the two ranked first overall. All trained in seconds. The convolutional network needs about 1 ms per decision on one CPU core, far inside the 100 ms budget. In our loop the monitor's clearest use was not better insertion but recognizing the unrecoverable state: with the true state, the hold response cut the peak force on debris trials from 16.5 N to 1.1 N. A monitor aimed at that decision, and at reporting the quality state for traceability, is easier to justify than one aimed at improving the seating rate.

Budget the cycle time around the slowest recovery. A recovery that reduces the force limit necessarily slows insertion, so a controller that is gentler per unit time can still fail more often if the cycle is cut short. Several of our own early conclusions were artifacts of that interaction.

5.4. Threats to Validity

We list the limitations that we think most constrain the conclusions.

The closed-loop results come only from simulation. The controller study runs in MuJoCo, and while the contact model, the force-limited velocity loop and the sensor noise are chosen for realism, simulation is not evidence about a physical cell. Contact stiffness, stick–slip and sensor dynamics are all idealized. The correct reading of Section 4.4 is that

the monitor's output carries enough information to improve a controller *in this model*, not that the improvement transfers unchanged to hardware.

The simulated data are generated by the same model that is controlled. Train and test trials are drawn from the same simulator, so the monitoring results do not measure sim-to-real transfer. We mitigate this by randomizing clearance, friction, insertion speed, force limit and sensor noise per trial, and by keeping the parameter ranges within each class overlapping, but a sim-to-real gap remains untested.

Hyperparameters are untuned and shared. Every neural model uses the same width, depth, optimizer and schedule, which makes the comparison architectural and treats all models equally, but leaves none at its own optimum. A tuned AdaStack member might fare better, and so might a tuned baseline. We report the untuned comparison because it is the one we can make fairly within the compute budget.

The form of the recovery policy is fixed by hand. The optimizer tunes the numbers in the policy table, but the table itself, one force-limit multiplier, search cap and stiffness multiplier per state with a fixed spiral rate and a hold action for debris, was chosen from mechanical reasoning. A richer reaction, for example one that also uses the posterior confidence or the time since the last state change, could do better, and the absolute closed-loop numbers depend on this choice.

The optimized policy is tuned in one simulator against weights we chose. The cost coefficients encode a judgement that damage matters most and cycle time least; a cell with different economics would get a different policy. Section 4.5 shows that the main conclusions do not change across four weightings, but a cell with a very different cost structure, for example one where cycle time dominates by an order of magnitude, was not tested. The search also covered only twelve parameters of a fixed policy form.

The simulated clearances are coarse. The introduction motivates the problem with industrial clearances of tens of micrometres, while the simulated cell uses 0.8–1.8 mm per side (0.4–0.9 mm in the tight-clearance test). MuJoCo's soft-contact model with a 0.5 ms step does not resolve micrometre gaps reliably, so we did not simulate them. The fault conditions are specified relative to the clearance (a lateral error of 1.2–3.5 clearances), which keeps the geometry of the problem similar across scales, and halving the clearance did not change the ordering of the strategies (Section 4.6). Still, at true precision-assembly clearances chamfer geometry, surface finish and stick–slip matter far more, and the force levels, search radii and seating rates we report should not be read as predictions for such parts.

Benchmark composition. Because LP2 and LP3 share identical sensor data, the five public tasks provide four independent signal sets, which further limits the power of any aggregate across tasks such as a mean rank or a critical difference diagram. We report per-task results for this reason and avoid aggregating LP2 and LP3 into a single score.

6. Conclusions

We asked whether monitoring accuracy or the recovery policy limits contact-rich robotic assembly, and the two halves of the answer point in opposite directions.

On monitoring, combining learners helped and specialized structure did not. AdaStack, which mixes a random forest, a convolutional network and a CNN-LSTM with weights fitted on pooled out-of-fold posteriors, attained the best mean rank (3.20) and the best mean macro F_1 (0.787) among ten methods under one protocol, and led on three of the five tasks. It significantly outperformed dynamic time warping on two tasks after correction, while differences against the stronger baselines stayed inside the resolution of a benchmark whose tasks hold 47 to 164 trials. How the stack weights were fitted mattered more than which members were in it: fitting them per inner fold rather than on pooled out-of-fold posteriors cost 0.135 macro F_1 on the smallest task.

On control, a small amount of work went a long way, in simulation. With a recovery policy written by hand, a spiral search that ignores the state outperformed the learned monitor and an oracle given the true state, which located the constraint in the policy rather than the diagnosis. Optimizing twelve policy parameters against an explicit cost, with disjoint optimization and evaluation seeds, produced a controller that matches the search's seating rate (0.765 against 0.785) while cutting mean peak contact force by 46% (17.2 N against 31.6 N) using the learned monitor, and by 61% given the true state. The parameters it found are mechanically interpretable and were not what a plausible hand design produced: recovery wants a light touch and a wide search, not the firmer press we had written. That lesson, however, does not need the monitor: the spiral search with its own three parameters optimized in the same way seated 0.790 at 18.3 N, and it matched or beat the monitor-driven policy under four cost weightings and five of six shifted operating conditions. The gain came from optimizing the reaction, not from conditioning it on the estimated state, and a perfect state estimate would add mainly a lower force, most visibly on the unrecoverable debris trials.

