RoboPocket: Improve Robot Policies Instantly with Your Phone

Junjie Fang[13*] Wendi Chen[12*] Han Xue[13*†] Fangyuan Zhou[12*]
Tian Le[13] Yi Wang[12] Yuting Zhang[3] Jun Lv[3] Chuan Wen[1‡] Cewu Lu[123‡]
[1]Shanghai Jiao Tong University [2]Shanghai Innovation Institute [3]Noematrix Ltd.
[*]Equal contribution [†]Project lead [‡]Corresponding authors

robo-pocket.github.io

Fig. 1: **RoboPocket** enables instant robot-free policy updates with smartphones by visualizing policy intent via AR Visual Foresight and collecting targeted corrections in minutes.

***Abstract*—Scaling robot learning is bottlenecked not only by data volume, but also by whether demonstrations target the policy's failures. Existing robot-free handheld interfaces enable portable in-the-wild collection, yet remain open-loop; interactive methods such as DAgger collect policy-aware corrections, but require deploying imperfect policies on physical robots. We present RoboPocket, a smartphone-based system for *robot-free policy iteration*. RoboPocket visualizes the policy's predicted trajectory in augmented reality as *AR Visual Foresight*, allowing users to identify weaknesses and collect targeted corrective data without robot execution. An asynchronous finetuning pipeline incorporates new corrections and updates the policy within minutes. Across diverse manipulation tasks, RoboPocket achieves an average 2× sample-efficiency gain over blindly scaling offline demonstrations.**

I. Introduction

While scaling data has transformed vision and language [9, 6], robot learning faces a data bottleneck [13]. Recent robot-free handheld interfaces [3, 17, 18] scale collection but remain open-loop: users record demonstrations without knowing if they address the policy's weaknesses. Conversely, interactive methods like DAgger [16] target policy-induced failures but require deploying imperfect policies on physical robots, making iteration costly and unsafe. Consequently, robot learning faces a trade-off: robot-free collection is scalable but policy-agnostic, while interactive correction is policy-aware but strictly robot-dependent.

We introduce **RoboPocket**, a portable system that resolves this trade-off by turning consumer smartphones into closed-loop interfaces for robot-free policy iteration. By leveraging the smartphone's integrated display, on-device compute, and low-latency communication, RoboPocket transforms demonstration collection from passive recording into computationally guided data generation. The core interface is **AR Visual Foresight**: the phone streams observations to an inference server and overlays the policy's predicted trajectory onto the real scene via augmented reality [11]. Users can therefore see the policy's intended behavior, identify failure modes before deploying a robot, and collect targeted corrective demonstrations in the corresponding states.

This extended abstract summarizes RoboPocket as a practical case study for demonstration quality. Our main lesson is that quality extends beyond individual trajectories to whether the collection interface exposes the right failures at the right time.

Related work. Teleoperation systems such as ALOHA [20] and GELLO [19] provide precise demonstrations but remain robot-bound. Robot-free handheld systems such as UMI [3] improve portability but record open-loop demonstrations. DAgger [16] collects policy-induced corrections but requires robot execution. RoboPocket combines handheld portability with policy-aware correction by visualizing predicted trajectories in AR.

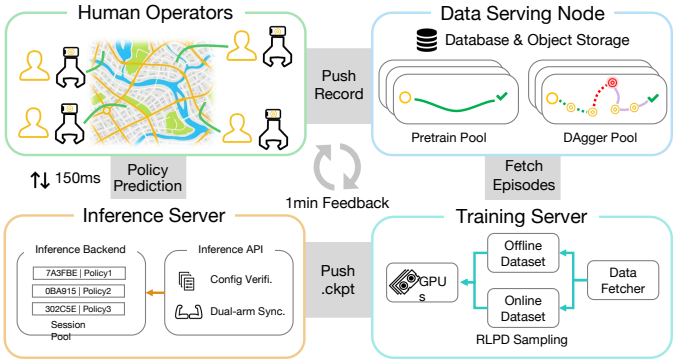


Fig. 2: **Robot-Free Instant Policy Iteration.** Users identify policy weaknesses via AR Visual Foresight, upload corrections immediately, and receive updated predictions from the inference server in real time (< 150ms).

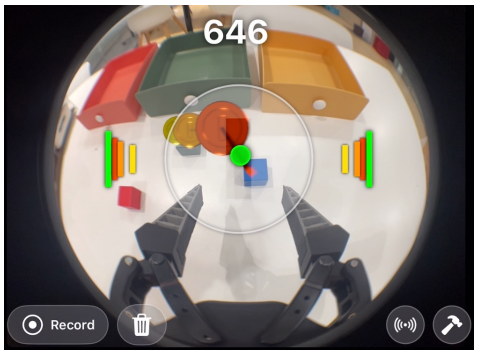


Fig. 3: **Screenshot of AR Visual Foresight.** Predicted actions are rendered as coins on the phone screen.

II. RoboPocket System

RoboPocket turns a consumer smartphone into a closed-loop interface for robot-free data collection and policy interaction. The system consists of a smartphone-based handheld collector, an isomorphic adaptive gripper, augmented sensing modules, and an iOS interface. The hardware is aligned with the Robotiq 2F-85 gripper, using a low-cost 3D-printed adaptive mechanism, fisheye visual context expansion, and an external gripper-width encoder. The phone runs visual-inertial tracking, kinematic checks, AR rendering, and networking on device. In tracking evaluation, RoboPocket achieves 2.8mm/0.4° average error for a single device and 4.0mm/0.7° under dual-device synchronization.

