Safer Imitation by Construction: Cutting Robot Failures at the Behavioral Fork

Socrates Osorio*
Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA, USA
Email: socratesj.osorio@berkeley.edu

Joy Zheyun Yang*
University of Oxford
Oxford, United Kingdom
Email: joy@robots.ox.ac.uk

***Abstract*—As robot policies are increasingly trained on large, heterogeneous logs, those logs inevitably contain failures and near-misses. Including a failed trajectory wholesale is not only bad for task performance. It is a *safety* problem, because the policy can internalize and later reproduce the unsafe behavior that caused the failure. We argue that the cleanest place to remove this hazard is at the source, the training data, and that an under-exploited signal makes this possible. Whenever a log contains matched success/failure attempts from the same initial condition (a paired counterfactual), it already encodes the *behavioral fork*: the moment a trajectory crosses from still-recoverable behavior into the failure mode. Everything before the fork is benign, and the unsafe content is concentrated in the post-fork suffix. CoP-Miner (Counterfactual-Prefix Miner) mines this fork automatically from each pair using state proximity, action-window divergence, and future-state divergence, and curates each failed trajectory down to its safe pre-fork prefix, without a reward, cost function, or hand-specified constraint. On the canonical paired benchmark from RoboMimic Can [11], imitating full failures collapses closed-loop success to 0.40, while cutting at the mined fork recovers 0.87 and is competitive with discarding failures entirely. A late-boundary stress test isolates the hazard directly: keeping the post-fork suffix ($+16$ steps) collapses success from 0.87 to 0.56. Prefix-location and length-matched controls show the active ingredient is the per-pair mined boundary, not retained data volume, and the benefit is largest when clean demonstrations are scarce ($+20.4$ points at 50 pairs, p=0.009). We position fork mining as a data-level safety intervention that complements runtime monitoring, and outline how the same fork signal can serve as a deployment-time divergence detector.**

I. Introduction

Robot learning increasingly relies on large, heterogeneous interaction logs: teleoperation sessions, autonomous rollouts, and replays that mix clean successes with near-misses and outright failures. Standard imitation learning has no built-in way to separate the useful behavior in such a log from the harmful behavior. This is usually framed as a performance problem. We argue it is first a *safety* problem. A failed trajectory ends in the very outcome we want robots to avoid, such as a collision, a drop, or an unsafe motion near a person or object, and behavior cloning on that trajectory teaches the policy to reproduce the actions that led there. Discarding failures avoids this hazard but throws away expensive, often safety-relevant, pre-failure behavior, while imitating them wholesale injects the hazard directly into the policy. Neither extreme is satisfying as foundation-model-scale policies start to be deployed around people.

Approaches to embodied safety divide naturally into two complementary families: reducing risk *at the source* (the data, architectures, and objectives a policy is trained from) and reducing risk *at runtime* (monitoring, shielding, and safe fallback during execution). This paper is about the first family applied to the most basic ingredient of a learned policy: its demonstration data. We claim that an under-exploited structure in robot logs makes a precise, automatic safety intervention possible at the data level.

a) The behavioral fork.: Whenever a log contains matched success/failure attempts from the same initial condition, it encodes a natural counterfactual. The two attempts start from the same problem instance, take similar early actions, and then diverge. We call the moment of divergence the *behavioral fork*. The fork is exactly the boundary between safe and unsafe behavior for that instance: before it, the failed attempt is still doing essentially what the success did. After it, the failed attempt commits the mistake and executes the unsafe suffix that ends in failure. The safety hazard in a failed trajectory is therefore not spread uniformly. It is localized after the fork.

b) CoP-Miner.: We make this concrete with CoP-Miner, an automatic fork-mining pipeline. It aligns each success/failure pair, scores candidate fork points using state proximity, action-window divergence, and future-state divergence, selects a per-pair fork, and then truncates each failed trajectory at its fork. The result is a curated dataset that keeps the benign pre-fork prefix of every failure while removing the unsafe suffix: a *safety-by-construction* data intervention that requires no reward model, cost function, or hand-specified safety constraint, and that any standard learner can ingest unchanged.

c) Findings.: On the canonical paired manipulation benchmark, RoboMimic Can Paired [11], we report three findings. First, imitating full failures is harmful: closed-loop success collapses to 0.40 versus 0.87 when failures are cut at the mined fork. Second, a late-boundary stress test isolates the hazard: deliberately keeping 16 post-fork steps drops success from 0.87 to 0.56, showing that the post-fork suffix is the dangerous content. Third, prefix-location and length-

*Equal contribution.

matched controls show the gain comes from the per-pair mined boundary, not from retaining more data. The benefit is largest when clean successful demonstrations are scarce ($+20.4$ points over success-only behavior cloning at 50 pairs), the regime in which discarding failures is most costly.

d) Contributions.:

1) We frame learning from failure-containing logs as a data-level safety problem and identify the behavioral fork as the boundary that localizes the hazard.
2) We present CoP-Miner, a reward-free, constraint-free fork-mining primitive that curates failed demonstrations into their safe pre-fork prefixes.
3) We provide controlled evidence, including a late-boundary stress test that directly measures the cost of retaining the unsafe suffix, isolating the per-pair fork as the active ingredient.
4) We outline how the same fork score extends from a design-time curation step to a runtime divergence monitor, bridging the data and deployment views of safety, and list open problems for scaling to image- and language-conditioned policies.

