# PROGRESSOR: A Perceptually Guided Reward Estimator with Self-Supervised Online Refinement

Supplementary Material

## 7. PROGRESSOR Training Details

### 7.1. Architecture and Training

PROGRESSOR can, in principle, be trained with any visual encoding architecture, requiring only minor modifications to the final layer to predict Gaussian parameters. In our experiments, we utilize the standard ResNet34 model [16], replacing its final fully-connected layer with an MLP of size [512, 512, 128]. Given that our method processes triplets of inputs $(\boldsymbol{o}_i, \boldsymbol{o}_j, \boldsymbol{o}_g)$, the resulting representation has a size of $[128 \times 3]$. This representation is then fed into an MLP with layers of size $[128 \times 3, 2048, 256]$. Finally, two prediction heads are derived from the 256-dimensional output, predicting $\mu$ and $\log \sigma^2$.

We pretrain PROGRESSOR for 30000 steps using the EPIC-KITCHENS dataset [6] for the real-world experiments, and for 10000 steps for experiments performed in simulation. In the pretrianing steps for both our simulated and real-world experiments, we first sample a trajectory (i.e., a video clip) from the pretraining dataset. We then randomly select an initial frame $\boldsymbol{o}_i$ as well as a goal frame $\boldsymbol{o}_g$ from the selected trajectory such that $\|g-i\| \leq 2000$. Finally, we uniformly randomly select an intermediate frame $\boldsymbol{o}_j$, where $i < j < g$.

### 7.2. Hyperparameters

| | Simulation | Real-World RWR |
|---|---|---|
| $\alpha$ | 0.4 | 0.4 |
| $\beta$ | 0.9 | – |

Table 2. Hyperparameters used by PROGRESSOR for experiments performed in simulation and the real world.

Table 2 lists the hyperparameters used for both the simulation online RL and real-world offline RL experiments. A consistent $\alpha = 0.4$ is used across all experiments. The push-back decay factor is employed exclusively in the simulated online RL experiments.

## 8. Simulation Experiment Details

In this section, we describe the tasks and the data generation process employed using the MetaWorld environment [44] for our simulation experiments.

### 8.1. Meta-World Tasks

We took six diverse tasks from the Meta-World environment [44], described in Table 3. In all tasks, the position of the target object, such as the drawer or hammer, is randomized between episodes.

| Task | Task Description |
|---|---|
| `door-open` | Open a door with a revolving joint. |
| `drawer-open` | Open a drawer. |
| `hammer` | Hammer a screw on the wall. |
| `peg-insert-side` | Insert a peg sideways. |
| `pick-place` | Pick and place a puck to a goal. |
| `reach` | Reach a goal position. |

Table 3. Meta-World [44] task descriptions.

### 8.2. Expert Data Generation

To collect expert trajectories for our simulated experiments, we execute Meta-World's oracle policies. For each task, we generated 100 successful rollouts for training and 10 for testing. This dataset is subsequently used for pretraining PROGRESSOR in our simulated experiments, following the steps outlined in Section 7.1.

## 9. Real-World Robot Experiment Details

### 9.1. Robotic Experiment Setup

The real-robot experiments are performed using a Universal Robots UR5 robot arm equipped with a Robotiq 3-Finger Gripper (Figure 8). The setup includes two RealSense cameras: one mounted on the robot's wrist that images the gripper, and the second on a fixed tripod facing the robot.

### 9.2. Robotic Demonstration Data Collection

We collect demonstrations by teleoperating the UR5 using a Meta Quest 3 controller [39]. We record each demonstration at 30 Hz for 400 steps. Figure 9 shows video frame sequences from correct and incorrect demonstrations for the four real-world tasks.

