

## A Additional Simulation Details

In this section, we provide additional details about BiGym.

**Observation Spaces.** For image observations, we allow users to specify the resolution of the images, where the default resolution is  $84 \times 84$ . Higher resolution may allow learning better policies, but we find the default value works across tasks. In the *whole-body* mode, the proprioception state  $s_{\text{proprio}}^{\text{fb}} \in \mathbb{R}^{76} = \{s_{\text{qpos}}^{\text{rb}}, s_{\text{qvel}}^{\text{rb}}, s_{\text{grip}}\}$ , where  $s_{\text{qpos}}^{\text{rb}} \in \mathbb{R}^{37}$  is the joint angle positions of the robot,  $s_{\text{qvel}}^{\text{rb}} \in \mathbb{R}^{37}$  is the corresponding velocities, and  $s_{\text{grip}} \in \mathbb{R}^2$  is the gripper opening amount of both grippers. On the contrary, the *bi-manual* mode greatly simplifies the locomotion by replacing the lower-body control with a predefined controller, i.e., a floating base. This reduces the dimension of  $s_{\text{qpos}}^{\text{rb}}$  and  $s_{\text{qvel}}^{\text{rb}}$  to  $s_{\text{qpos}}^{\text{bm}} \in \mathbb{R}^{29}$  and  $s_{\text{qvel}}^{\text{bm}} \in \mathbb{R}^{29}$ . Furthermore, an additional state  $s_{\text{base}} = (x, y, z, \theta) \in \mathbb{R}^4$  is included in  $s_{\text{proprio}}$  to indicate the position and orientation of the floating base. As a result,  $s_{\text{proprio}}^{\text{bm}} \in \mathbb{R}^{64} = \{s_{\text{qpos}}^{\text{bm}}, s_{\text{qvel}}^{\text{bm}}, s_{\text{base}}, s_{\text{grip}}\}$ .

**Action Spaces.** In the *whole-body* mode where the agent has the full control over the body, an action space  $\mathcal{A}_{\text{wb}} \in \mathbb{R}^{23}$  is defined as  $\mathcal{A}_{\text{wb}} = \{\mathcal{A}_{\text{arms}}, \mathcal{A}_{\text{legs}}, \mathcal{A}_{\text{torso}}, \mathcal{A}_{\text{grip}}\}$ , where  $\mathcal{A}_{\text{arms}} \in \mathbb{R}^{10}$  controls both arms,  $\mathcal{A}_{\text{legs}} \in \mathbb{R}^{10}$  controls the legs,  $\mathcal{A}_{\text{torso}} \in \mathbb{R}^1$  controls the main torso joints, and  $\mathcal{A}_{\text{grip}} \in \mathbb{R}^2$  controls the opening amount of grippers. In *bi-manual* mode, the user controls the floating base instead of the leg joints. Therefore the action space becomes  $\mathcal{A}_{\text{bm}} \in \mathbb{R}^{16} = \{\mathcal{A}_{\text{arms}}, \mathcal{A}_{\text{base}}, \mathcal{A}_{\text{grip}}\}$ , with  $\mathcal{A}_{\text{base}} \in \mathbb{R}^4$  controlling the delta actions  $(\delta x, \delta y, \delta z, \delta \theta)$  of the base.

**Simulation Performance.** We present the simulation speed in Figure 5. The benchmark was done on a headless server of NVIDIA L4 GPU and Intel Xeon Gold 6438Y+ CPU, in a single process. Benefiting from the highly optimised MoJoCo engine, BiGym runs at around 400FPS to 1400FPS depending on the number of cameras. The performance could be further improved by using parallel environments or MuJoCo XLA, which speeds up the execution with XLA just-in-time compilation.