II. Related Work

a) Learning from failed and imperfect demonstrations.: Prior work treats failures as informative negative evidence or learns confidence weights for imperfect data, including segment-level reuse of failed trajectories [7, 15, 8, 17, 16]. CoP-Miner is closest in spirit to segment-level reuse, but differs by using paired success/failure counterfactuals to mine a temporal fork and then keeping only the pre-fork prefix, framing the cut as a removal of unsafe content rather than as a reweighting.

b) Data quality and curation for robots.: Behavior cloning is brittle under closed-loop distribution shift [13], and recent analyses show that *which* demonstrations are included can matter as much as the policy class [2]. Robot-learning curation has been studied via influence functions and intervention residuals [1, 4]. We view curation through a safety lens: the goal is not only to improve return but to keep unsafe behavior out of the policy at the source, and to do so with a localized per-pair boundary rather than a global filter.

c) Suboptimal offline imitation and offline RL.: DemoDICE, discriminator-weighted BC, IQL, TD3-BC, and advantage-weighted regression use imperfect data via objective changes or transition reweighting [9, 18, 10, 6, 12]. Ranking- and preference-based methods infer reward from comparisons [3, 5]. CoP-Miner introduces no new learner: it is an upstream data transformation, and our supplementary checks show that even an offline-RL learner trained on the raw mixed log underperforms the same learner trained on the fork-curated data (Appendix).

d) Runtime safety.: The complementary family of methods monitors and shields policies during execution (uncertainty/OOD detection, safe fallback). We do not propose a new runtime monitor, but we observe that the fork score is itself a divergence signal between a current rollout and known safe behavior, and discuss in Sec. VI how the design-time primitive extends to deployment-time monitoring.

III. Method

A. Paired Setting

The input is a set of paired attempts. Each pair holds one successful trajectory $(o^s_{1:T_s}, a^s_{1:T_s})$ and one failed trajectory $(o^f_{1:T_f}, a^f_{1:T_f})$ recorded from the same or a near-matched initial state. The method requires only that the two attempts are progress-alignable. It assumes no reward, no cost, and no explicit safety specification.

B. Aligning the Pair

The default alignment maps each success index i to $j(i) = \mathrm{round}(i(T_f-1)/(T_s-1))$. A progress-alignment variant uses a banded dynamic-time-warping map over normalized observations [14]. Both alignments preserve the result (Sec. V-E).

C. The Fork Score

For each aligned index i, CoP-Miner computes three signals: current-state proximity $d_o(i) = \|o^s_i - o^f_{j(i)}\|_2$, local action-window divergence $d_a(i) = \|a^s_{i:i+H} - a^f_{j(i):j(i)+H}\|_2$ with $H{=}8$, and future-state divergence $d_+(i) = \|o^s_{i+\Delta} - o^f_{j(i)+\Delta}\|_2$ with $\Delta{=}12$. The raw fork score

$$r(i) = d_a(i)\,\exp\big(-d_o(i)/\sigma_o\big) \times \big(1+\exp(-(d_+(i)-m_+)/\mathrm{mad}_+)\big)^{-1} \quad (1)$$

rewards locations where the two attempts are still close in state, the local action chunks already disagree, and the consequence shortly afterward is large. Here σ_o is the median aligned observation distance and m_+, mad_+ are the median and median absolute deviation of d_+. Scores are smoothed with a width-5 moving average. The selected fork is the earliest index in the top score quintile that also has below-60th-percentile current-state distance and above-70th-percentile future divergence, with the score argmax as a fallback. This yields fork indices (τ_s, τ_f).

Conceptually, $r(i)$ targets the safety boundary directly: state proximity says "the mistake has not yet propagated", action divergence says "a different action is being taken", and future divergence says "the attempts are about to separate". Together they mark where behavior crosses from safe to unsafe.

D. Curation and Training

The mined fork τ_f is a hard truncation boundary: each failed trajectory is retained as a prefix of length $\max(12, \tau_f)$, dropping the unsafe suffix. The CoP-prefix dataset is all successful trajectories plus these safe prefixes. We compare against baselines that isolate alternative explanations: BC-all (success + full failures), BC-success (success only), a transition-budget-matched success-only control, a midpoint-prefix rule, a length-matched shuffled-prefix control that breaks the pair-specific fork, and a global-median-prefix rule using a single split-level cutoff. All policy methods use the same native RoboMimic BC-RNN/GMM stack (5-mode GMM head, 300 epochs, 50 closed-loop rollouts per run) and differ only in the dataset supplied.

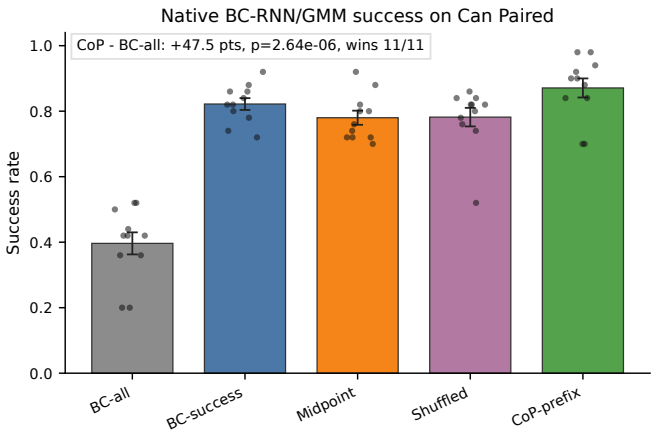

Fig. 1. Native BC-RNN/GMM closed-loop success on RoboMimic Can Paired. Cutting failures at the mined fork (CoP-prefix) strongly outperforms imitating full failures (BC-all) and also beats a midpoint-prefix rule and a length-matched shuffled-prefix rule. Error bars show standard error, and dots are individual runs.

TABLE I
NATIVE BC CLOSED-LOOP SUCCESS ON CAN PAIRED (11 MATCHED RUNS).