During collection, the software monitors tracking reliability and kinematic feasibility, alerting users before unusable data is stored. In a low-texture setting, active verification reduces abnormal trajectories from 30% to 0%, where abnormality means more than 1m localization drift or more than 5cm trajectory jumps.

III. Robot-Free Instant Policy Iteration

Interactive imitation learning improves robustness by collecting corrective data from states induced by the current policy [16]. RoboPocket approximates this without robot execution: users query the current policy in the real world, observe its predicted behavior through AR, and collect corrections where the policy is likely to fail. A cloud server returns real-time action predictions for AR visualization, letting users see intended motion and immediately provide targeted demonstrations without waiting for robot failures. Observations are captured exclusively while the visualized coins are followed by the user, and corrections are appended to the dataset through explicit recording; the loop is thus rendered insensitive to operator hesitation or deviation.

To turn feedback into policy improvement, each corrective trajectory is uploaded immediately. The training server incorporates new corrections while retaining the original dataset, and updated checkpoints are synchronized to the inference server. The next AR Visual Foresight query therefore reflects the improved policy rather than a fixed pretrained model. The end-to-end update completes within 90s.

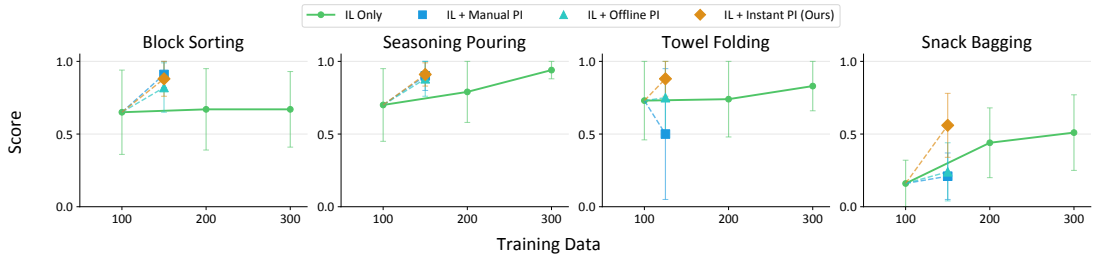
IV. Experiments

We evaluate RoboPocket on a data-scaling validation task, four manipulation tasks (**Block Sorting**, **Seasoning Pouring**, **Towel Folding**, and bimanual **Snack Bagging**), and a distributed multi-user study. Robot evaluations use Flexiv Rizon 4 arms with Robotiq 2F-85 grippers. Policies are trained with Diffusion Policy [4].

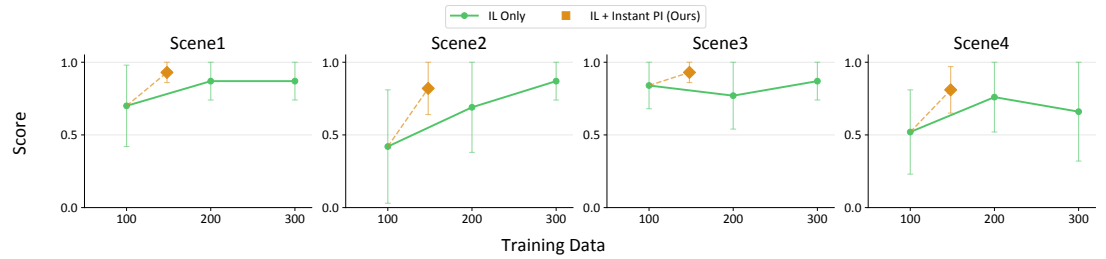
Data fidelity and scaling. We collect 1,600 Mouse Arrangement demonstrations across 64 environment-object pairs. OOD performance follows clear data-scaling trends with environment-object diversity [7]. This validates RoboPocket as a scalable data engine.

Robot-Free Instant Policy Iteration (PI) improves scaling efficiency. Across four manipulation tasks, pure imitation learning (IL) improves with more offline data, but its gains are limited by covariate shift and task-specific failure modes: wrong execution order in Block Sorting, poor recovery after large rotations in Seasoning Pouring, incorrect semantic grasp points in Towel Folding, and grasping or occlusion failures in bimanual Snack Bagging. By exposing these failures through AR Visual Foresight, RoboPocket guides users to collect corrections near policy-relevant states. **IL + Instant PI** reaches or exceeds the performance of substantially larger offline datasets, achieving up to $2\times$ higher data efficiency than pure offline scaling.

Online feedback supports distributed and non-expert correction. Four users collect Block Sorting data in four distinct rooms. After training a base policy from 100 demonstrations (25 per scene), each user performs Robot-Free Instant Policy Iteration in their own environment and collects only 12 corrective demonstrations. With these 12 interactions per scene, success improves substantially in difficult environments (Scene 2: $0.42 \rightarrow 0.82$, Scene 4: $0.52 \rightarrow 0.81$). In a separate user study, PCA and KNN analyses show that non-expert users capture state distributions close to experienced experimenters, including critical states with high feature variation. This democratization of collection suggests a viable path toward crowdsourcing policy-aware demonstrations at scale, without requiring robot hardware in user homes.