### 9.3. Task Descriptions

The tasks involve a variety of object manipulation challenges designed to test reward weighting in offline Reinforcement Learning. In the `Drawer-Close` task, the objective is to close a drawer starting from an open position. The `Drawer-Open` task requires the agent to pull a drawer open from a closed state. In the `Push-Block` task, the

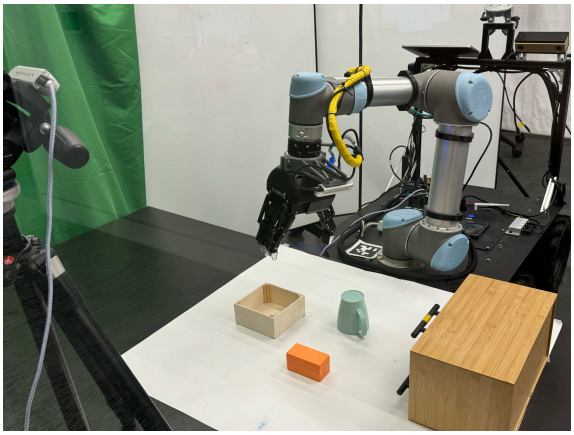


Figure 8. The real-world experiments were conducted using a Universal Robots UR5 robot arm equipped with a Robotiq 3-Finger Gripper. One RealSense camera is mounted to the end-effector and the other RealSense camere is fixed in the environment using a tripod (visible in the left of the image).

goal is to push a block toward a specified target, a cup, ensuring the block moves into proximity with the cup. Finally, `Pick-Place-Cup` involves picking up a cup and carefully placing it into a designated box. Figure 10 displays the goal frames representing the completion of each task.

### 9.4. Training and Evaluation Details

| Task | Success Criterion |
|---|---|
| `Drawer-Open` | Drawer is open by more than 5 cm. |
| `Drawer-Close` | Drawer is within 1 cm of being fully closed. |
| `Push-Block` | Block is within 5 cm of the cup. |
| `Pick-Place-Cup` | Cup is placed inside the box. |

Table 4. Success criterion for the real-robot experiments.

Our few-shot offline RL implementation builds upon the Action-Chunking Transformer (ACT) [46]. The inputs to the model include (i) two 640×480 RGB images from the RealSense cameras, and (ii) proprioceptive data consisting of the 6-DoF joint angles and the binary open or close state of the gripper. The action space consists of the 6-DoF translational and rotational velocity of the end-effector and a binary open or close command of the gripper.

In the reward-weighted regression (RWR) setup, rewards are computed by providing all reward models with the final frame of a correct demonstration from each task as the goal image. For all reward predictions, frames from the fixed RealSense camera were used. Figure 10 illustrates the goal images used for the four robotic tasks, which were consistently used across all reward predictions. The temperature scale in RWR was set to $\omega = 0.1$ for all tasks.

| Hyperparameter | Value |
|---|---|
| prediction horizon | 30 |
| learning rate | $10^{-5}$ |
| batch size | 64 |
| epochs | 5000 |
| $\omega$ | 0.1 |

Table 5. Hyperparameters for reward-weighted ACT training.

Table 5 shows the hyperparameters used to train the {PROGRESSOR, VIP, R3M}-RWR-ACT models. For the vanilla ACT, we used the same hyperparameters as listed in Table 5, except for $\omega = 0$.

During inference, the model predicts a sequence of actions (a "chunk") of length 30 and then executes each action before predicting the next chunk. We did not use a temporal ensemble as proposed by Zhao et al. [46], since we found that it causes the gripper to drift and negatively impacts performance.

For evaluation, we conduct 20 test rollouts for each task and report the success rate. The success criteria for each task are outlined in Table 4.