Method	Runs	Mean success	Std	Min	Max
BC-all	11	0.396	0.112	0.200	0.520
BC-success	11	0.822	0.060	0.720	0.920
Midpoint-prefix	11	0.780	0.072	0.700	0.920
Shuffled-prefix	11	0.782	0.094	0.520	0.860
CoP-prefix	11	0.871	0.096	0.700	0.980

IV. EXPERIMENTAL SETUP

We use RoboMimic Can Paired low-dimensional [11], which provides 100 task initializations each with one successful and one failed demonstration, the cleanest available case in which the paired-counterfactual assumption holds exactly. We build a pair manifest matching successes to nearest-initial-state failures (maximum recorded initial distance 2.5×10^{-16}). Splits are at the pair level so the success and failure from a pair never cross train/eval boundaries. The independent statistical unit is the matched split/train run. Paired comparisons use two-sided paired t-tests on seed-level success rates, with wins/losses/ties counting strict paired deltas.

V. RESULTS

A. Imitating Full Failures Is the Hazard

Imitating full failed trajectories is actively harmful: BC-all reaches only 0.396 closed-loop success, while cutting each failure at its mined fork (CoP-prefix) recovers 0.871 (paired $\Delta = +0.475$, $p=2.6\times10^{-6}$, wins 11/11; Table I, Fig. 1). Discarding failures entirely (BC-success) reaches 0.822, confirming that the full failed suffix, not the presence of failure data, is what damages the policy. Removing that suffix at the source restores safe, performant behavior while keeping the benign pre-fork content.

TABLE II
MATCHED PAIRED COMPARISONS FOR THE MAIN RESULT AND PREFIX CONTROLS.

Comparison	Runs	Method mean	Baseline mean	Delta	p-value	Wins/Losses/Ties
CoP-prefix - BC-all	11	0.871	0.396	+0.475	2.64e-06	11/0/0
CoP-prefix - BC-success	11	0.871	0.822	+0.049	0.200	7/4/0
CoP-prefix - Midpoint	11	0.871	0.780	+0.091	0.020	8/2/1
CoP-prefix - Shuffled	11	0.871	0.782	+0.089	0.037	9/2/0
BC-success - BC-all	11	0.822	0.396	+0.425	3.27e-08	11/0/0

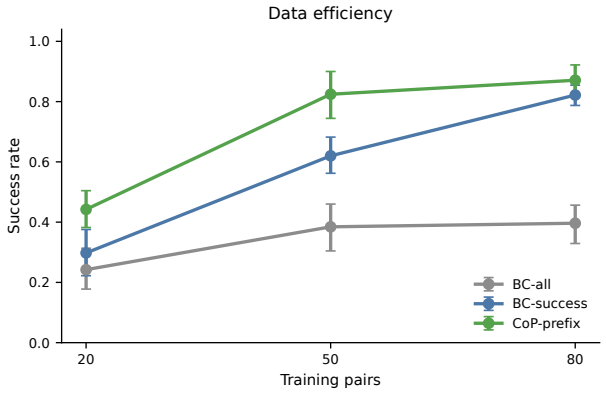

Fig. 2. Data efficiency across 20, 50, and 80 training pairs. Recovering safe pre-fork behavior helps most when clean successful demonstrations are scarce.

B. The Hazard Is the Post-Fork Suffix

The clearest safety statement comes from a late-boundary stress test. If we keep the mined prefix but deliberately extend it 16 steps past the fork, re-admitting the unsafe suffix, closed-loop success drops from 0.870 to 0.563 (-0.307, p=0.013, losses 6/6; Table IV). This is a direct measurement of the cost of retaining unsafe content: the post-fork chunk is the harmful part, and the fork is the boundary that separates it from safe behavior.

C. The Per-Pair Fork Is the Active Ingredient

Two prefix-location controls retain comparable volumes of failed data but place the boundary differently. CoP-prefix beats a midpoint-prefix rule by $+0.091$ (p=0.020) and a length-matched shuffled-prefix rule, which preserves the CoP-prefix prefix-length distribution but reassigns prefixes across pairs, by $+0.089$ (p=0.037; Table II). Equating transition budgets between success-only BC and CoP-prefix preserves the CoP-prefix advantage ($+0.082$, p=0.007), and a global-median-prefix rule using a single split-level cutoff is worse than per-pair forks ($+0.046$, p=0.045; Appendix). The safety benefit comes from locating *each failure's* boundary, not from how much data is kept.

D. Safe Prefixes Help Most When Clean Data Is Scarce

When clean demonstrations are scarce, discarding failures is most costly, and recovering their safe prefixes matters most. At 20 training pairs CoP-prefix reaches 0.442 versus 0.298 for BC-success ($+0.144$, p=0.038). At 50 pairs the gap widens to $+0.204$, p=0.009 (Table III, Fig. 2). In practice 50-pair CoP-prefix (0.824) essentially matches 80-pair BC-success (0.822):

TABLE III
DATA EFFICIENCY. MEAN CLOSED-LOOP SUCCESS [95% CI].

Train pairs	BC-all mean [95% CI]	BC-success mean [95% CI]	CoP-prefix mean [95% CI]	CoP - BC-all	CoP - BC-success
20	0.242 [0.178, 0.313]	0.298 [0.222, 0.376]	0.442 [0.382, 0.504]	+0.200, p=0.002	+0.144, p=0.038
50	0.384 [0.304, 0.460]	0.620 [0.562, 0.682]	0.824 [0.744, 0.900]	+0.440, p=1.77e-04	+0.204, p=0.009
80	0.396 [0.329, 0.456]	0.822 [0.787, 0.855]	0.871 [0.815, 0.922]	+0.475, p=2.64e-06	+0.049, p=0.200

TABLE IV
ROBUSTNESS AND MECHANISM DIAGNOSTICS.

