Robot Fine-Tuning Made Easy: Pre-Training Rewards and Policies for Autonomous Real-World Reinforcement Learning

Jingyun Yang[*] **Max Sobol Mark**[*] **Brandon Vu** **Archit Sharma**
Jeannette Bohg **Chelsea Finn**
Stanford University

Abstract

The pre-train and fine-tune approach in machine learning has been highly successful across various domains, enabling rapid task learning by utilizing existing data and pre-trained models from the internet. We seek to apply this approach to robotic reinforcement learning, allowing robots to learn new tasks with minimal human involvement by leveraging online resources. We introduce ROBOFUME, a reset-free fine-tuning system that pre-trains a versatile manipulation policy from diverse prior experience datasets and autonomously learns a target task with minimal human input. In real-world robot manipulation tasks, our method can incorporate data from an external robot dataset and improve performance on a target task in as little as 3 hours of autonomous real-world experience. We also evaluate our method against various baselines in simulation experiments. Website: `robofume.github.io`

1 Introduction

In many domains that involve machine learning, a widely successful paradigm for learning task-specific models is to first pre-train a general-purpose model from an existing diverse prior dataset, and then adapt the model with a small addition of task-specific data [1–5]. This paradigm is attractive to real-world robot learning, since collecting data on a robot is expensive, and fine-tuning an existing model on a small task-specific dataset could substantially improve the data efficiency for learning a new task. Pre-training a policy with offline reinforcement learning and then fine-tuning it with online reinforcement learning is a natural way to implement this paradigm in robotics. However, numerous challenges arise when using this recipe in practice. First, off-the-shelf robot datasets often use different objects, fixture placements, camera viewpoints, and lighting conditions compared to the local robot platform. This causes non-trivial distribution shifts between pre-training and online fine-tuning data, which makes effectively fine-tuning a robot policy difficult. Second, training or fine-tuning a policy in the real world often requires extensive human supervision, which includes manually resetting the environment between trials [6–8] and engineering reward functions [9–11]. In this work, our goal is to address these two challenges and develop a practical framework that enables robot fine-tuning with minimal time and human effort.

Over the past few years, there has been a lot of progress in designing efficient and autonomous reinforcement learning algorithms. Some works propose to reduce the need for manual environment resets using reset-free RL [9–20], where an agent alternates between running a task policy and a reset policy during training while updating both with online experience. However, these works do not leverage diverse off-the-shelf robot datasets. Recent advances in offline RL algorithms have enabled policies to leverage diverse offline data and improve further via online fine-tuning [21–28], but these new methods have not been integrated into a system that aims at minimizing human supervision in

[*]Equal Contribution.

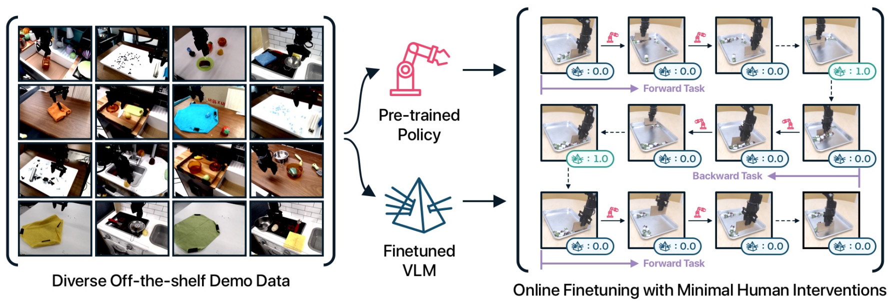
Diverse Off-the-shelf Demo Data
Pre-trained Policy
Finetuned VLM
: 0.0
: 1.0
Forward Task
Backward Task
Online Finetuning with Minimal Human Interventions

Figure 1: We propose a system that enables autonomous and efficient real-world robot learning. First, we pre-train a **multi-task policy** and fine-tune a pre-trained **Vision-Language Model (VLM) as a reward model** using diverse off-the-shelf demonstration datasets. Then, we fine-tune the **pre-trained policy** online reset-free with the **VLM reward model**.

the fine-tuning phase. There are also works that propose to eliminate the need for human-specified reward functions by learning reward prediction models [18, 19, 29, 30], but we found that many of these proposed models can be brittle when deployed in a real-world RL fine-tuning setup. In summary, although prior works have presented individual components that are vital to building a working system for efficient and human-free robot learning, it is not clear which components one should use to put together such a system and how.