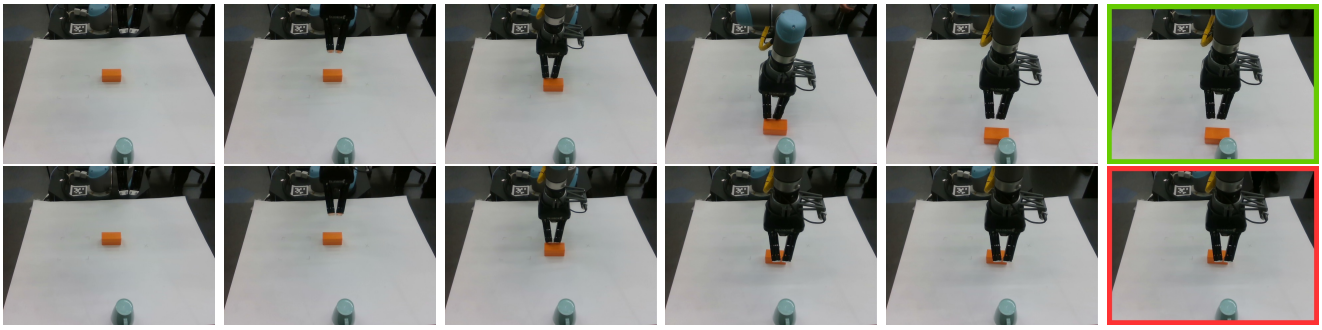
## 10. Ablation

In this section, we present an ablation study evaluating different values of the push-back decay factor ($\beta$) while training a DrQ-v2 agent on Meta-World's `hammer` task, using a fixed seed of 121. The case of $\beta = 0$ (PROGRESSOR without Push-back) is discussed in the main paper. Figure 11 depicts the environment rewards accumulated during training. As shown in the figure, the agent achieves higher rewards with $\beta = 0.9$.

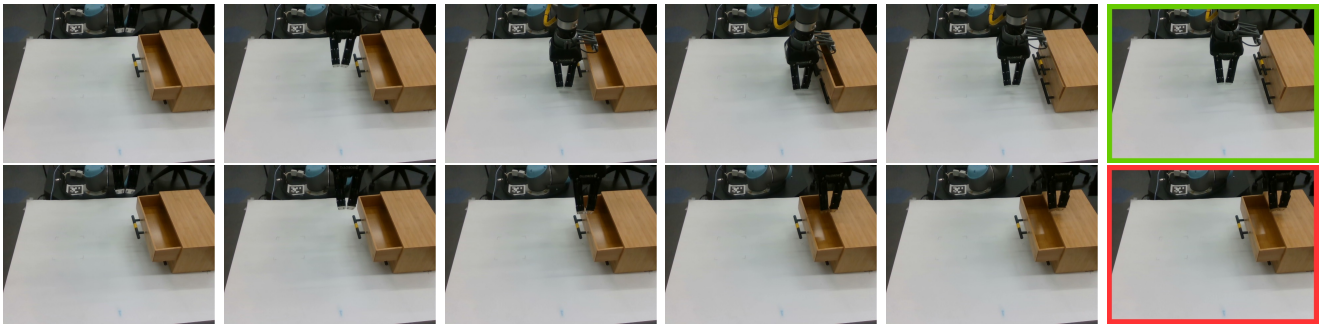
## 11. Qualitative Analysis

Figure 12 presents zero-shot reward predictions from PROGRESSOR pretrained on the EPIC-KITCHENS dataset. This figure serves as an extension to Figure 7 for completeness. It includes zero-shot reward predictions for sample correct trajectories from our collected real-robot demonstrations for the tasks `Drawer-Open`, `Drawer-Close`, and `Push-Block`.

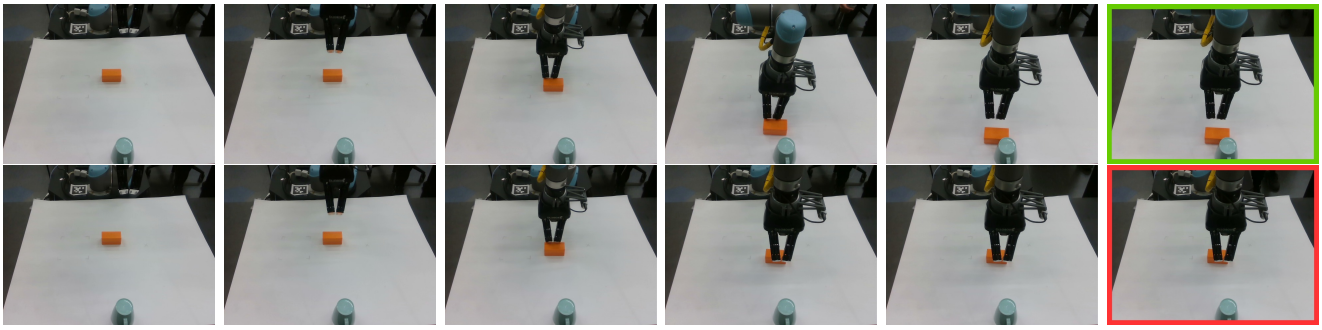
For completeness, we include plots comparing the mean reward predictions of R3M and VIP—using their publicly available pretrained weights—with the reward estimated by PROGRESSOR. The plots in Figure 13 provide a qualitative comparison of reward predictions from our robotic demonstration dataset for three additional real-world tasks: `Drawer-Close`, `Push-Block`, and `Pick-Place-Cup`. As illustrated in the figure, for all tasks, our reward model consistently predicts lower average rewards for the sub-trajectories in the incorrect demonstrations, where the failures occur.