Diagnostic	Runs	Result
Progress alignment	6	CoP-progress - BC-all +0.483, p=9.01e-04; vs time CoP +0.023, p=0.514
Chunk length	6	H=1/H=8/H=16 means 0.897/0.870/0.853; H1-H16 +0.043, p=0.003
Late +16 stress test	6	CoP +16 - base CoP -0.307, p=0.013, wins/losses 0/6
Reward-scope AUC	3	Local - trajectory AUC +0.016, p=0.008, wins/losses 3/0

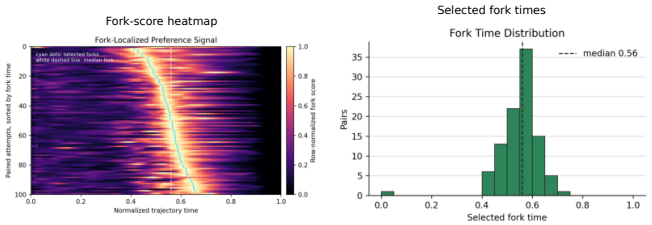

Fig. 3. Fork diagnostics over all 100 paired Can attempts. Fork scores and selected fork times concentrate in the middle of the trajectory rather than at trivial start or end timestamps.

salvaging safe pre-fork behavior substitutes for roughly $1.6\times$ as many clean successful demonstrations.

E. Robustness and Mechanism

The curation benefit is robust. Progress alignment via banded DTW preserves it (CoP-progress beats BC-all by $+0.483$, $p{=}9{\times}10^{-4}$, and ties time-aligned CoP-prefix). Chunk lengths $H{\in}\{1, 8, 16\}$ give success $0.897/0.870/0.853$. Selected forks concentrate around mid-trajectory divergence points (mean normalized time 0.551, with median selected score $4.47\times$ the median trajectory score) rather than collapsing to trivial start or end positions (Fig. 3), so the boundary the method removes is a genuine behavioral transition, not an artifact.

VI. FROM DESIGN-TIME CURATION TO RUNTIME MONITORING

Fork mining is a design-time intervention: it removes unsafe behavior before the policy ever sees it. The same signal also points toward the runtime view of safety. The fork is detected as the point where a failed attempt's state and action diverge from a matched safe attempt, and our similarity diagnostic shows matched success–failure distances are small pre-fork and grow sharply across the fork (Appendix). At deployment, that comparison can be inverted: given a library of safe trajectories and an in-progress rollout, a rising divergence score is an early, reward-free indicator that the current execution is approaching a known failure fork, a candidate trigger for caution, human handoff, or a safe fallback policy. A complementary label-quality diagnostic supports this direction: fork-window preference labels achieve held-out AUC 1.000 versus 0.984 for whole-trajectory labels ($\Delta{=}+0.016$, $p{=}0.008$; Appendix), indicating that the fork-localized signal is cleaner than a trajectory-level one. We present this as a bridge, not a runtime result: the matched counterfactual that powers curation is unavailable at test time, and the library-based divergence substitute has not been evaluated closed-loop, so we make no runtime safety claim. The contribution here is the design-time curation primitive, and the monitor is an open direction it enables.

VII. LIMITATIONS AND OPEN PROBLEMS

All policy-return evidence is on RoboMimic Can Paired low-dimensional data. This should be read as a focused, controlled study, not a broad benchmark. We did not find exact paired success/failure entries for other RoboMimic tasks in the queried public listing (Appendix), so scaling requires either new paired datasets or approximate-pair construction. CoP-Miner assumes matched or progress-alignable attempts and does not address arbitrary unpaired mixed-quality data.

a) Multi-strategy tasks.: The pairing further assumes the matched success is a true counterfactual for the failure. In tasks that admit several distinct valid strategies, a strategy that appears only among failures has no success executing the same early behavior: the nearest success diverges immediately, the mined fork collapses toward the start, and the failure is over-truncated. Natural guards are strategy-aware pairing (clustering attempts by early behavior before matching) and treating pairs with low pre-fork similarity as unpaired data to be discarded or down-weighted rather than cut.

b) Toward embodied foundation models.: The fork score uses Euclidean distances in low-dimensional state and action space. Whether the peak structure survives in image space or under language conditioning is open. The lift is concrete because the score needs only distances on observations and action chunks: d_o and d_+ can be computed in a pretrained visual-encoder latent space (e.g., the vision backbone of a VLA policy), and d_a over the same action chunks that chunked VLA heads already predict. Approximate pairs can be constructed from large heterogeneous logs by matching language instructions and initial-observation embeddings. Because the output is only a curated dataset, the intervention drops into standard VLA fine-tuning pipelines unchanged. Closing the loop on the Sec. VI monitor as a deployment-time shield remains open.

VIII. CONCLUSION

Failure-containing logs are a safety liability for imitation: a policy trained on full failures can reproduce the unsafe behavior that caused them. But the hazard is localized. When a log contains matched success/failure counterfactuals, the behavioral fork marks where safe behavior ends and unsafe behavior begins, and a simple, reward-free, constraint-free mining step can excise the unsafe suffix at the source. On RoboMimic Can Paired this converts failures into useful, safe imitation data, with the largest gains where clean data is scarce, and points toward a matching runtime monitor built from the same signal.

Acknowledgments

We thank the anonymous reviewers and the workshop organizers for feedback that improved the discussion of multi-strategy pairing, the foundation-model extension, and the scope of the runtime-monitoring claim.

References

[1] Christopher Agia, Rohan Sinha, Jingyun Yang, Rika Antonova, Marco Pavone, Haruki Nishimura, Masha Itkina, and Jeannette Bohg. Cupid: Curating data your robot loves with influence functions. In *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 2907–2932. PMLR, 2025. URL https://proceedings.mlr.press/v305/agia25a.html.

[2] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Data quality in imitation learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2306.02437.

[3] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 783–792. PMLR, 2019. URL https://proceedings.mlr.press/v97/brown19a.html.