We design ROBOFUME, a system that enables autonomous and efficient real-world robot learning by leveraging diverse offline datasets and online fine-tuning. Our system operates in two phases. In the pre-training phase, we assume access to a diverse prior dataset, a few task demonstrations and reset demonstrations of the target task, and a small collection of sample failure observations in the target task. From this data, we learn a language-conditioned, multi-task policy with offline RL. To cope with the distribution shift between the offline dataset and online interactions, we utilize calibrated offline RL techniques [28] that correct the scale of the learned Q-values by underestimating the predicted values of the learned policy from offline data. To ensure the online fine-tuning phase requires minimal human feedback, we propose to utilize large vision-language models (VLMs) to provide a robust pre-trained representation and fine-tune it with a small amount of in-domain data so that it is tailored for the reward classification setup. In the fine-tuning phase, a robot adapts the policy in the real world autonomously by alternating between attempting the task and attempting to reset the environment to the initial state distribution of the task. Meanwhile, the agent uses the pre-trained VLM model as a surrogate reward for updating the policy.

We evaluate our system by pre-training it on the Bridge dataset [31] and testing it on a diverse set of real-world downstream tasks: candy sweeping, cloth folding, cube covering, pot lid, and pot pick-and-place. We find that our system provides substantial improvements over offline-only methods with as little as 3 hours of real-world training. We perform more quantitative experiments in a simulation setup, where we illustrate that our method outperforms imitation learning and offline RL methods that either do not perform fine-tuning online or do not incorporate diverse prior data.

2 ROBOFUME

Our work focuses on designing an efficient and scalable framework for pre-training on a diverse set of prior demonstrations and autonomously fine-tuning on target tasks. Our system consists of an offline pre-training phase and an online fine-tuning phase. In Section 2, we discuss how we pre-train a language-conditioned multi-task policy on diverse data that can be fine-tuned online efficiently. Online fine-tuning requires a reward function to label successes and failures. In Section 2, we introduce a VLM-based classifier for providing a reward signal to the policy in the fine-tuning phase. Finally, in Section 2, we describe how to autonomously adapt the pre-trained policy in the fine-tuning phase by utilizing the VLM-based reward classifier as a reward signal and chaining forward and backward behaviors to practice the task with minimal human interventions.

Preliminaries Our method assumes access to a prior dataset $\mathcal{D}_{\text{prior}} = \cup_{j=1}^{N} \mathcal{D}_j = \cup_{j=1}^{N} \{(s_i^j, a_i^j, s_i'^j)\}_{i=1}^{K}$, which consists of demonstrations of N different tasks $\tau_1, \dots \tau_k$. We assume that all demonstration data uses image observations. The method will be tested on a downstream task τ_f, which is different from any of the prior tasks. To facilitate learning on the downstream task, we also assume the availability of a small set of target task demos $\mathcal{D}_f$, target task reset demos $\mathcal{D}_b$, and target task failure states $\mathcal{D}_{☹}$. The reset demos $\mathcal{D}_b$ come from the reset task τ_b which resets the environment from an end state of τ_f to the initial state distribution of τ_f. The failure states $\mathcal{D}_{☹}$ consist entirely of image observations that correspond to unsuccessful states and are collected to aid with the VLM reward learning. In addition to all the given data ($\mathcal{D}_{\text{prior}}$, $\mathcal{D}_f$, $\mathcal{D}_b$, $\mathcal{D}_{☹}$), each task τ is also accompanied with a language description l.

Pre-Training a Multi-Task Policy on Diverse Prior Data Prior work has shown that training a policy using a conservative Q-value function is an effective way to obtain a good policy from an offline dataset [32, 33]. However, fine-tuning can be critical to learn competent policies as prior data may not provide sufficient coverage, especially for new tasks or scenes. We leverage CalQL [28] which modifies the conservative Q-learning algorithm CQL such that it enables efficient online fine-tuning by enforcing calibration on the Q-function (i.e. making the Q-value of the learned policy no lower than the Monte-Carlo returns in the prior dataset). CalQL allows us to improve the pre-trained policy efficiently with respect to online interactions.

CalQL requires the training of an actor and a critic. Since we use image observations, we additionally train an encoder $\phi(s_{\text{img}})$ that projects the images into a lower-dimensional space before giving them as inputs to the actor and critic. The encoder ϕ is a 4-layer CNN, and is optimized exclusively against the critic loss. To best utilize the multi-task data, we encode task descriptions l using pre-trained CLIP embeddings, resulting in an embedding $z = \text{CLIP}(l)$ which is used as the task representation. The policy then takes as inputs a concatenation of the encoded image observation $\phi(s_{\text{img}})$, task representation z, and proprioceptive information s_p, processes the concatenated vector through an MLP, and produces the output action a. We train the policy π and the critic Q with datasets $\mathcal{D}_{\text{prior}}, \mathcal{D}_f, \mathcal{D}_b$. After the offline learning phase, the policy and critic contain knowledge of all tasks in the prior data and the target task.