For practitioners the recommendation is to run force-limited insertion with a continuous search, spend the next increment of effort on the reaction rather than the classifier, optimize a state-independent reaction before adding a state-dependent one, and use an oracle substitution to check whether better perception could help before investing in it. For the field, we would ask that small assembly benchmarks be reported with paired statistics that correct for fold dependence, that simple convolutional and feature baselines be included as a matter of routine, and that closed-loop papers report their policy with the care they give their network. The limitations are real: the closed-loop results come only from simulation with millimetre rather than micrometre clearances, the cost weights are ours (although the conclusions held under three alternatives), and the public monitoring data remain too small to separate architectures at all.

Author Contributions: Conceptualization, F.Z. and Q.T.; methodology, F.Z. and Q.T.; software, F.Z. and R.X.; validation, F.Z., R.X. and Q.T.; formal analysis, F.Z.; investigation, F.Z. and R.X.; resources, Q.T.; data curation, F.Z. and R.X.; writing—original draft preparation, F.Z.; writing—review and editing, R.X. and Q.T.; visualization, F.Z. and R.X.; supervision, Q.T.; project administration, Q.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The UCI Robot Execution Failures dataset is publicly available from the UCI Machine Learning Repository. All code for the simulated cell, the models, the training and evaluation protocol, the policy optimization and the figures, together with the generated dataset, the trained monitors and all result files, will be deposited in a public repository with a citable DOI before publication; until then they are available from the corresponding author on request. The simulator is deterministic given a trial seed, so every simulated trial in this paper can be regenerated from the seed ranges in Table 3.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Raibert, M.H.; Craig, J.J. Hybrid Position/Force Control of Manipulators. *J. Dyn. Syst. Meas. Control* **1981**, *103*, 126–133. [CrossRef]
2. Yan, W.; Yu, Y. A Study on the Force/Position Hybrid Control Strategy for Eight-Axis Robotic Friction Stir Welding. *Metals* **2025**, *15*, 442. [CrossRef]