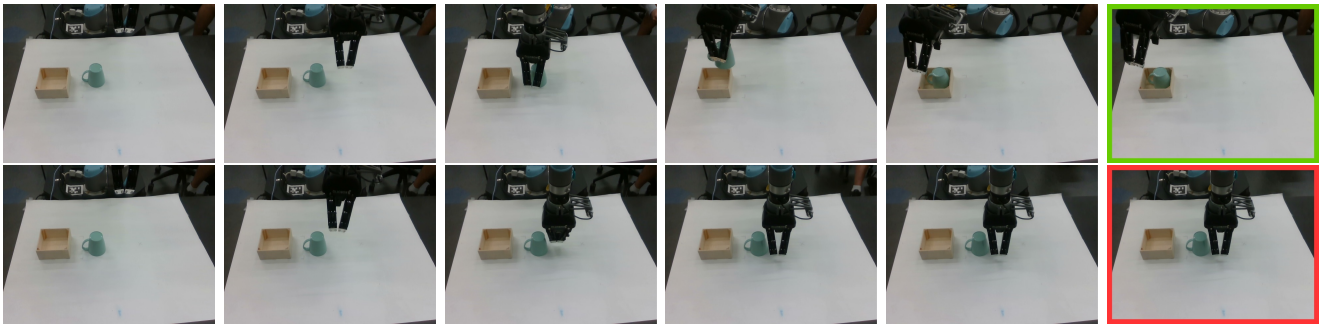

(a) `Push-Block`: (top) correct and (bottom) incorrect




(b) `Drawer-Close`: (top) correct and (bottom) incorrect

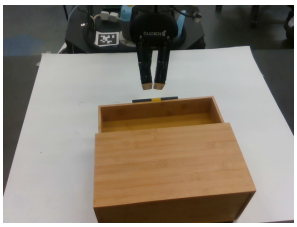



(c) `Push-Block`: (top) correct and (bottom) incorrect

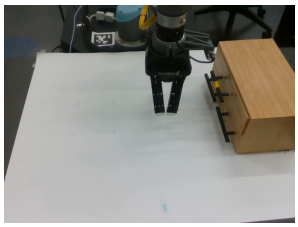



(d) `Pick-Place-Cup`: (top) correct and (bottom) incorrect

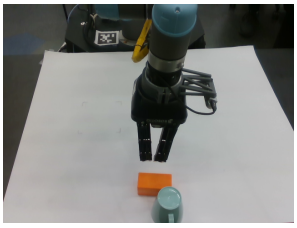
Figure 9. Correct and incorrect demonstrations (every 80th frame) for each real-robot task from a third-person camera view used in our experiments. To see how PROGRESSOR and the baselines differentiate between correct and incorrect trajectories, see Figure 6 (`Drawer-Open`) and Figure 13 (other three tasks).



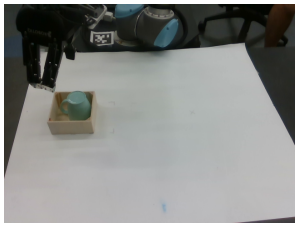
(a) `Drawer-Open`



(b) `Drawer-Close`



(c) `Push-Block`



(d) `Pick-Place-Cup`

Figure 10. The goal images that we use for each task for reward weighting in the offline reinforcement learning experiments.