Fine-Tuning A Vision-Language Model for Rewards To improve the autonomy of the policy fine-tuning phase, our agent needs to perform online fine-tuning without manually labeled or engineered reward functions. To achieve this, we propose to fine-tune off-the-shelf vision-language models as reward predictors. We design a VLM-based reward model that takes the current observation and the task name as input and outputs a binary label of whether the current observation corresponds to a successful state or an unsuccessful state with respect to the task. Given a task name (eg. 'put green cabbage into sink'), we first use GPT4 to convert the name to a short question that could serve as a prompt to know if the task has been completed or not (eg. 'is green cabbage placed in the sink?'). Then, we pass the converted prompt to a VLM together with the current image of the environment. The VLM outputs a sparse binary reward, returning success if the 'yes' token has a higher probability than 'no' token.

We use MiniGPT4 [34] as the VLM for receiving (image, task prompt) pairs and answering whether the task has been successfully completed. We find the zero-shot performance of the pre-trained VLM to be unsatisfactory. To improve the VLM's performance for reward modeling, we fine-tune it using the prior and target task data. In particular, for every demonstration, the last 3 states are used as success states and the ground truth answer is labeled as 'Yes'; for all other states, we label the ground-truth answer as 'No'. To provide the model with more information about failed states, we collect a small dataset $\mathcal{D}_{☹}$ of images that correspond to unsuccessful states for forward and backward target tasks. We find that fine-tuning leads to a more accurate reward model.

Autonomous Online Fine-tuning The offline pre-training phase produces a single language-conditioned policy $\pi(\cdot|s, l)$ that can perform the target and reset tasks when provided their respective language instructions l_f and l_b. The policy is then deployed in a hardware setup for online fine-tuning.

Since we aim for a fully autonomous setup, we roll out the policy in a reset-free manner, alternating between attempting the target task τ_f with $\pi(\cdot|s, l_f)$ and the reset task τ_b with $\pi(\cdot|s, l_b)$. We use the fine-tuned VLM from the previous subsection as the sparse reward function for the RL algorithm. When the VLM predicts the task has been completed successfully, we terminate the episode and switch the language instruction for the policy to complete the other task. In addition to switching tasks upon completion as predicted by the VLM, we switch after a fixed number of timesteps (150) to ensure the robot does not become stuck in bad states. As mentioned in Section 2, we fine-tune the policy using CalQL with an additional BC regularization term on the critic. We find that without a BC regularization term, behaviors degrade over the course of training. By constraining the policy to stay close to the expert demonstrations from the target and reset tasks, the agent becomes less likely

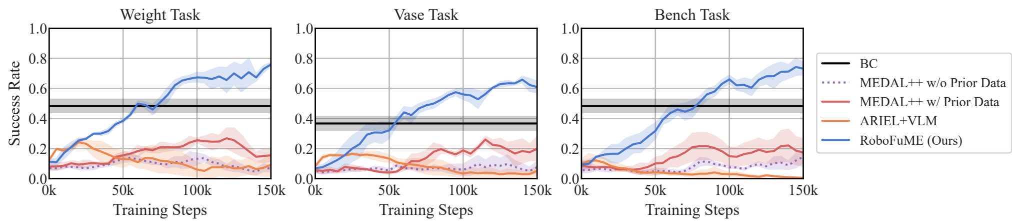

Figure 2: Performance of our method on three simulated environments. We report the success rate over the course of training, averaged over three seeds. Our method RoboFuME outperforms BC, ARIEL+VLM [11], and MEDAL++ [19] consistently on all three domains.

to exploit false-positives from the VLM reward model. We use the same fixed BC regularization weight throughout fine-tuning as we did on the offline pre-training phase. Our fine-tuning pipeline is implemented on top of the implementation of MEDAL++ [19]. Please refer to this work for more details on our training procedure.

3 Results & Discussion

Real Robot Experiments We evaluate RoboFuME on five different real-robot manipulation tasks. We use a WidowX 250 robotic arm with a single third-person camera (Logitech C920, resizing images to 100 × 100 pixels). Our method runs autonomously executing back and forth the target task and the reset target task for a fixed number of steps or until the VLM predicts success. More real robot experimental details are in the Appendix Section 4.2.

Task	BC	RoboFuME (offline)	RoboFuME (FT 30k steps)
Cloth Covering	45%	60%	**80%**
Cloth Folding	60%	70%	**85%**
Candy Sweeping	31%	47%	**66%**
Pot Lid	60%	40%	**95%**
Pot PNP	45%	35%	**55%**

Table 1: Real-robot results on 5 manipulation tasks. Our method significantly improves over both offline-only and BC performance after 30k steps of online interaction (2-4 hours).