[4] Yuxin Chen, Chen Tang, Jianglan Wei, Chenran Li, Ran Tian, Xiang Zhang, Wei Zhan, Peter Stone, and Masayoshi Tomizuka. Mereq: Max-ent residual-q inverse rl for sample-efficient alignment from intervention. *arXiv preprint arXiv:2406.16258*, 2024. URL https://arxiv.org/abs/2406.16258.

[5] Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.nips.cc/paper/7017-deep-reinforcement-learning-from-human-preferences.

[6] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021. URL https://arxiv.org/abs/2106.06860.

[7] Daniel H. Grollman and Aude G. Billard. Robot learning from failed demonstrations. *International Journal of Social Robotics*, 4(4):331–342, 2012. URL https://doi.org/10.1007/s12369-012-0161-z.

[8] Brendan Hertel and Seyed Reza Ahmadzadeh. Learning from successful and failed demonstrations via optimization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7807–7812, 2021. URL https://doi.org/10.1109/IROS51168.2021.9636679.

[9] Geon-Hyeong Kim, Seokin Seo, Jongmin Lee, Wonseok Jeon, HyeongJoo Hwang, Hongseok Yang, and Kee-Eung Kim. Demodice: Offline imitation learning with supplementary imperfect demonstrations. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=BrPdX1bDZkQ.

[10] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

[11] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 1678–1690. PMLR, 2022. URL https://proceedings.mlr.press/v164/mandlekar22a.html.

[12] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019. URL https://arxiv.org/abs/1910.00177.

[13] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635. PMLR, 2011. URL https://proceedings.mlr.press/v15/ross11a.html.

[14] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978. URL https://doi.org/10.1109/TASSP.1978.1163055.

[15] Kyriacos Shiarlis, Joao Messias, and Shimon Whiteson. Inverse reinforcement learning from failure. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems*, pages 1060–1068, 2016. URL https://www.cs.ox.ac.uk/publications/publication10272-abstract.html.

[16] Kun Wu, Ning Liu, Zhen Zhao, Di Qiu, Jinming Li, Zhengping Che, Zhiyuan Xu, and Jian Tang. Learning from imperfect demonstrations with self-supervision for robotic manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 16899–16906, 2025. URL https://doi.org/10.1109/ICRA55743.2025.11127918.

[17] Yueh-Hua Wu, Nontawat Charoenphakdee, Han Bao, Voot Tangkaratt, and Masashi Sugiyama. Imitation learning from imperfect demonstration. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6818–6827. PMLR, 2019. URL https://proceedings.mlr.press/v97/wu19a.html.

[18] Haoran Xu, Xianyuan Zhan, Honglei Yin, and Huiling Qin. Discriminator-weighted offline imitation learning

from suboptimal demonstrations. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 24725–24742. PMLR, 2022. URL https://proceedings.mlr.press/v162/xu22l.html.

APPENDIX

This appendix provides implementation, experimental, and reproducibility details for CoP-Miner. The main paper should be read as a controlled study of paired success/failure robot demonstrations. The strongest empirical claim is that failed demonstrations are harmful when imitated wholesale, but become useful imitation data when restricted to mined pre-fork prefixes. The policy-return evidence is concentrated on RoboMimic Can Paired low-dimensional data [11]: the dataset has matched success/failure attempts from the same task initializations, which directly matches the counterfactual premise of the method. We do not claim broad generality to arbitrary unpaired failed demonstrations.

A. Input Assumptions

The method assumes a set of paired attempts. Each pair contains one successful trajectory and one failed trajectory from the same or a near-matched initial state. The implementation preserves pair integrity when splitting data. The primary split uses 80 training pairs and held-out pairs for validation and diagnostics. Additional split seeds reshuffle at the pair level. For RoboMimic Can Paired, we build a 100-pair manifest by matching each successful demonstration to the nearest unused failed demonstration in initial state / initial-observation space. All 100 pairs have near-zero initial distance, and the maximum recorded distance in the manifest is 2.5×10^{-16}. Pair-level splitting is used throughout, so matched success and failure demonstrations from the same initial condition are never split across train/validation/test partitions.

B. Fork Mining

For each pair, the successful and failed trajectories are aligned by normalized time or by progress. Let $(o^s_{1:T_s}, a^s_{1:T_s})$ and $(o^f_{1:T_f}, a^f_{1:T_f})$ denote the low-dimensional observation-action sequences. Time alignment uses $j(i) = \mathrm{round}(i(T_f - 1)/(T_s - 1))$. Progress alignment uses a banded DTW map over observations with radius 25 and then monotonizes the resulting map. Given aligned timesteps $(i, j(i))$, the implementation computes current-state, action-chunk, and future-state divergences:

$$
\begin{aligned}
d_o(i) &= \|o^s_i - o^f_{j(i)}\|_2, \\
d_a(i) &= \|a^s_{i:i+H} - a^f_{j(i):j(i)+H}\|_2, \\
d_+(i) &= \|o^s_{i+\Delta} - o^f_{j(i)+\Delta}\|_2.
\end{aligned}
$$

The default hyperparameters are action chunk length $H = 8$, future offset $\Delta = 12$, smoothing width 5, and minimum retained prefix length 12. With $\sigma_o = \mathrm{median}(d_o)$ and m_+, mad_+ the median and median absolute deviation of d_+, the raw fork score is

$$
\begin{aligned}
r(i) = d_a(i) &\exp(-d_o(i)/\sigma_o) \\
&\times \left(1 + \exp(-(d_+(i) - m_+)/\mathrm{mad}_+)\right)^{-1}.
\end{aligned}
$$

Scores are smoothed by a width-5 moving average. Candidate forks must be in the top score quintile, have current-state distance no larger than the 60th percentile, and have future-state distance at least the 70th percentile. The selected fork is the earliest candidate. If the candidate set is empty, the score argmax is used. The selected aligned index gives (τ_s, τ_f), and each failed demonstration is retained as a prefix of length $\max(12, \tau_f)$.