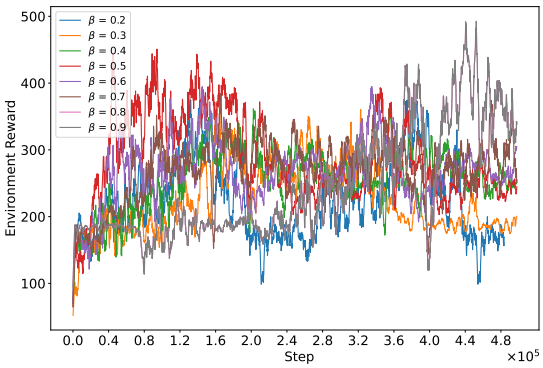
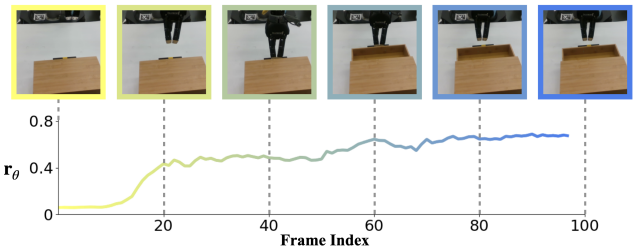
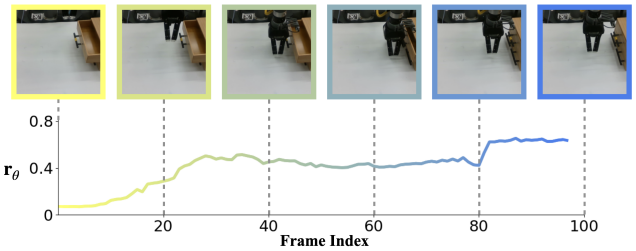


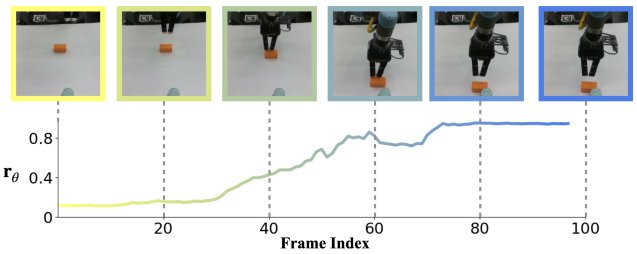

Figure 11. Visualization of policy learning performance in Meta-World's `hammer` environment, evaluating the effect of different values for the push-back decay factor $\beta$. The plot highlights the accumulated rewards over training, demonstrating how varying $\beta$ values influences the agent's ability to optimize performance.