(a) Instant PI improves data efficiency across four tasks.



(b) Distributed users improve policies with 12 corrections per scene.

Fig. 4: **Empirical summary.** AR-guided instant PI improves sample efficiency and scales across scenes.

V. Practical Lessons

Anecdote: manual correction can hurt when correction quality is poor. In Towel Folding, expert manual policy iteration (PI) (inspecting failure videos to collect corrections) can unexpectedly reduce performance ($0.73 \rightarrow 0.50$), whereas RoboPocket's **IL + Instant PI** yields stable improvement (0.88). A correction is not automatically useful just because it addresses a failure, especially if the operator cannot see the policy's intent.

Hypothesis: showing policy intent improves correction quality. AR Visual Foresight gives users a concrete signal about what the policy intends to do, allowing them to collect corrections near wrong decision points. This changes the operator's role from reactive safeguarding to proactive data collection. In our user study, all users report real-time feedback is beneficial, 7/10 rate Visual Foresight as "Very Helpful" for discovering failures, and 8/10 believe instant policy iteration helps realize model improvement.

Practical rule I: validate demonstrations during collection, not only after. Post-hoc curation is insufficient when tracking failure, kinematic infeasibility, and policy ambiguity can be detected online. For 10 Seasoning Pouring demonstrations, a UMI-style pipeline required 8m34s collection, 1m24s transfer, and 9m12s offline SLAM; RoboPocket required 3m51s acquisition and 1m37s transfer without offline SLAM.

Practical rule II: collect fewer, more policy-aware corrections. Offline scaling improves performance, but additional demonstrations are inefficient when they miss policy failures. In Block Sorting, proactive intervention achieves scores comparable to following the full AR trajectory (5.25 vs. 5.30/6.0) but halves the time per effective correction (75.7s to 39.1s).

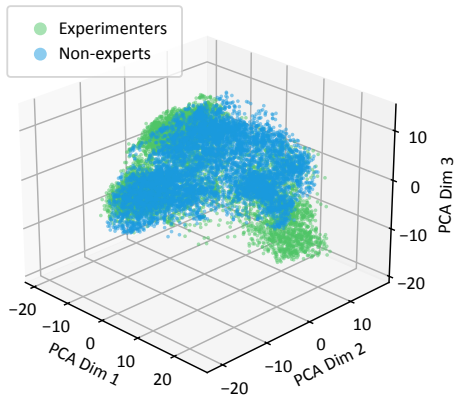


Fig. 5: **State coverage.** Non-expert data covers states comparable to expert data.

VI. Conclusion

RoboPocket brings policy feedback into robot-free demonstration collection. Through AR Visual Foresight and asynchronous finetuning, users identify weaknesses, collect targeted corrections, and improve policies without physical robot deployment. Our practical lessons are to make policy intent visible, validate data online, and prefer policy-aware corrections over blindly scaling offline demonstrations.

Limitations. AR Visual Foresight currently reveals errors in predicted kinematic trajectories; it cannot anticipate failures in contact-rich interactions (*e.g.*, inserting a key), where success depends on forces invisible to vision. Future work could expose such contact-level intent through richer channels, such as rendering predicted contact forces haptically in VR.

References

[1] Apple. Arkit. https://developer.apple.com/documentation/arkit, 2026. 5

[2] Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pages 1577–1594. PMLR, 2023. 6

[3] Cheng Chi, Zhenjia Xu, Chuer Pan, Eric Cousineau, Benjamin Burchfiel, Siyuan Feng, Russ Tedrake, and Shuran Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024. 1, 9

[4] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025. 2, 9

[5] Flexiv Rizon 4 robotic arm. https://www.flexiv.com/products/rizon, 2024. 9

[6] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022. 1

[7] Yingdong Hu, Fanqi Lin, Pingyue Sheng, Chuan Wen, Jiacheng You, and Yang Gao. Data scaling laws in imitation learning for robotic manipulation. *arXiv preprint arXiv:2410.18647*, 2024. 2, 7, 9

[8] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025. 10

[9] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 1

[10] Yoshihiko Nakamura and Hideo Hanafusa. Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control. *Journal of Dynamic Systems, Measurement, and Control*, 108(3):163–171, September 1986. ISSN 0022-0434. doi: 10.1115/1.3143764. 5

[11] Rhys Newbury, Akansel Cosgun, Tysha Crowley-Davis, Wesley P Chan, Tom Drummond, and Elizabeth A Croft. Visualizing robot intent for object handovers with augmented reality. In *2022 31st IEEE international conference on robot and human interactive communication (RO-MAN)*, pages 1264–1270. IEEE, 2022. 1

[12] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 7, 10

[13] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024. 1

[14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021. 10

[15] Robotiq Inc. Robotiq 2f-85 adaptive gripper. https://robotiq.com/products/adaptive-grippers, 2026. Accessed: 2026-01-22. 9

[16] Stéphane 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*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011. 1, 2, 5

[17] Generalist AI Team. Gen-0: Embodied foundation models that scale with physical interaction. *Generalist AI Blog*, 2025. https://generalistai.com/blog/nov-04-2025-GEN-0. 1

[18] RDT Team. Rdt2: Enabling zero-shot cross-embodiment generalization by scaling up umi data, September 2025. URL https://github.com/thu-ml/RDT2. 1

[19] Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12156–12163. IEEE, 2024. 1

[20] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023. 1

Appendix

This section provides additional implementation details of the RoboPocket hardware and software system described in Sec. II.