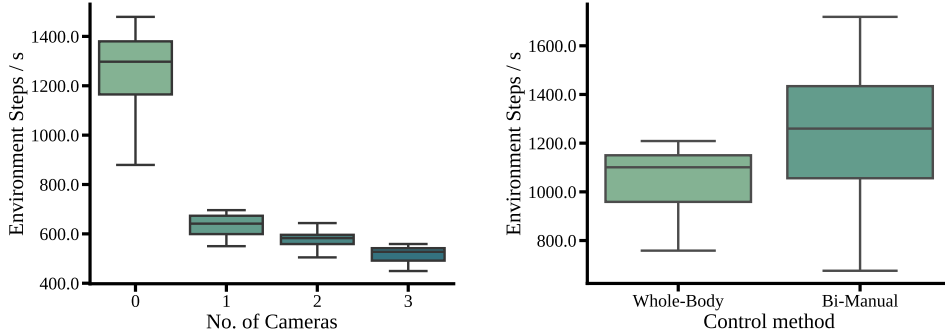


Figure 5: The environment run speed of BiGym with (a) different number of cameras and (b) different action modes. In (a), we use the bi-manual control method for measuring the performance.

## B Details of Task Success Detectors

In this section, we detail the definitions of all task success detectors.

### Reach Target Tasks.

- (1) `reach_target_single`: The distance from the robot left wrist to the target is smaller than a tolerance value. The default tolerance value is 0.1.
- (2) `reach_target_multi_modal`: The distance from either the robot left wrist or the right wrist is smaller than a tolerance value. The default tolerance value is 0.1.
- (3) `reach_target_dual`: The distance from the left wrist and the right wrist to their corresponding goals are smaller than a tolerance value. The default tolerance value is 0.1.

439 **Table-Top Manipulation Tasks.**

- 440 (4) `stack_blocks`: The three blocks are stacked on each other, i.e. in collision with each other, in a  
441 target region on the table.
- 442 (5) `move_plate`: The following conditions must be met: (a) the orientation of the plate is upright,  
443 (b) the plate is not colliding with the table, (c) the plate is colliding with the rack and (d) the robot  
444 has released the plate from its gripper.
- 445 (6) `move_two_plates`: Transfer two plates to the target rack and meet all conditions similar to the  
446 `move_plate` task.
- 447 (7) `flip_cup`: The following criteria must be met: (a) The cup is in collision with the counter. (b)  
448 The orientation of the cup is upright. (c) The robot has released the cup from its gripper.
- 449 (8) `flip_cutlery`: Similar to the `flip_cup` task, but cutlery is used instead.

450 **Dishwasher Tasks.**

- 451 (9) `dishwasher_open`: The joint angles of the dishwasher door and both trays are close to 1 with a  
452 tolerance value. The default value is 0.1.
- 453 (10) `dishwasher_close`: The joint angles of the dishwasher door and both trays are close to 0 with  
454 a tolerance value. The default value is 0.1.
- 455 (11) `dishwasher_open_trays`: The joint angles of dishwasher trays are close to 1 with a tolerance  
456 value. The default value is 0.1.
- 457 (12) `dishwasher_close_trays`: The joint angles of dishwasher trays are close to 0 with a tolerance  
458 value. The default value is 0.1.
- 459 (13) `dishwasher_load_plates`: All plates are in collision with the bottom tray of the dishwasher  
460 and the robot has released the plates from its grippers.
- 461 (14) `dishwasher_load_cups`: All cups are in collision with the middle tray of the dishwasher and  
462 the robot has released the cup from its gripper.
- 463 (15) `dishwasher_load_cutlery`: All cutlery are in collision with the dishwasher cutlery basket  
464 and the robot has released the cutlery from its gripper.
- 465 (16) `dishwasher_unload_plates`: All plates are moved from the bottom tray of the dishwasher  
466 to the drainer on the table, and placed onto the rack positioned on the counter-top.
- 467 (17) `dishwasher_unload_cups`: All cups are moved from the middle tray of the dishwasher to the  
468 cabinet and in collision with the cabinet counter. Additionally, all cups are released from the robot  
469 gripper.
- 470 (18) `dishwasher_unload_cutlery`: All cutlery are moved from the dishwasher basket to the tray  
471 and are in collision with the tray.
- 472 (19) `dishwasher_unload_plate_long`: All conditions of `dishwasher_close` and  
473 `dishwasher_unload_plates` must be met. Additionally, all plates are placed inside the  
474 wall cabinet. Finally, all joint angles of the wall cabinet doors are close to 0 with a tolerance. The  
475 default value is 0.1.
- 476 (20) `dishwasher_unload_cup_long`: Similar to `dishwasher_unload_plate_long` but with  
477 cups.
- 478 (21) `dishwasher_unload_cutlery_long`: Similar to `dishwasher_unload_cutlery_long` but  
479 with cutlery and instead of the cabinet, the cutlery must be placed in a closed drawer.