C. Algorithmic Summary

1) Build a pair-level manifest of matched success/failure attempts and split pair IDs into train/validation/test partitions.
2) For each pair, flatten the configured low-dimensional observations and align success timesteps to failure timesteps by normalized time or progress DTW.
3) Compute $d_o(i)$, $d_a(i)$, $d_+(i)$ and the smoothed fork score $r(i)$ for each valid aligned timestep.
4) Select the earliest high-score candidate satisfying the state and future-divergence gates, falling back to the score argmax when needed.
5) Construct the dataset from every successful trajectory plus each failed trajectory truncated to $\max(12, \tau_f)$ transitions.
6) Train the native RoboMimic BC-RNN/GMM policy on the resulting HDF5 filter key and evaluate closed-loop success over 50 rollouts.

D. Baselines and Controls

- BC-all uses all successful trajectories and all failed trajectories, testing whether naive full-failure imitation is harmful.
- BC-success uses successful trajectories only, testing whether prefix retention adds value beyond discarding failures.
- Midpoint-prefix uses successful trajectories plus failed prefixes cut at the failure-trajectory midpoint, controlling for simple truncation.
- Shuffled-prefix preserves the CoP-prefix prefix-length distribution but shuffles prefix lengths across failed trajectories, controlling for retained failed-data volume while breaking pair-specific fork localization.

E. Policy Backbone

The primary policy-return experiments use native RoboMimic BC-RNN/GMM policies. The policy is trained for 300 epochs with a recurrent low-dimensional policy and a 5-mode GMM action head. Each trained policy is evaluated for 50 closed-loop rollouts with horizon 400. The main result aggregates 11 matched split/train runs.

F. Statistical Protocol

The independent unit for policy-return statistics is a matched split/train run. Each run contributes one closed-loop success rate from 50 rollouts. Rollouts are not treated as independent samples for significance testing. Paired comparisons use two-sided paired t-tests on seed-level success rates. Wins, losses, and ties are strict comparisons of the paired seed-level deltas. Reported confidence intervals are 95% intervals over seed-level means. We treat the diagnostic p-values as exploratory and do not use them as corrected claims of family-wise significance.

Table V reports the primary policy success rates. CoP-prefix reaches 0.8709 success, compared with 0.3964 for BC-all. The corresponding paired comparison in Table VI gives a $+0.4745$ delta, $p = 2.64\times10^{-6}$, and wins 11/11. The full-data comparison between CoP-prefix and BC-success is positive on mean but not statistically decisive (0.8709 versus 0.8218, $p = 0.200$). We therefore frame CoP-prefix as strongly correcting the harm from full failed trajectories and as competitive with success-only BC at full data.

TABLE V
MAIN NATIVE POLICY SUCCESS.

Method	Runs	Mean success	Std	Min	Max
BC-all	11	0.396	0.112	0.200	0.520
BC-success	11	0.822	0.060	0.720	0.920
Midpoint-prefix	11	0.780	0.072	0.700	0.920
Shuffled-prefix	11	0.782	0.094	0.520	0.860
CoP-prefix	11	0.871	0.096	0.700	0.980

TABLE VI
MATCHED PAIRED COMPARISONS.

Comparison	Runs	Method mean	Baseline mean	Delta	p-value	Wins/Losses/Ties
CoP-prefix - BC-all	11	0.871	0.396	+0.475	2.64e-06	11/0/0
CoP-prefix - BC-success	11	0.871	0.822	+0.049	0.200	7/4/0
CoP-prefix - Midpoint	11	0.871	0.780	+0.091	0.020	8/2/1
CoP-prefix - Shuffled	11	0.871	0.782	+0.089	0.037	9/2/0
BC-success - BC-all	11	0.822	0.396	+0.425	3.27e-08	11/0/0

Table VII reports two additional controls. First, BC-success-equal-transitions duplicates successful demonstrations so that success-only BC has the same transition budget as CoP-prefix. Second, Global-median-prefix keeps failed prefixes using one split-level median fork length instead of a per-pair mined fork. CoP-prefix outperforms both controls over the same 11 matched runs, addressing the concern that the result is explained only by additional transitions or by any fixed failed-prefix rule. A third stress test shifts mined fork lengths across pairs before truncating failures. This mismatched-fork-prefix control reaches 0.8255 mean success versus 0.8709 for CoP-prefix ($\Delta = +0.0455$, $p = 0.181$), which we treat as a scope diagnostic rather than proof of robustness to arbitrary unpaired data.

TABLE VII
ADDITIONAL DATA-VOLUME AND FIXED-PREFIX CONTROLS.

Comparison	Runs	CoP mean	Baseline mean	Delta	p	W/L/T
CoP-prefix - BC-success-equal-transitions	11	0.871	0.789	+0.082	0.00748	8/2/1
CoP-prefix - Global-median-prefix	11	0.871	0.825	+0.045	0.0449	7/3/1

Table VIII shows that failed prefixes are most useful when successful demonstrations are scarce. At 20 training pairs, CoP-prefix reaches 0.4422 compared with 0.2978 for BC-success and 0.2422 for BC-all. At 50 training pairs, CoP-prefix reaches 0.8244 compared with 0.6200 for BC-success and 0.3844 for BC-all. Table IX shows the paired low-data comparisons. Both comparisons against success-only BC are statistically decisive over 9 matched split/train runs.

TABLE VIII
DATA-EFFICIENCY SUMMARY.

Train pairs	BC-all mean [95% CI]	BC-success mean [95% CI]	CoP-prefix mean [95% CI]	CoP - BC-all	CoP - BC-success
20	0.242 [0.178, 0.313]	0.298 [0.222, 0.376]	0.442 [0.382, 0.504]	+0.200, p=0.002	+0.144, p=0.038
50	0.384 [0.304, 0.460]	0.620 [0.562, 0.682]	0.824 [0.744, 0.900]	+0.440, p=1.77e-04	+0.204, p=0.009
80	0.396 [0.329, 0.456]	0.822 [0.787, 0.855]	0.871 [0.815, 0.922]	+0.475, p=2.64e-06	+0.049, p=0.200

TABLE IX
EXPANDED LOW-DATA PAIRED COMPARISONS.