A. Isomorphic Adaptive Gripper Design

RoboPocket is designed to match the geometry and passive behavior of the Robotiq 2F-85 gripper used in our robot setup. Unlike generic handheld parallel grippers, our collector follows the visual geometry and grasping configuration of the target robot gripper, reducing the embodiment gap between robot-free demonstrations and robot execution.

To better reproduce contact-rich interactions, the handheld gripper approximates the adaptive finger behavior of the Robotiq 2F-85. We integrate pre-compressed torsion springs into the distal finger joints to emulate the passive degree of

freedom of the real gripper. This allows the handheld device to exhibit passive finger deformation during compliant grasping or unintended surface contact, so the collected trajectories better reflect the physical behavior of the deployed gripper.

The gripper also uses a leverage-based linkage mechanism to amplify human finger input. This makes it easier for users to apply and maintain sufficient grasping force during long data collection sessions. The main structure is 3D printed using standard FDM processes, and the full handheld assembly has a bill of materials cost of approximately $70, excluding the smartphone.

B. Augmented Sensing Modules

Although modern smartphones provide high-quality cameras, IMUs, and visual-inertial tracking, they do not directly provide all sensing channels needed for robot policy learning. RoboPocket therefore augments the phone with two additional sensing components.

First, we attach an off-the-shelf fisheye lens through a custom 3D-printed mount. This expands the effective field of view of the wrist-mounted camera, allowing the observation to capture both the gripper-object interaction and a larger portion of the surrounding workspace. This is important for manipulation tasks where object context, task progress, and gripper pose must be inferred from the same wrist-view observation.

Second, RoboPocket measures gripper width using an external magnetic encoder. The encoder signal is transmitted to an ESP32-based interface board through a 1 Mbps RS485 bus, achieving an angular resolution of 0.088° at 30Hz. The ESP32 streams the gripper-width state to the iPhone via BLE GATT. The firmware follows a lightweight publisher-subscriber pattern, which keeps the gripper state synchronized with visual and pose observations while leaving the interface extensible to additional sensors.

C. Real-Time Data Verification

RoboPocket performs online verification to identify unusable or risky data during collection rather than after offline post-processing. The verification module monitors both tracking reliability and kinematic feasibility.

For tracking reliability, the system monitors visual-inertial odometry quality using indicators such as feature density and sudden pose or velocity jumps. Frames with unstable tracking are marked as invalid, and the interface immediately notifies the user through visual feedback. This allows users to adjust the scene, motion speed, or viewpoint before severe drift corrupts the trajectory.

For kinematic feasibility, RoboPocket runs an on-device inverse-kinematics solver to map the handheld gripper motion to the robot joint space. We use a damped least-squares Jacobian solver [10] and check for infeasible configurations such as joint-limit violations or near-singular motions. When the recorded pose is difficult or impossible for the robot to execute, the interface provides real-time warnings so that the user can correct the motion during collection.

D. AR Trajectory Replay

In addition to algorithmic checks, RoboPocket provides AR trajectory replay for in-situ trajectory inspection. After a demonstration, the recorded end-effector trajectory is rendered back into the real scene through the phone display. This allows the user to verify whether the reconstructed trajectory aligns with the physical motion and whether the demonstration completes the intended task. In practice, this replay function helps reveal SLAM drift, discontinuous pose jumps, unstable grasps, and task-level failures immediately after collection.

E. Multi-Device Spatiotemporal Synchronization

For bimanual manipulation, RoboPocket synchronizes multiple smartphone collectors in both space and time. Spatial alignment is achieved through a peer-to-peer map sharing protocol based on ARKit [1]. Devices exchange world maps and establish a shared coordinate frame, allowing trajectories from different phones to be represented in the same world frame.

Temporal alignment is handled by a lightweight network synchronization protocol that aligns device clocks with approximately 5ms precision. Each sensor packet, including images, poses, and gripper-width measurements, is timestamped on device and later aligned across devices. This ensures that demonstrations collected from multiple handheld interfaces can be fused into synchronized multi-arm trajectories for policy learning.

This section provides additional formulation and implementation details for the Robot-Free Instant Policy Iteration workflow described in Sec. III.

F. Connection to Interactive Imitation Learning

We formulate a manipulation task as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. Standard imitation learning trains a policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ from a static demonstration dataset $\mathcal{D}_{demo} = \{(\mathbf{s}_t, \mathbf{a}_t)\}$, but the learned policy may encounter out-of-distribution states due to compounding errors. Let d_π denote the state distribution induced by policy π. The target objective is to minimize the imitation loss under the policy-induced distribution:

$$J(\pi) = \mathbb{E}_{\mathbf{s}\sim d_\pi}[\ell(\pi(\mathbf{s}), \pi^*(\mathbf{s}))], \tag{1}$$

where π^* denotes the expert policy.