Table 1 shows the results of our method after pretraining (labeled offline) and after autonomous fine-tuning (labeled FT 30k steps), comparing with a behavior cloning (BC) baseline. BC trains a language-conditioned policy on all the prior and target data. After 30k steps of autonomous online interaction, our method shows relative improvement of 51% upon the pre-trained performance, and outperforms BC by 58% on an average. For pick and place tasks (*pot lid* and *pot pnp*), the fine-tuned policy was more likely to retry if it initially failed to grasp the object. For candy sweeping, BC and the pre-trained policy were prone to overshooting and pushing on the border of the tray after the first sweep, while fine-tuning the policy achieved higher success by chaining multiple sweeping attempts.

Simulation Experiments We use a suite of simulated robotic manipulation environments to ablate the contributions of different components of our algorithm. We test on three simulated environments used in [35]. More simulation experiment details can be found in the Appendix Section 4.3. We compare our method against the following baselines: (1) *BC* behavior clones on all prior and target data; (2) *MEDAL++* learns separate forward and backward policies from target forward and backward task demonstrations and performs reset-free learning using an adversarially trained classifier as a reward signal; (3) *MEDAL++ with prior data* modifies MEDAL++ to a single language-conditioned multi-task policy and adds all prior demonstration data into the replay buffer; (4) *ARIEL+VLM* modifies ARIEL [11] to use our VLM reward models as reward signal, instead of a handcrafted ground-truth reward. As shown in Figure 2 our method RoboFuME consistently outperforms prior methods in all tasks, achieving success rates at least 20% higher than all baselines within 200k steps of online fine-tuning. We also show additional ablations and analysis in Appendix Section 4.4

Discussion We introduced an autonomous framework that leverages existing diverse prior robot demonstration datasets and improves performance in a new robot manipulation skill by finetuning online. Integrating this work with new VLM models that can exhibit robust zero-shot performance on unseen manipulation tasks and improving the reset efficiency of this framework are promising directions for future research.

References

[1] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[5] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[6] Vikash Kumar, Emanuel Todorov, and Sergey Levine. Optimal control with learned local models: Application to dexterous manipulation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 378–383. IEEE, 2016.

[7] Ali Ghadirzadeh, Atsuto Maki, Danica Kragic, and Mårten Björkman. Deep predictive policy training using reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2351–2358. IEEE, 2017.

[8] Kai Ploeger, Michael Lutter, and Jan Peters. High acceleration reinforcement learning for real-world juggling with binary rewards. In *Conference on Robot Learning*, pages 642–653. PMLR, 2021.

[9] Abhishek Gupta, Justin Yu, Tony Z Zhao, Vikash Kumar, Aaron Rovinsky, Kelvin Xu, Thomas Devlin, and Sergey Levine. Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6664–6671. IEEE, 2021.

[10] Charles Sun, Jdrzej Orbik, Coline Manon Devin, Brian H Yang, Abhishek Gupta, Glen Berseth, and Sergey Levine. Fully autonomous real-world reinforcement learning with applications to mobile manipulation. In *Conference on Robot Learning*, pages 308–319. PMLR, 2022.

[11] Homer Rich Walke, Jonathan Heewon Yang, Albert Yu, Aviral Kumar, Jędrzej Orbik, Avi Singh, and Sergey Levine. Don't start from scratch: Leveraging prior data to automate robotic reinforcement learning. In *Conference on Robot Learning*, pages 1652–1662. PMLR, 2023.

[12] Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, and Sergey Levine. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017.

[13] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Reset-free lifelong learning with skill-space planning. *arXiv preprint arXiv:2012.03548*, 2020.

[14] Henry Zhu, Justin Yu, Abhishek Gupta, Dhruv Shah, Kristian Hartikainen, Avi Singh, Vikash Kumar, and Sergey Levine. The ingredients of real-world robotic reinforcement learning. *arXiv preprint arXiv:2004.12570*, 2020.

[15] Sehoon Ha, Peng Xu, Zhenyu Tan, Sergey Levine, and Jie Tan. Learning to walk in the real world with minimal human effort. *arXiv preprint arXiv:2002.08550*, 2020.

[16] Archit Sharma, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Autonomous reinforcement learning via subgoal curricula. *Advances in Neural Information Processing Systems*, 34: 18474–18486, 2021.

[17] Abhishek Gupta, Corey Lynch, Brandon Kinman, Garrett Peake, Sergey Levine, and Karol Hausman. Demonstration-bootstrapped autonomous practicing via multi-task reinforcement learning. *arXiv preprint arXiv:2203.15755*, 1, 2022.

[18] Archit Sharma, Rehaan Ahmad, and Chelsea Finn. A state-distribution matching approach to non-episodic reinforcement learning. *arXiv preprint arXiv:2205.05212*, 2022.