Train pairs	Comparison	Runs	Method mean	Baseline mean	Delta	p	W/L/T
20	CoP - BC-all	9	0.442	0.242	+0.200	0.00188	8/1/0
20	CoP - BC-success	9	0.442	0.298	+0.144	0.0385	7/2/0
50	CoP - BC-all	9	0.824	0.384	+0.440	0.000177	9/0/0
50	CoP - BC-success	9	0.824	0.620	+0.204	0.00868	8/1/0

Table X summarizes the main robustness and mechanism diagnostics. Progress alignment preserves the benefit over BC-all. Chunk lengths $H = 1$, $H = 8$, and $H = 16$ all remain high. Moving the boundary late by 16 steps substantially degrades performance, supporting the mechanism that post-fork failure suffixes are harmful.

TABLE X
ROBUSTNESS AND MECHANISM DIAGNOSTICS.

Diagnostic	Runs	Result
Progress alignment	6	CoP-progress - BC-all +0.483, p=9.01e-04; vs time CoP +0.023, p=0.514
Chunk length	6	H=1/H=8/H=16 means 0.897/0.870/0.853; H1-H16 +0.043, p=0.003
Late +16 stress test	6	CoP +16 - base CoP -0.307, p=0.013, wins/losses 0/6
Reward-scope AUC	3	Local - trajectory AUC +0.016, p=0.008, wins/losses 3/0

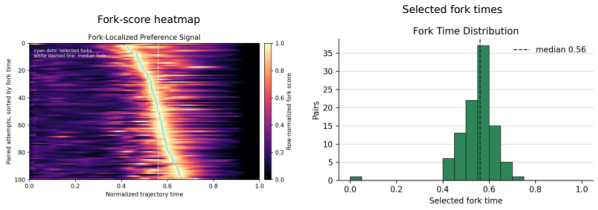

Fig. 4. Fork diagnostics. Selected forks concentrate around meaningful mid-trajectory divergence points rather than at trivial start or end positions.

a) Retained failed-prefix distribution.: At 80 training pairs, CoP-prefix keeps 0.546 of failed transitions on average over split seeds 0/1/2, compared with 0.497 for midpoint-prefix and 1.000 for BC-all. Thus CoP-prefix uses about 0.797 of the full BC-all transition budget while removing the post-fork failed suffixes that harm BC.

b) Fork-rule sensitivity.: Table XI varies the boundary-selection rule. Changing the future offset from 12 to 6 or 24 barely changes the median selected boundary, and argmax-candidate selection recovers the same selected boundaries as the default earliest-candidate rule on this dataset. Removing smoothing, loosening gates, or dropping score components moves boundaries more, so we report these as real sensitivity checks rather than claiming invariance.

Table XII reports the completed downstream policy-return sensitivity runs available at package build time. The completed three-run variants remain near the default CoP-prefix policy-return level, suggesting that the method is not brittle to reasonable future-offset or smoothing choices.

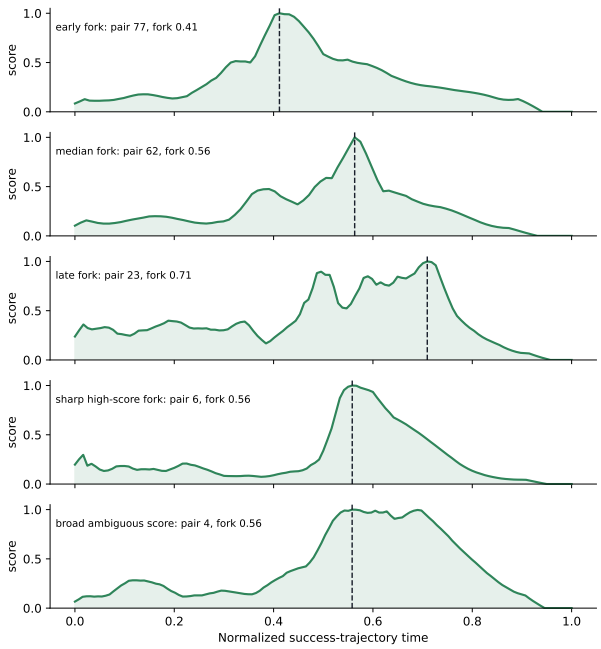

Fig. 5. Representative fork-score curves. The dashed vertical line marks the selected fork for an early, median, late, sharp, and broad-score pair. Across all 100 pairs, the selected fork is at the maximum score percentile in the smoothed curve, with median selected-score / median-score ratio 4.47.

TABLE XI
BOUNDARY-LEVEL FORK SENSITIVITY BEFORE DOWNSTREAM POLICY EVALUATION. DELTAS ARE ABSOLUTE SHIFTS IN THE SELECTED FAILURE INDEX RELATIVE TO THE DEFAULT FORK RULE.

Variant	Pairs	Median fork time	Median —shift—	Mean —shift—
argmax_gate	100	0.561	0.0	0.0
full_no_gates	100	0.463	7.0	10.5
future24	100	0.554	0.0	2.1
future6	100	0.556	0.0	2.5
loose_gates	100	0.545	4.0	5.1
smooth1	100	0.550	1.0	4.3
state_future	100	0.399	10.5	14.6

c) Fork-score component ablations.: Table XIII ablates the terms in the fork score. On Can, action-window divergence is the dominant signal. Future-only scoring is weak, while adding future consequence to action divergence remains close to the default rule. These ablations are diagnostic rather than a claim that every heuristic term is necessary on every dataset.