480 **Kitchen Counter Tasks.**

- 481 (22) `drawer_top_open`: The joint angle of the top drawer is close to 1 with a tolerance value. The  
482 default value is 0.1.
- 483 (23) `drawer_top_close`: The joint angle of the top drawer is close to 0 with a tolerance value. The  
484 default value is 0.1.
- 485 (24) `drawers_open_all`: The joint angles of all drawers are close to 1 with a tolerance value. The  
486 default value is 0.1.
- 487 (25) `drawers_close_all`: The joint angles of all drawers are close to 0 with a tolerance value. The  
488 default value is 0.1.
- 489 (26) `wall_cupboard_open`: The joint angle of two doors of the wall cupboard is close to 1 with a  
490 tolerance value. The default value is 0.1.

491 (27) `wall_cupboard_close`: The joint angle of two doors of the wall cupboard is close to 0 with a  
492 tolerance value. The default value is 0.1.  
493 (28) `cupboards_open_all`: The joint angles of the two doors and all drawers of the kitchen set are  
494 close to 1 with a tolerance value. The default value is 0.1.  
495 (29) `cupboards_close_all`: The joint angles of the two doors and all drawers of the kitchen set  
496 are close to 0 with a tolerance value. The default value is 0.1.  
497 (30) `take_cups`: All cups are in collision with the counter on the table and the robot has released  
498 the cups from its gripper.  
499 (31) `put_cups`: All cups are in collision with the cupboard shelf and the robot has released the cups  
500 from its gripper.  
501 (32) `pick_box`: The box is in collision with the counter and the robot has released the box from its  
502 grippers.  
503 (33) `store_box`: The box is in collision with the shelf and the robot has released the box from its  
504 grippers.  
505 (34) `saucepan_to_hob`: The saucepan is in collision with the hob and the robot has released the  
506 saucepan from its grippers.  
507 (35) `store_kitchenware`: Both the saucepan and the pan are in collision with the shelf, and the  
508 robot has released the objects from its grippers.  
509 (36) `sandwich_toast`: All the following conditions must be met: (a) The sandwich is in collision  
510 with the pan. (b) The orientation of the sandwich is either up or down. (c) The pan is in collision  
511 with the hob.  
512 (37) `sandwich_flip`: Similar to `sandwich_toast`. In addition the sandwich orientation must be  
513 flipped.  
514 (38) `sandwich_remove`: All the following conditions must be met: (a) The sandwich is in collision  
515 with the board. (b) The orientation of the sandwich is either up or down.  
516 (39) `store_groceries_lower`: All items are in collision with the shelf of the cabinet below the  
517 counter. Additionally, all items are released from the robot gripper.  
518 (40) `store_groceries_upper`: All items are in collision with the shelf of the cabinet on the wall.  
519 Additionally, all items are released from the robot gripper.

## 520 C Experiments

### 521 C.1 Implementation Details

522 We implemented all algorithms using PyTorch [50].

523 **ACT**. Following the official implementation<sup>4</sup>, we train a ResNet-18 encoder [51] to extract visual  
524 features and a transformer model to predict a sequence of actions. Inputs to the transformer model  
525 are multi-view image features and proprioceptive features from a conditional variational autoencoder  
526 (CVAE) [52]. During execution, we use receding horizon control for all tasks by training the policy  
527 to output an action sequence of length 16 and executing only the first step in the sequence. Following  
528 the official implementation, we enable *temporal ensembling* to improve the smoothness of the policy.

529 **Diffusion Policy**. Our implementation of Diffusion Policy closely follows the official release<sup>5</sup>. To be  
530 consistent with ACT, we use ResNet-18 as vision encoders for all camera observations. As discussed  
531 in Chi et al. [8], the Diffusion Policy is susceptible to the choice of backbones and their parameters:  
532 the UNet-1D backbone might outperform the causal transformer backbone in certain tasks and vice  
533 versa. Thus, we benchmark both the UNet-1D backbone and the causal Transformer backbone,  
534 and report the highest achieved performance between them in our main results. In addition, for all  
535 Diffusion Policy variants, we use action sequence length of 16 and execution length of 1, which we  
536 find to achieve strong performance in general. Following ACT, we also enable *temporal ensembling*  
537 for Diffusion Policies, which we find to be crucial for stabilising the inference.