[19] Archit Sharma, Ahmed M Ahmed, Rehaan Ahmad, and Chelsea Finn. Self-improving robots: End-to-end autonomous visuomotor reinforcement learning. *arXiv preprint arXiv:2303.01488*, 2023.

[20] Kelvin Xu, Zheyuan Hu, Ria Doshi, Aaron Rovinsky, Vikash Kumar, Abhishek Gupta, and Sergey Levine. Dexterous manipulation from images: Autonomous real-world rl via substep guidance. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5938–5945. IEEE, 2023.

[21] Nair Ashvin, Dalal Murtaza, Gupta Abhishek, and L Sergey. Accelerating online reinforcement learning with offline datasets. *CoRR, vol. abs/2006.09359*, 2020.

[22] Rafael Rafailov, Tianhe Yu, Aravind Rajeswaran, and Chelsea Finn. Offline reinforcement learning from images with latent space models. In *Learning for Dynamics and Control*, pages 1154–1168. PMLR, 2021.

[23] Jiafei Lyu, Xiaoteng Ma, Xiu Li, and Zongqing Lu. Mildly conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1711–1724, 2022.

[24] Alex Beeson and Giovanni Montana. Improving td3-bc: Relaxed policy constraint for offline learning and stable online fine-tuning. *arXiv preprint arXiv:2211.11802*, 2022.

[25] Jialong Wu, Haixu Wu, Zihan Qiu, Jianmin Wang, and Mingsheng Long. Supported policy optimization for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:31278–31291, 2022.

[26] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pages 1702–1712. PMLR, 2022.

[27] Max Sobol Mark, Ali Ghadirzadeh, Xi Chen, and Chelsea Finn. Fine-tuning offline policies with optimistic action selection. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

[28] Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. *arXiv preprint arXiv:2303.05479*, 2023.

[29] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

[30] Yecheng Jason Ma, William Liang, Vaidehi Som, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control. *arXiv preprint arXiv:2306.00958*, 2023.

[31] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. *arXiv preprint arXiv:2109.13396*, 2021.

[32] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[33] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191, 2020.

[34] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

[35] Aviral Kumar, Anikait Singh, Frederik Ebert, Yanlai Yang, Chelsea Finn, and Sergey Levine. Pre-training for robots: Offline rl enables learning new tasks from a handful of trials. *arXiv preprint arXiv:2210.05178*, 2022.

[36] Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, et al. Bridgedata v2: A dataset for robot learning at scale. *arXiv preprint arXiv:2308.12952*, 2023.

[37] Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 8547–8556, Red Hook, NY, USA, 2018. Curran Associates Inc.

[38] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[39] 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.

[40] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142, 2020.

[41] Wenxuan Zhou, Sujay Bajracharya, and David Held. Plas: Latent action space for offline reinforcement learning. In *Conference on Robot Learning*, pages 1719–1735. PMLR, 2021.

[42] Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[43] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[44] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[45] Jonathan Ho, Jayesh Gupta, and Stefano Ermon. Model-free imitation learning with policy optimization. In *International conference on machine learning*, pages 2760–2769. PMLR, 2016.

[46] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[47] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58. PMLR, 2016.

[48] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[49] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. *arXiv preprint arXiv:1904.07854*, 2019.

[50] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[51] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[52] Wenhao Yu, Nimrod Gileadi, Chuyuan Fu, Sean Kirmani, Kuang-Huei Lee, Montse Gonzalez Arenas, Hao-Tien Lewis Chiang, Tom Erez, Leonard Hasenclever, Jan Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.

[53] Yuqing Du, Ksenia Konyushkova, Misha Denil, Akhil Raju, Jessica Landon, Felix Hill, Nando de Freitas, and Serkan Cabi. Vision-language models as success detectors. *arXiv preprint arXiv:2303.07280*, 2023.

4 Appendix

4.1 Method Details

4.1.1 Details on Pre-training

In addition to updating the policy using CalQL, we regularize policy learning with a behavior cloning (BC) loss, which encourages the behaviors to stay close to the seen demonstrations. Not only does this regularization improve the performance of the offline pre-training, but we find that it also makes it less likely for the autonomous fine-tuning procedure to exploit false positive rewards from the VLM reward model. The weight of the BC regularization term is chosen such that the scales of the RL loss and the BC loss are similar throughout the pre-training phase.

4.1.2 Details on Reward Model Design Choice

Leveraging existing vision-language models offers a number of benefits compared to utilizing a pre-trained visual representation or training a reward model from scratch using in-domain data: First, VLMs are trained on an Internet-scale dataset that contains diverse image and language contents. Such models possess better inductive biases and thus, can be more robust to natural shifts, such as perturbations to lighting conditions, or distractor objects that might be seen at test time. Second, since VLMs can take both visual and language information as input, they provide a natural interface for communicating the current observation and current task to the model when requesting a reward label.