Table XIV reports supplementary checks targeting two questions: whether a standard offline-RL method can absorb the full mixed success/failure dataset, and whether the full fork score is necessary relative to a simpler action-divergence-only rule. For the offline-RL baselines, IQL and TD3-BC train on the same full success-plus-failure data used by BC-all. IQL improves over naive BC-all in some settings but remains far below CoP-prefix, and TD3-BC fails on the completed 50- and 80-pair settings. These results are supplementary baseline evidence, not a broad offline-RL benchmark: they use the native RoboMimic implementations with the same 300-epoch, 50-rollout budget and no algorithm-specific hyperparameter sweep. The final two rows compare the default full fork score against an action-only fork score at 50 training pairs over six matched seeds. The action-only variant is respectable but lower (0.703 versus 0.790 mean success), supporting the use of state proximity and future divergence in the rule.

TABLE XII
FORK-RULE POLICY-RETURN SENSITIVITY. RUN COUNTS SHOW HOW MANY MATCHED TRAINING SEEDS HAD FINISHED BY THE PACKAGE BUILD.

Variant	Runs	Mean success	Min	Max
argmax candidate	3	0.847	0.720	0.940
future offset=24	3	0.860	0.820	0.900
future offset=6	3	0.867	0.760	0.940
looser gates	3	0.920	0.860	0.960
smooth=1	3	0.900	0.880	0.920

TABLE XIII
FORK-SCORE COMPONENT ABLATIONS ON SPLIT SEED 0.

Variant	Runs	Mean success	Min	Max
Action only	3	0.927	0.900	0.980
Full score, no gates	3	0.913	0.880	0.940
Action + future	3	0.887	0.860	0.940
Full CoP score	11	0.871	0.700	0.980
Current-state gate + future	3	0.833	0.800	0.900
Action + current-state gate	3	0.780	0.720	0.840
Future state only	3	0.460	0.420	0.480

TABLE XIV
SUPPLEMENTARY RESCUE CHECKS. OFFLINE-RL BASELINES TRAIN ON FULL MIXED SUCCESS/FAILURE TRAJECTORIES, WHILE FORK-SCORE ROWS COMPARE THE DEFAULT FULL SCORE WITH AN ACTION-ONLY SCORING RULE AT 50 PAIRS.

Method	Train pairs	Runs	Mean success	Min	Max
IQL BC-all	20	3	0.113	0.08	0.16
IQL BC-all	50	3	0.267	0.20	0.36
IQL BC-all	80	3	0.307	0.30	0.32
TD3-BC BC-all	50	3	0.000	0.00	0.00
TD3-BC BC-all	80	3	0.000	0.00	0.00
CoP full fork score	50	6	0.790	0.68	0.86
CoP action-only fork score	50	6	0.703	0.60	0.84

d) Approximate-pair task feasibility.: Table XV records two attempts to move beyond exact Can pairs without changing the core setting. Lift-MG sparse contains both successes and failures, so we constructed nearest-initial-state approximate pairs. The three-seed pilot is low-success and mixed, which we treat as a stress diagnostic rather than a generalization claim. Square-MH and the queried Transport-MH low-dimensional files contain only successful demonstrations in the public file queried by our tooling, so they cannot instantiate the success/failure-pair setting. These results clarify the scope of the current evidence and why the main closed-loop claims remain on Can Paired.

The reward-scope comparison is diagnostic rather than a native policy-return ablation. Local fork-window preferences achieve held-out AUC 1.0000, while whole-trajectory preferences achieve AUC 0.9838. The paired AUC delta is $+0.0162$ with $p = 0.0082$. This supports the claim that fork-window

TABLE XV

APPROXIMATE-PAIR TASK FEASIBILITY AND PILOT RESULTS. BLANK POLICY ENTRIES INDICATE THAT NO SUCCESS/FAILURE PAIRS COULD BE BUILT, SO GPU TRAINING WAS NOT LAUNCHED.

Task	Status	Success/failure	Pairs	Median init dist	BC-all	BC-success	CoP-prefix
lift_mg_sparse_approx	prepared	316/1184	316	0.0968	0.093	0.113	0.100
square_mh_approx	insufficient_success_failure_pairs	300/0	0				
transport_mh_approx	insufficient_success_failure_pairs	300/0	0				

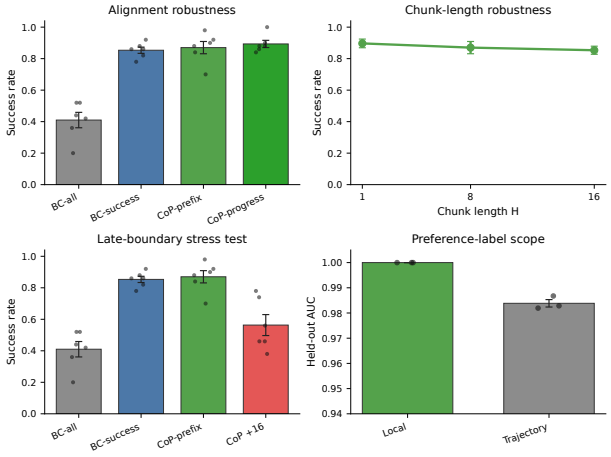

Fig. 6. Robustness diagnostics for alignment, chunk length, late-boundary stress, and reward-scope AUC.

labels are cleaner in the paired setting, but it should not be interpreted as proof that a native trajectory-preference policy baseline is worse in closed-loop return.

The reproducibility package includes source code, configs, Modal entry points, paper source dependencies, final figures and tables, command manifests, run manifests, and cost estimates. The represented paper tables contain 187 unique native RoboMimic run IDs and 6 reward-scope training runs. The conservative runtime estimate is 85.63 compute hours and $53.87, which is an estimate rather than a billing export.