Interactive imitation learning methods such as DAgger [16] address this issue by aggregating corrective data from the state distributions induced by intermediate policies. Let d_{π_i} be the state distribution induced by policy π_i at iteration i. After N iterations, the aggregated distribution can be written as

$$\bar{d}_N(\mathbf{s}) = \frac{1}{N}\sum_{i=1}^{N} d_{\pi_i}(\mathbf{s}), \quad \hat{\pi}_{N+1} = \arg\min_{\pi\in\Pi} \mathbb{E}_{\mathbf{s}\sim\bar{d}_N}[\ell(\mathbf{s}, \pi)]. \tag{2}$$

Under the standard no-regret reduction, one obtains a best policy $\hat{\pi} \in \hat{\pi}_{1:N}$ satisfying

$$J(\hat{\pi}) \le J(\pi^*) + T\epsilon_N + O(1), \tag{3}$$

where ϵ_N is the average surrogate loss and T is the task horizon. This formulation motivates RoboPocket's goal: collecting corrective data in states that are relevant to the current policy, while avoiding physical robot execution.

G. Remote Inference Implementation

RoboPocket uses an edge–cloud architecture for policy interaction. The smartphone acts as a lightweight client responsible for observation capture, pose tracking, user interaction, and AR visualization. Policy inference is offloaded to a GPU-equipped inference server. When a client starts a session, the server loads the corresponding policy checkpoint and configuration, maintains the policy state across queries, and returns predicted action trajectories to the phone.

During interaction, the smartphone streams the current observation to the inference server. The server predicts an action trajectory and sends it back to the client for AR visualization. By maintaining persistent model states and avoiding repeated model initialization, the system achieves real-time interaction with a round-trip latency below 150ms over standard Wi-Fi in our setup.

H. AR Visual Foresight Rendering

RoboPocket visualizes the predicted policy trajectory in the real scene through augmented reality. Since our camera uses a fisheye lens to expand the manipulation field of view, directly rendering virtual objects with a standard pinhole camera model would cause visible misalignment. We therefore apply distortion-aware rendering based on the calibrated camera intrinsics. In practice, the predicted trajectory is rendered as a sequence of virtual markers in the fisheye camera view, and a real-time vertex displacement procedure aligns the virtual trajectory with the distorted image.

The AR interface is designed to make policy intent immediately interpretable. Users can follow the visualized trajectory when it appears correct, or provide a corrective demonstration when the prediction is wrong, uncertain, or likely to fail. When the device reaches the end of the current prediction horizon, the system captures a new observation and queries the policy again, allowing users to continuously interact with the current policy during data collection.

I. Proactive Intervention

In addition to automatic policy queries, RoboPocket provides a physical intervention button that allows users to request a new policy prediction at any time. This enables users to actively probe the current policy rather than passively following a fixed trajectory. For example, when the predicted trajectory approaches an ambiguous object configuration or an out-of-distribution state, the user can trigger a new query, inspect the updated prediction, and decide whether to collect a corrective demonstration.

Instead of waiting for a robot to fail, users can explore the policy's weak regions in a robot-free setting and collect demonstrations that directly target those states.

J. Asynchronous Online Finetuning

The backend contains three services: an *Inference Server*, a *Data Serving Node*, and a *Training Server*. The Inference Server handles real-time policy queries from smartphones. The Data Serving Node receives newly collected trajectories and manages the growing correction dataset. The Training Server monitors the Data Serving Node and continuously finetunes the policy when new corrective data arrives.

To balance adaptation and retention, we train on a mixture of the original offline dataset $\mathcal{D}_{demo}$ and the newly collected correction dataset $\mathcal{D}_{on}$. Following the intuition of replay-based policy updates such as RLPD [2], each finetuning batch samples 50% of data from $\mathcal{D}_{demo}$ and 50% from $\mathcal{D}_{on}$. This allows the policy to quickly fit recent corrections while reducing catastrophic forgetting of the original demonstration distribution.

Updated checkpoints are periodically synchronized from the Training Server to the Inference Server. In our implementation, model weights are pushed every N training steps. The next policy query from the smartphone therefore uses the latest available checkpoint. This closes the loop between AR-guided correction, online finetuning, and updated policy feedback.

To validate whether RoboPocket enables users without task-specific training (*i.e.*, non-expert users) to identify policy weaknesses and collect correction data in a robot-free setting, we design a user study.

K. Participants and Procedure

We invite a group of volunteers with varying levels of experience in imitation learning and UMI-based data collection. The demographic distribution of the participants is illustrated in Fig. 6. The participants are invited to perform a Block Sorting task using *Robot-Free Instant Policy Iteration*.

The experimental procedure was structured as follows:

1) **10-Minute Warm-up:** Each participant undergoes a brief warm-up session to familiarize themselves with the device operation.
2) **Offline PI:** Participants collect 5 trajectories using the Offline (Policy Iteration) PI workflow.
3) **Instant PI:** Participants collect 5 trajectories using our proposed Robot-Free Instant Policy Iteration workflow.
4) **Feedback:** Upon completion, participants fill out a questionnaire. The corrective data collected is also stored for subsequent visualization analysis.