<sup>4</sup><https://github.com/tonyzhaozh/aloha>

<sup>5</sup>[https://github.com/real-stanford/diffusion\\_policy](https://github.com/real-stanford/diffusion_policy)

**Other Baselines.** For BC and demo-driven RL baselines, we adopt the same network architectures which consist of an CNN-based image encoder and a fully-connected output head. The image encoder encodes each camera image with 3 layers of CNNs, each has kernel size 3 and 32 channels. In between the layers, we use SiLU activation function [53] and layer normalisation [54]. We flatten the CNN features and concatenate with the proprioception states to form the final observation feature vector. The head has 2 fully connected layers of dimension 512, and bottlenecks the output to dimension 64. After the bottleneck layer, we normalise the output with layer normalisation followed by tanh activation.

**Training Details.** We use a frame stack of 4, Adam optimiser [55] with a learning rate of 0.0001, and batch size of 256 for all IL and demo-driven RL algorithms. In addition, specifically for all RL algorithms, we follow AW-Opt [56] and keep the demonstration ratio for each batch to be 50% by using a separate demonstration replay buffer. This helps the exploration of the agent during sparse reward settings. We run 150K training steps for IL algorithms and 100K steps for demo-driven RL methods. We observe that all algorithms converge after 100K steps and longer training does not give additional performance boost. All results are averaged over the last 3 checkpoints.

## C.2 Results and Discussions

In Table 2, we provide the performance of IL and demo-driven RL methods on 40 BiGym tasks. Overall, we observe that BiGym tasks are challenging and pose a variety of unique and interesting challenges for future researches. We outline our observations as below:

**The mobile manipulation of articulated or rigid-body objects is challenging for the state-of-the-art algorithms.** BiGym has presented a series of tasks that involve interactions with articulated or rigid-body objects, which typically require high-precision manipulation, e.g., `move_two_plates`, `cupboards_open_all`, and `stack_blocks`. When coupled with the mobile base, these tasks become more challenging because (i) accurately measuring the grasping poses while moving is hard, and (ii) correctly estimating the posterior distribution of the states given partial history information of a POMDP is difficult. For instance, while we observe that ACT and Diffusion Policy achieve the overall best performance across all tasks, they still struggle in the seemingly simple tasks, e.g., `stack_blocks`, which requires the agent to pick 3 cubes, and stack them to a target region located on the other side of the table. We believe a more robust system with stronger memory mechanisms to track the “beliefs”, i.e., estimating the posterior distributions of the states, is necessary to solve such challenging BiGym tasks.

**The long-horizon tasks in BiGym requires both task and motion planning of the agent.** BiGym introduces a series of long-horizon tasks, e.g., `dishwasher_unload_cups_long` and `put_cups`. All algorithms fail on these tasks. Intuitively, these tasks are composed of multiple sub-tasks, and the difficulty level of achieving these long-horizon tasks grows exponentially at the same time. As model-free agents, our baselines are not capable of performing task-level reasoning. Hierarchical methods [12] could work as a better policy representation for these tasks. We leave it for future study.

**The complex policy space of BiGym requires carefully designed agent architectures.** We observe that almost on all tasks, ACT and Diffusion Policy achieves superior performance to BC and demo-driven RL baselines. We hypothesize this is because both ACT and Diffusion Policy utilise powerful policy classes based on generative representation learning, i.e., CVAE and Diffusion models, and they also use expressive network architecture such as transformers or UNets. In contrast, BC and all demo-driven RL approaches use simple CNN + MLP architectures. It is likely that these weaker architectures struggle to deal with the complex multi-modal noisy demonstrations introduced in BiGym. We believe this can motivate future research on finding appropriate policy representations for mobile bi-manual manipulation.

**Demo-driven RL approaches struggle with the complex task space and sparse reward in BiGym.** We observe that demo-driven RL algorithms fail on most of the BiGym tasks. For instance, CQN [49], which exhibits strong performance on fixed single-arm demo-driven RL setups, fails to solve most of the BiGym tasks. It is notable that all RL algorithms only achieve non-zero success rates on simple

Table 2: Success rates (%) of IL and demo-driven RL algorithms on 40 BiGym tasks, evaluated on 50 episodes. We report the results aggregated over the last three checkpoints.