3. Sha, S.; Wang, N.; Ma, S.; Yu, B.; Li, C.; Jiang, L.; Liu, G.; Qin, Z.; Zhao, R. Research on force-position decoupling control technology of bonnet polishing of robotic arm. *Precis. Eng.* **2025**, *94*, 315–329. [CrossRef]
4. Wang, B.; Zhang, J.; Wang, S.; Wu, D. A Robust and Compliant Robotic Assembly Control Strategy for Batch Precision Assembly Task. *IEEE Trans. Autom. Sci. Eng.* **2026**, *23*, 5502–5515. [CrossRef]
5. Qu, J.; Bi, D.; Liu, Y. Adaptive multimodal fusion-driven reinforcement learning for robotic peg-in-hole assembly. *J. Intell. Manuf.* **2026**. [CrossRef]
6. Li, S.; Li, B.; Yu, J. Integrating admittance control and residual reinforcement learning for robotic assembly with visual and force feedback. *Robotica* **2025**, *44*, 37–51. [CrossRef]
7. Li, B.; Wang, Z. Two-stage DRL with hybrid perception of vision and force feedback for lunar construction robotic assembly control. *Acta Astronaut.* **2025**, *229*, 357–373. [CrossRef]
8. Ren, S.; Zeng, C.; Yang, C.; Wang, N. Toward high-precision robotic assembly in large workspaces using multimodal reinforcement learning. *Robot. Intell. Autom.* **2026**, *46*, 321–332. [CrossRef]
9. Wang, Z.; Yan, W.; Oates, T. Time series classification from scratch with deep neural networks: A strong baseline. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 1578–1585. [CrossRef]
10. Ismail Fawaz, H.; Lucas, B.; Forestier, G.; Pelletier, C.; Schmidt, D.F.; Weber, J.; Webb, G.I.; Idoumghar, L.; Muller, P.A.; Petitjean, F. InceptionTime: Finding AlexNet for time series classification. *Data Min. Knowl. Discov.* **2020**, *34*, 1936–1962. [CrossRef]
11. Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.A. Deep learning for time series classification: A review. *Data Min. Knowl. Discov.* **2019**, *33*, 917–963. [CrossRef]
12. Nie, Y.; Nguyen, N.H.; Sinthong, P.; Kalagnanam, J. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. *arXiv* **2023**, arXiv:2211.14730.
13. Zhang, J.; Wu, Z.; Gan, H. Multivariate time series anomaly detection using neighbor relation-guided Transformer. *Pattern Recognit.* **2026**, *179*, 113835. [CrossRef]
14. Xie, X.; Liu, K.; Wang, Y.; Wu, M.; Zhang, H.; Wan, T. CAAE: Contrastive Adversarial AutoEncoder for multivariate time series anomaly detection. *Pattern Recognit.* **2026**, *179*, 113687. [CrossRef]
15. Darban, Z.Z.; Webb, G.I.; Pan, S.; Aggarwal, C.C.; Salehi, M. CARLA: Self-supervised contrastive representation learning for time series anomaly detection. *Pattern Recognit.* **2025**, *157*, 110874. [CrossRef]
16. Lan, Y.; Long, X.; Zeng, Y.; Chen, H.; Xiao, Y.; Yang, J. Causal mixture-of-experts for robust multimodal fusion. *Pattern Recognit.* **2026**, *180*, 114306. [CrossRef]
17. Chen, B.; Cao, L.; Liu, W.; Liu, J.; Zhao, B.; Liu, Y. Cross-modal attention fusion of RGB and skeleton for multimodal-driven video anomaly detection. *Pattern Recognit.* **2026**, *179*, 113815. [CrossRef]
18. Wang, D.; Guo, X.; Tian, Y.; Liu, J.; He, L.; Luo, X. TETFN: A text enhanced transformer fusion network for multimodal sentiment analysis. *Pattern Recognit.* **2023**, *136*, 109259. [CrossRef]
19. Li, Q.; Wan, X.; Guo, H.; Wang, W.; Liu, M.; Cui, Y. Uncertainty-aware multi-modal time series anomaly detection via reduced-order evidence fusion. *Pattern Recognit.* **2026**, *180*, 114212. [CrossRef]
20. Zhao, Y.; Li, B.; Zhang, X.; Zhang, L.; Lu, H. DecoupleMAD: Boosting sensitivity in multimodal anomaly detection via representation decoupling. *Pattern Recognit.* **2026**, *180*, 114025. [CrossRef]
21. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable DETR: Deformable Transformers for End-to-End Object Detection. *arXiv* **2021**, arXiv:2010.04159.
22. Lopes, L.S.; Camarinha-Matos, L.M. Feature Transformation Strategies for a Robot Learning Problem. In *Feature Extraction, Construction and Selection*; Springer: Boston, MA, USA, 1998; pp. 375–391. [CrossRef]
23. Lopes, L.S.; Camarinha-Matos, L. Learning a Taxonomy of Failures in Assembly. In *Making Robots Smarter*; Springer: Boston, MA, USA, 1999; pp. 161–184. [CrossRef]
24. Todorov, E.; Erez, T.; Tassa, Y. MuJoCo: A physics engine for model-based control. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 5026–5033. [CrossRef]
25. Lea, C.; Flynn, M.D.; Vidal, R.; Reiter, A.; Hager, G.D. Temporal Convolutional Networks for Action Segmentation and Detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognit (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1003–1012. [CrossRef]
26. Yuan, J.; Tian, Y. A Multiscale Feature Learning Scheme Based on Deep Learning for Industrial Process Monitoring and Fault Diagnosis. *IEEE Access* **2019**, *7*, 151189–151202. [CrossRef]
27. Wang, Y.; Wang, J.; Zeng, T.; Qi, J. Data-mining-based biomarker evaluation and experimental validation of SHTN1 for bladder cancer. *Cancer Genet.* **2024**, *288–289*, 43–53. [CrossRef] [PubMed]
28. Kawahara, D.; Kishi, M.; Kadooka, Y.; Hirose, K.; Murakami, Y. Integrating radiomics and gene expression by mapping on the image with improved DeepInsight for clear cell renal cell carcinoma. *Cancer Genet.* **2025**, *292–293*, 100–105. [CrossRef] [PubMed]

29. Lee, D.; Malacarne, S.; Aune, E. Explainable time series anomaly detection using masked latent generative modeling. *Pattern Recognit.* **2024**, *156*, 110826. [CrossRef]
30. Peng, G.; Han, L. 3D human pose estimation with decoupled spatio-temporal hybrid transformer. *Pattern Recognit.* **2026**, *180*, 114125. [CrossRef]
31. Korban, M.; Youngs, P.; Acton, S.T. A dynamic predictive transformer with temporal relevance regression for action detection. *Pattern Recognit.* **2025**, *166*, 111644. [CrossRef]
32. Wang, Z.; Yao, J.; Xu, M.; Jiang, M.; Su, J. Transformer-based network with temporal depthwise convolutions for sEMG recognition. *Pattern Recognit.* **2024**, *145*, 109967. [CrossRef]
33. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. *arXiv* **2017**, arXiv:1706.03762.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.