To eliminate the bias, the order in which each user performs Offline PI and Instant PI is randomized.

L. Qualitative Results

The results of our user survey are summarized in Fig. 7. Regarding the system's core features, all users report that the real-time constraints and feedback mechanisms are beneficial for data collection, with over half of the participants rating them as "Very Helpful". There is a consensus among all users that our Virtual Foresight feature effectively assists in discovering model failure cases, with 7 out of 10 participants rating it as "Very Helpful". Additionally, 8 out of 10 users

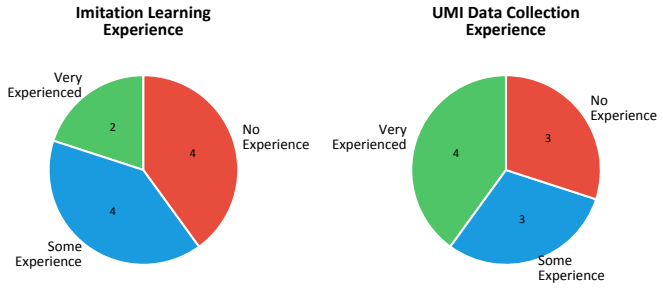


Fig. 6: **Participant Demographics.** The distribution of volunteers based on their prior experience with imitation learning and UMI data collection.

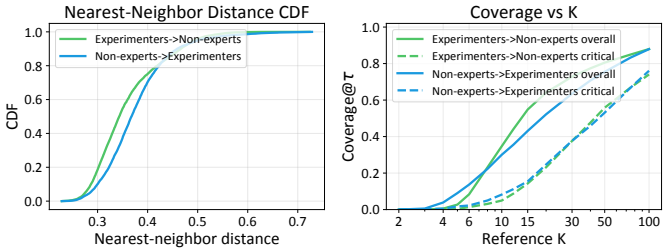


Fig. 8: **Quantitative State Coverage Analysis.** Cross-dataset KNN coverage between non-expert and expert data in the PCA feature space.

believe that the Instant Policy Iteration is highly beneficial for realizing the model's improvement. When asked about the most significant advantage of RoboPocket, the majority of users provide feedback that the "Real-time Uploading" and "Online Finetuning" mechanisms make the collection process significantly more effective.

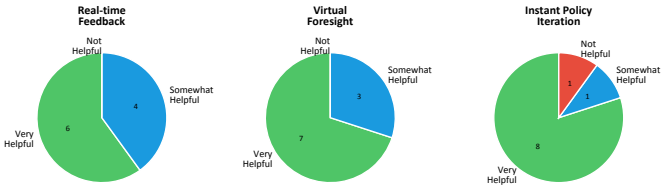


Fig. 7: **User Study Results.** Visualization of user feedback regarding the helpfulness of Real-time Feedback, AR Visual Foresight, and the Instant Policy Iteration workflow.

M. Visualization Analysis

To assess the quality of the data collected by non-experts, we perform a visualization analysis. We utilize PCA to project the DINOv2 [12] features of the state data and compare non-expert data with the data collected by experimenters. Here, we use the Offline PI data for comparison since each participant's online iterations are limited. The PCA visualization is shown in the main text. Here, we further quantify state coverage in the 3D PCA feature space using cross-dataset K-nearest-neighbor (KNN) analysis.

As shown in Fig. 8, most states from one dataset fall within the neighborhood of the other dataset when $K{=}1$, and the coverage curves converge rapidly at larger K. This pattern also holds for critical states, defined as the top 10% states with the highest feature variation. This shows that non-expert users can capture state distributions close to those collected by experienced experimenters.

In Hu et al. [7], the authors conduct a comprehensive empirical study to investigate how the generalization performance of robot policies scales with the number of training environments, objects, and demonstrations. They discover that policy performance follows a power-law relationship with data diversity, concluding that increasing the diversity of environments and objects is significantly more critical for zero-shot generalization than merely increasing the number of demonstrations per scene.

Following their experimental protocol, we collect data for the "Mouse Arrangement" task to verify that our RoboPocket system is capable of generating high-quality data that adheres to these established scaling laws. We collect training data across 32 environments and 47 object pairs, which are shown in Fig. 9 and Fig. 10, respectively. We randomly take 2 object pairs for data collection in 1 environment and collect 25 demonstrations for 1 env-object pair. To broaden the coverage of the dataset, the environments include both indoor and outdoor settings with diverse lighting conditions and textures. The mice and mouse pads are combined to generate object pairs. The evaluation is done in 3 different scenes with 2 initial robot poses and 3 initial object poses.

For all experiments, we use normalized scores to evaluate policy performance. The specific scoring criteria for each task are detailed below:

a) Mouse Arrangement: We adopt similar scoring criteria to Data Scaling Laws. The robot is required to pick up a mouse and then place it on a mouse pad. The maximum score is 6 points, with 3 points for picking and 3 points for placing.

Picking:

- **0 points:** The gripper fails to approach the mouse.
- **1 point:** The gripper contacts the mouse, but drops it immediately during the lift.
- **2 points:** The gripper displaces the mouse significantly before grasping, or successfully grasps the mouse but drops it at a higher height.
- **3 points:** The gripper successfully grasps the mouse.