4.1.3 Overview of RoboFuME

We present an overview of our system in Algorithm 1.

Algorithm 1: RoboFuME

```
Initialize agent $\mathcal{A} = \{\phi, \pi, Q\}$ and pre-trained VLM $\hat{r}$.
Initialize forward and backward tasks $\tau_f, \tau_b$.
// Prepare data and train VLM reward classifier.
$\mathcal{D}_{\text{prior}}, \mathcal{D}_f, \mathcal{D}_b, \mathcal{D}_{☹} \leftarrow$ load_data().
$\hat{r} \leftarrow$ finetune_vlm($\hat{r}, \{\mathcal{D}_{\text{prior}}, \mathcal{D}_f, \mathcal{D}_b, \mathcal{D}_{☹}\}$).
// Offline pre-training phase.
$\mathcal{A}$.update_buffer($\mathcal{D}_{\text{prior}}, \mathcal{D}_f, \mathcal{D}_b$).
for $t = 1$ to $T_{offline}$ do
    $\mathcal{A}$.update_params_with_calql().
// Online fine-tuning phase.
$s \leftarrow$ env.reset(); $l \leftarrow \tau_f$.get_task_lang().
for $t = 1$ to $T_{online}$ do
    $a \leftarrow \mathcal{A}$.act($s, l$); $s' \leftarrow$ env.step($a$).
    $\mathcal{A}$.update_buffer({$s, a, s', \hat{r}(s)$}).
    for $i = 1$ to $N_{utd\_ratio}$ do
        $\mathcal{A}$.update_params_with_calql().
    if switch then
        // Switch task after a fixed interval.
        $l \leftarrow$ env.switch($\tau_f, \tau_b$).get_task_lang().
    if interrupt then
        // Allow occasional human intervention.
        $s \leftarrow$ env.reset(); $l \leftarrow \tau_f$.get_task_lang().
    else
        $s \leftarrow s'$.
```

4.2 Details on Real Robot Experiments

We run real robot experiments in five real-world tasks, illustrated in Figure 3. For tasks involving deformable objects (the two cloth tasks) we manually reset the object to the initial forward pose every 15-25 episodes, and for the rest of the tasks we reset every 30-35 episodes. Tasks that use the kitchen-sink environment (*pot lid* and *pot pnp*) frequently experience episode interruptions when the robot arm applies more than the maximum allowed torque, for example, when close to the sink borders. All tasks use 50 forward and 50 backward demos for the target task, and fewer than 20 combined trajectories of failures. We use demos from the BridgeDataV2

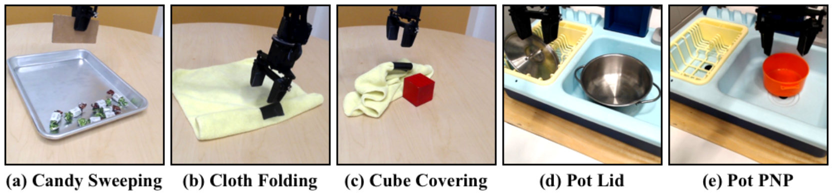

Figure 3: **Illustrations of the five real-world evaluation tasks.** (a) Sweep candies to the top of the tray. (b) fold the yellow cloth. (c) cover a red wooden cube using the cloth. (d) place the lid on top of the metallic pot. (e) move the orange pot from the sink to the drying rack.

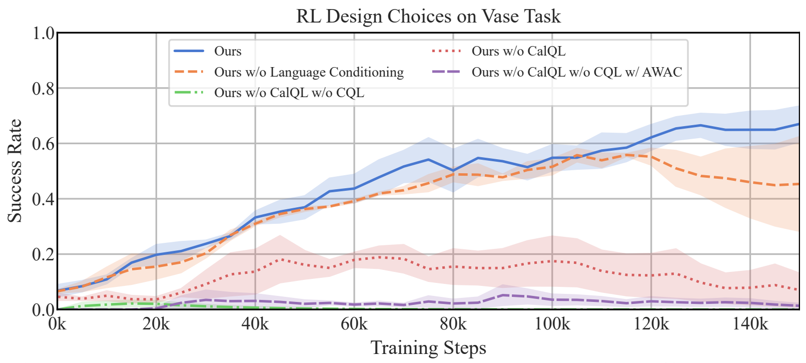

Figure 4: **Performance of our method on the Vase simulated task with different actor-critic update objectives.** Fine-tuning with CalQL is critical to obtain stable improvements on this task, as training with CQL, AWAC, or SAC yields poor performance. We also find that language conditioned policies perform slightly better than one-hot task IDs in simulation.

[31, 36] for pre-training our language-conditioned policy, selecting approximately 1,000 trajectories with relevant behaviors per task.