(a) Reward plot for `Drawer-Open` task

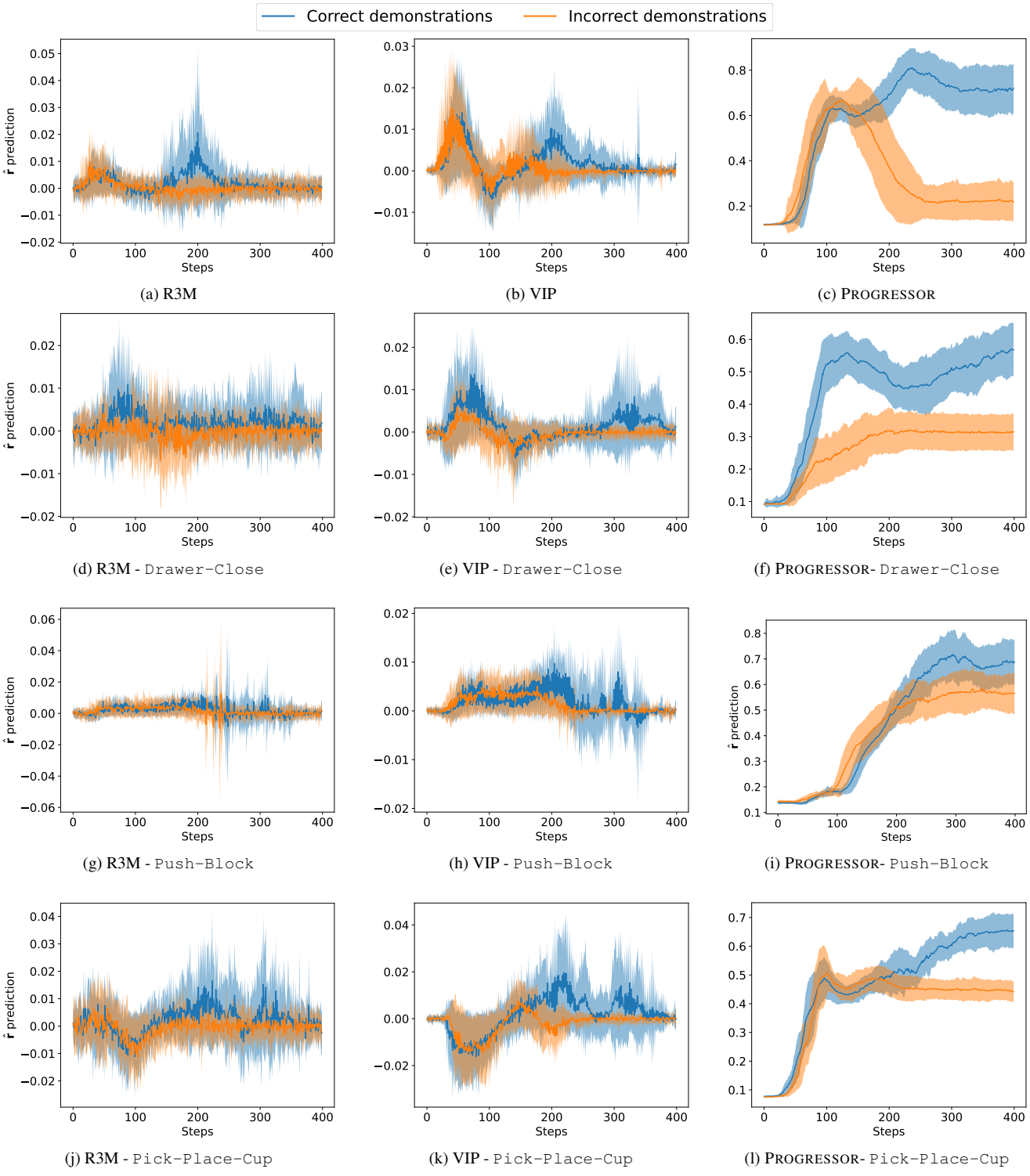



(b) Reward plot for `Drawer-Close` task




(c) Reward plot for `Push-Block` task

Figure 12. Visualization of the predicted reward by PROGRESSOR, pretrained on EPIC-KITCHENS and evaluated zero-shot on correct robotic demonstrations for (**a**) `Drawer-Open` (**b**) `Drawer-Close` and (**c**) `Push-Block` tasks.




(a) R3M (b) VIP (c) PROGRESSOR

(d) R3M - `Drawer-Close` (e) VIP - `Drawer-Close` (f) PROGRESSOR- `Drawer-Close`

(g) R3M - `Push-Block` (h) VIP - `Push-Block` (i) PROGRESSOR- `Push-Block`

(j) R3M - `Pick-Place-Cup` (k) VIP - `Pick-Place-Cup` (l) PROGRESSOR- `Pick-Place-Cup`

Figure 13. Mean reward predictions $\hat{r}$ of (**left column**) R3M, (**middle column**) VIP, and (**right column**) PROGRESSOR for correct and incorrect demonstrations for the `Drawer-Close`, `Push-Block`, and `Pick-Place-Cup` tasks. PROGRESSOR provides reward predictions (weights) that better differentiate between correct and incorrect trajectories, consistently outperforming the baseline models.