Placing:

- **0 points:** The gripper hovers without approaching the mouse pad, or releases the mouse prematurely from a height.
- **1 point:** The mouse lands outside the mouse pad or flips over due to an excessive release height.
- **2 points:** The mouse rests only partially on the pad, or exhibits bouncing or shifting caused by a high release point.

Fig. 9: **Training environments for verifying Data Scaling Laws.** These environments encompass a wide range of lighting conditions and textures.

- **3 points:** The gripper descends and places the entire mouse securely and stably on the pad.

b) Block Sorting: The robot is required to sort the Red (R), Green (G), and Blue (B) blocks into their corresponding boxes in a sequential order. The maximum score is 9 points, with 3 points allocated for each block. The detailed breakdown is:

- **0 points:** The gripper moves to an area without blocks and closes.
- **1 point:** The gripper attempts to approach the correct block but fails to grasp it successfully.
- **2 points:** The gripper successfully grasps the correct block but places it into the wrong box.
- **3 points:** The gripper grasps the correct block and places it into the correct box.

If the model completes the sorting in an incorrect order, the score is calculated based on the Longest Increasing Subsequence (LIS) matching the ground truth. For example, completing the task in R-B-G order yields 6 points, while B-G-R yields 3 points.

c) Seasoning Pouring: The task involves picking up seasoning jars in a left-to-right order, pouring them over a plate, and placing them back on the table. The maximum score is 12 points, with 4 points for each jar.

- **0 points:** The gripper moves to an area without jars and closes.
- **1 point:** The gripper attempts to approach the correct jar but fails to grasp it.
- **2 points:** The gripper grasps the correct jar but fails to reach the pouring position above the plate or drops the jar.
- **3 points:** The gripper successfully pours the seasoning but fails to place the jar stably back on the table.
- **4 points:** The gripper successfully pours and places the jar stably on the table.

Similar to Block Sorting, if the execution order is incorrect, the final score is calculated based on the LIS match.

d) Towel Folding: The robot must grasp the towel's corners and place them in the correct positions to complete two folds. The maximum score is 6 points, with 3 points per fold.

- **0 points:** The gripper moves to a wrong target location.
- **1 point:** The gripper moves to the correct corner but fails to grasp the towel.
- **2 points:** The gripper grasps the correct corner but fails to place it in the correct position.
- **3 points:** The gripper grasps the correct corner and places it in the correct position.

e) Snack Bagging (Bimanual): The left arm holds the bag while the right arm sequentially places four snacks into it. The maximum score is 12 points, with 3 points for each snack.

- **0 points:** The gripper moves to a wrong target location.
- **1 point:** The gripper moves to the correct snack but fails to grasp it.
- **2 points:** The gripper grasps the snack but fails to put it into the bag.