4.3 Details on Simulation Experiments

We consider three bin-sorting tasks in which different objects (a vase, a tiny bench, and a dumbbell weight) have to be placed on the correct bin based on the language instruction, given only a sparse binary reward. We provide 10 forward and reset demonstrations for each task, 30 failure demos, and 10 demos each for 20 prior tasks that show picking and placing diverse objects on the same environment. For all methods that require online experience, we reset the environment every 1,000 environment steps, i.e. every 25 episodes of interactions.

4.4 Ablations and Analysis

Ablations on RL Algorithm Design Choices. We evaluate our method trained with different critic and actor optimization procedures on the Vase simulated task, shown in Figure 4. Training with CalQL was the only method that yielded strong improvements in this task, with the other methods either failing completely or obtaining very poor performance. We find that training without the CalQL stabilizes training, while the losses for other methods would explode given the limited data.

Ablations on Reward Models. We compare our VLM reward function against other choices of automatic reward functions on the Vase simulated task in Figure 5. VICE [37] adversarially trains a binary classifier using positive samples from successful demonstrations, and labeling online experience as negative. We find that offline pre-training sufficiently limits the exploitation of the frozen VLM reward, outperforming VICE and thus, bypassing the need for adversarially trained reward functions. Such adversarial training can often learn to discriminate based on spurious shifts in the real world, such as lighting or scene changes, leading to instability in training outside simulation. VIP [29] trains a representation function such that the distance in representation space between the current observation and a goal image can be used to construct a dense reward function. We find that in the Vase simulated task, VIP fails to obtain good behaviors. Qualitatively, we observe VIP to be prone to false positives, which are exploited by the RL algorithm. To test the importance of the VLM large-scale pre-training compared to our fine-tuning procedure, we train a CNN classifier from scratch using the same data as we used to fine-tune the VLM, leading to unsatisfactory performance compared to fine-tuning a VLM.

How Accurate is the VLM Reward? We analyze the performance of the VLM reward over the course of fine-tuning for real-robot experiments. In Table 2, we report the false positive rate, false negative rate, accuracy,

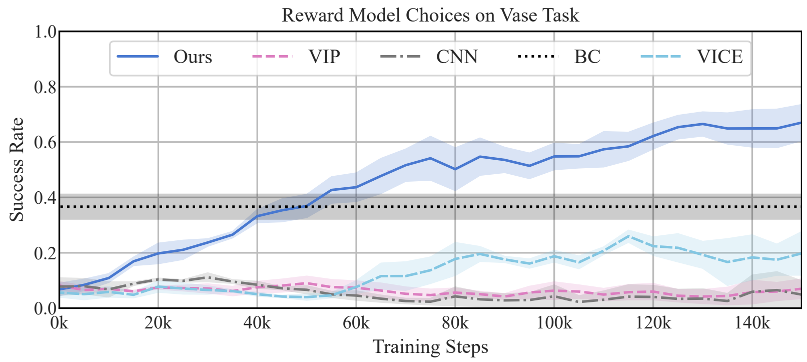

Figure 5: **Performance of our method on the simulated Vase task using different reward functions.** Our method uses a fine-tuned VLM reward function and outperforms VICE rewards, whereas CNN and VIP rewards fail to improve online.

Task	FP	FN	Accuracy	Precision
Cloth Covering	6.3%	80.9%	89.4%	15.3%
Cloth Folding	1.2%	59.8%	84.1%	92.0%
Pot PNP	6.1%	81.3%	86.9%	24.3%

Table 2: **VLM reward model accuracy during real robot fine-tuning.** The low false positive (FP) rate indicates that online training has minimal reward exploitation.

and precision metrics for the VLM reward. The metrics are computed on the data collected during fine-tuning against a hand-engineered ground truth reward. We observe that while false negative rates are high, false positive rates are low across all tasks. This asymmetry is crucial for successful RL fine-tuning, as RL policies can learn poor behaviors by exploiting false positives, but labeling some successful rollouts as negatives does not necessarily impede learning.

Is the learned policy robust to distractors? We find that policies learned by ROBOFUME (both offline and after fine-tuning) to be more robust to scene distractors on the *candy sweeping* task, as reported in Table 3. The policies were trained without any distractors, but multiple objects not seen during training were placed in the background during evaluation. ROBOFUME policies retained 68% of its original performance, compared to BC which retained only 10% of its original performance. We hypothesize that BC might be more sensitive to spurious features, whereas ROBOFUME learns from more predictive features, leading to more robust policies.

How Important is Diverse Prior Data and Language Conditioning? We ablate the contribution of diverse prior data and language-conditioned policies to ROBOFUME by evaluating the offline performance on the *candy sweeping* task, reported in Table 4. When pre-training without using prior data, that is, exclusively using target data, our method is able to sweep less than half the amount of candies on average. Similarly, we find that one-hot task encodings perform substantially worse than language-conditioned policies, as the prior dataset used in real-robot training is larger and more diverse compared to the simulation experiments.

4.5 More Related Works

Offline RL. Offline RL algorithms [32, 33, 38–41] provide a framework for initializing robot manipulation policies from offline demonstrations or interaction datasets. Such algorithms can also be extended to include an online fine-tuning phase after training a policy offline [21–28]. Our work utilizes a recent offline RL algorithm, calibrated Q-learning (CalQL) [28], a state-of-the-art method that effectively learns from offline data and continuously improves the policy's performance online by explicitly correcting the scale of the learned Q-values. We show that integrating CalQL helps our framework effectively utilize diverse prior datasets that have large distribution shifts from real-world online interactions.

Reset-free RL. Training an RL policy on a real robot typically requires manual environment resets. To eliminate such need to manually reset environments, prior works have studied approaches to learn robot policies in a 'reset-free' setup. Some work [9, 13, 15, 17, 20] cast the 'reset-free' learning problem as a multi-task learning problem, observing that by learning a set of tasks where some of the tasks could reset others, an agent could then be trained to perform all of those tasks without needing manual resets. Other works [10–12, 14, 16, 18, 19] learn both a task policy and a reset policy for performing the task and resetting to the initial state distribution. Our work takes an approach between the two classes of approaches, learning a language-conditioned multi-task policy that can perform both the target task and the reset for the target task. Most of these prior works learn

Task	BC	RoboFuME (offline)	RoboFuME (fine-tuned @30k)
Candy Sweeping	31% $\rightarrow$ 3%	47% $\rightarrow$ 31%	**66%** $\rightarrow$ **45%**

Table 3: **Robustness of learned policy to distractors.** Entries in this table show the performance of the learned policy “before” $\rightarrow$ “after” adding distractors to the scene in the candy-sweeping task. Our system learns a policy that is much more robust to the distractors.

Task	RoboFuME (offline)	RoboFuME w/o Prior Data	RoboFuME w/o Language Cond.
Candy Sweeping	47%	23%	13%

Table 4: **Evaluating effectiveness of prior data and language conditioned policies.** Results show that using prior data and using language conditioning positively affected the offline performance of our system.

from scratch rather than incorporating prior data and assume that a reward function is available. ARIEL [11] combines incorporating prior data with reset-free learning but assumes a hand-crafted reward function for each environment. They also collect their own prior dataset on the same robot hardware set-up as their target task. MEDAL++ [19] learns a reward classifier with demonstration and online interaction data via adversarial training, but does not consider incorporating diverse prior data. Leveraging diverse, off-the-shelf prior demonstration datasets is desirable since these datasets are readily available to use and can help a system obtain a policy initialization for efficient fine-tuning on a target task. Our system offers an approach to both incorporate diverse prior data and improve the autonomy of the fine-tuning phase by learning a model for predicting rewards. In particular, we found out that by leveraging diverse demonstration data, our system requires only about 3 hours of training in the real world compared to 10-30 hours in MEDAL++.

Reward learning. Early works have studied the problem of learning a reward or cost function in imitation learning. These works leverage inverse optimal control (IOC) or inverse reinforcement learning (IRL) to extract a reward function directly from expert demonstrations [42–44]. With the advent of deep neural networks, more recent works have explored learning a reward model for an imitation learning or RL policy [18, 19, 37, 45–48]. When using classifier-based reward models in reinforcement learning, RL agents can exploit the learned model by exploring states unseen during classifier training, tricking the model to output incorrect rewards. To solve such an exploitation issue, many works that learn reward models leverage adversarial learning, where a system learns a discriminator that identifies states similar to those in demonstrations as positives and those visited by the policy as negatives [18, 19, 37, 46]. However, prior work has found this training objective to be sensitive to distribution shifts between offline and online setups, such as lighting and camera view changes [49]. In this work, we fine-tune vision language models (VLM), pre-trained on internet-scale data, to construct a reward model. Large scale pre-training can learn representations that are robust to natural variations such as lighting, camera shifts and distractors [29, 50].

Leveraging pre-trained representations as reward predictors. Several recent works have shown positive results in utilizing pre-trained vision models [29, 51], large language models (LLMs) [52] or vision language models (VLMs) [53] as reward predictors. We tried VIP [29], a method that pre-trains a visual representation for generating dense reward functions for novel robotic tasks, and found it insufficient for the real-world robot fine-tuning setup. In this work, we fine-tune a pre-trained VLM [34] and find that it performs most effectively as a reward model. Our proposed system is flexible and can easily be adapted to use other pre-trained visual representations and VLMs.