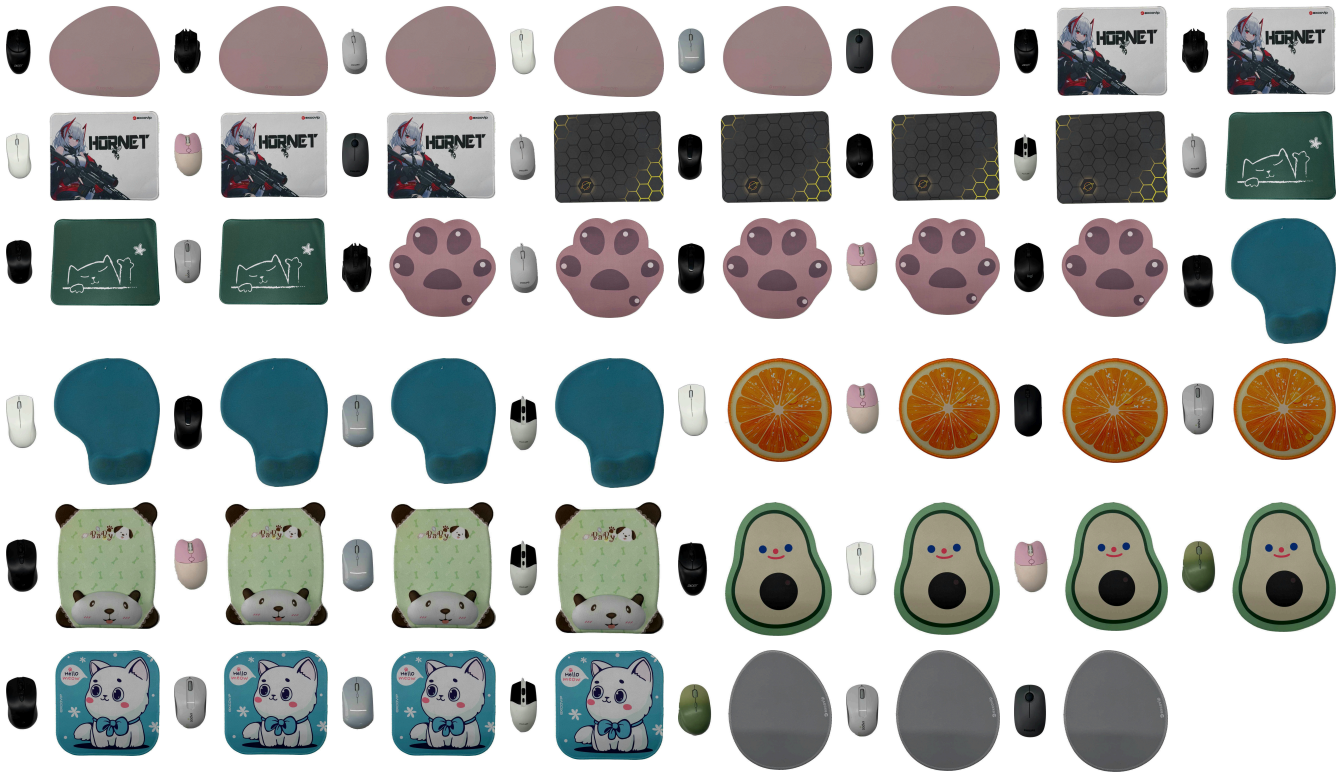


Fig. 10: **Training object pairs for verifying the Data Scaling Laws.** We combine mice and mouse pads to obtain a diverse set of object pairs.

- **3 points:** The gripper grasps the snack and successfully puts it into the bag.

Following Data Scaling Laws [7], we construct a custom movable lifting table. As shown in Fig. 11, the setup features a Flexiv Rizon 4 robot arm [5] equipped with a Robotiq 2F-85 adaptive gripper [15], connected via a 90-degree adapter. To ensure physical isomorphism with our handheld device and maintain adaptive grasp performance, the gripper is fitted with the same TPU soft fingers used in the RoboPocket collector. Notably, in the Data Scaling Laws experiments, we use the DH-ROBOTICS AG-160-95 gripper and a corresponding isomorphic handheld collector. Additionally, an iPhone mount is attached to the gripper, where the RoboPocket APP streams camera feeds to a workstation in real-time. The workstation, powered by an Intel Core i9-12900K CPU and an NVIDIA GeForce RTX 3090 GPU, is housed on the table's lower shelf. This workstation also acts as the Data Serving Node and the Training Server. The entire system is powered by an EcoFlow DELTA 3 MAX portable power station. For bimanual tasks, two such platforms can be arranged side-by-side to support dual-arm manipulation.

During *Robot-free Instant Policy Iteration*, we use another workstation as an Inference Server, which is equipped with an Intel Core i9-13900K CPU and an NVIDIA GeForce RTX 4090 GPU.

We use the CNN-based Diffusion Policy based on Diffusion

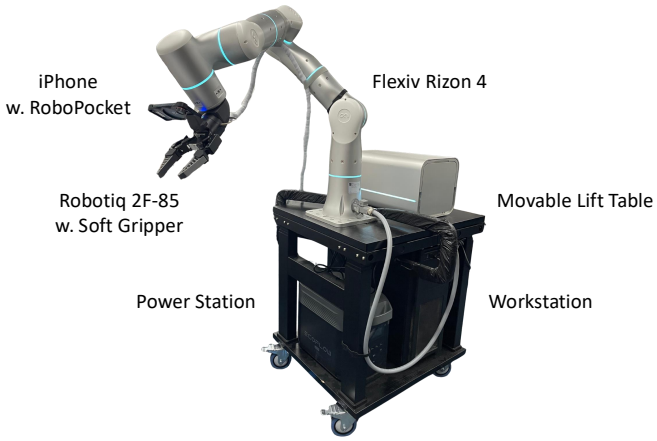


Fig. 11: **Robot Evaluation Setup.** The setup includes a movable lifting table, a Flexiv Rizon 4 robot with a Robotiq 2F-85 gripper, and a workstation powered by a mobile power station.

Policy [4] and UMI [3] code base for policy training. We set the observation horizon to $T_{\text{obs}} = 1$ so that observations collected during *Robot-free Instant Policy Iteration* exclude any explicit or implicit velocity information. This design prevents instability in human motion speed from being encoded

into the observations. Although we discard historical context, large-scale results [8] have confirmed that it does not impose an excessive impact on the scalability. All data is originally collected at 30Hz but is slowed down by a factor of 3 to 10Hz to align with the robot's physical joint velocity constraints. The specific hyperparameters used for training are listed in Tab. I.

TABLE I: Hyperparameters for Policy Training

Hyperparameter	**Value**
Observation Horizon (T_{obs})	1
Action Prediction Horizon (T_{pred})	16
Action Execution Horizon (T_{exec})	8
Training Epochs	600
Batch Size	64
Optimizer	AdamW
Optimizer Momentum	$\beta_1 = 0.95, \beta_2 = 0.999$
U-Net Learning Rate	3.0×10^{-4}
Observation Encoder	DINOv2 (Towel Folding) [12] CLIP (Others) [14]
Observation Encoder Learning Rate	3.0×10^{-5}
Learning Rate Schedule	Cosine Decay
Training Denoising Steps	50
Inference Denoising Steps	16

During the *Robot-Free Instant Policy Iteration*, we adjust several training parameters, which are summarized in Tab. II.

TABLE II: Hyperparameters for Robot-free Instant Policy Iteration

Hyperparameter	**Value**
Batch Size	32
Learning Rate	1.0×10^{-4}
Observation Encoder Learning Rate	1.0×10^{-5}
Learning Rate Schedule	Constant
Model Sync Interval N	100 Steps