Task	IL Algorithms			RL Algorithms			
	BC	ACT	DiffPolicy	DrQV2	AWAC	IQL	CQN
reach_target_single	66.0±0.0	<b>100.0±0.0</b>	61.3±5.8	<b>100.0±0.0</b>	94.0±2.0	82.0±5.3	92.7±1.2
reach_target_multi_modal	75.3±2.3	<b>98.7±1.2</b>	63.3±3.1	<b>100.0±0.0</b>	<b>100.0±0.0</b>	53.3±8.1	69.3±2.3
reach_target_dual	23.3±2.3	<b>90.7±1.2</b>	19.3±3.1	24.0±2.0	<b>77.3±6.1</b>	48.7±20.2	40.0±10.6
stack_blocks	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
move_plate	2.7±1.2	<b>30.0±3.5</b>	20.0±2.0	0.0±0.0	0.0±0.0	0.0±0.0	<b>0.7±1.2</b>
move_two_plates	7.3±2.3	11.3±7.0	<b>12.0±4.0</b>	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
flip_cup	0.0±0.0	<b>21.3±1.2</b>	6.0±2.0	0.0±0.0	0.0±0.0	<b>1.3±1.2</b>	0.0±0.0
flip_cutlery	0.7±1.2	<b>22.0±2.0</b>	1.3±1.2	0.0±0.0	0.7±1.2	<b>1.3±1.2</b>	<b>1.3±1.2</b>
dishwasher_open	6.0±5.3	<b>72.0±45.0</b>	4.0±4.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
dishwasher_close	84.7±20.0	<b>100.0±0.0</b>	99.3±1.2	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
dishwasher_open_trays	16.7±5.8	<b>100.0±0.0</b>	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
dishwasher_close_trays	0.0±0.0	<b>100.0±0.0</b>	52.0±18.3	<b>2.0±2.0</b>	0.0±0.0	0.0±0.0	0.0±0.0
dishwasher_load_plates	0.0±0.0	<b>34.0±8.7</b>	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
dishwasher_load_cups	0.0±0.0	<b>46.0±0.0</b>	8.7±5.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
dishwasher_load_cutlery	8.7±2.3	<b>42.0±8.7</b>	3.3±2.3	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
dishwasher_unload_plates	5.3±1.2	2.0±3.5	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
dishwasher_unload_cups	9.3±4.2	<b>15.3±10.1</b>	0.7±1.2	<b>0.7±1.2</b>	<b>0.7±1.2</b>	0.0±0.0	0.0±0.0
dishwasher_unload_cutlery	3.3±2.3	<b>18.0±3.5</b>	1.3±1.2	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
dishwasher_unload_plates_long	0.0±0.0	0.7±1.2	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
dishwasher_unload_cups_long	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
dishwasher_unload_cutlery_long	1.3±2.3	<b>14.7±8.3</b>	5.3±5.8	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
drawer_top_open	9.3±16.2	<b>100.0±0.0</b>	3.3±3.1	0.0±0.0	<b>8.7±15.0</b>	2.0±2.0	0.0±0.0
drawer_top_close	<b>100.0±0.0</b>	<b>100.0±0.0</b>	<b>100.0±0.0</b>	<b>100.0±0.0</b>	<b>100.0±0.0</b>	<b>100.0±0.0</b>	<b>100.0±0.0</b>
drawers_open_all	10.7±10.1	<b>100.0±0.0</b>	16.7±8.3	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
drawers_close_all	0.0±0.0	<b>100.0±0.0</b>	27.3±18.6	<b>100.0±0.0</b>	<b>100.0±0.0</b>	<b>100.0±0.0</b>	44.0±10.4
wall_cupboard_open	22.0±31.2	97.3±1.2	<b>100.0±0.0</b>	27.3±5.0	12.0±17.3	9.3±2.3	0.0±0.0
wall_cupboard_close	<b>100.0±0.0</b>	<b>100.0±0.0</b>	<b>100.0±0.0</b>	<b>100.0±0.0</b>	26.0±41.6	97.3±4.6	70.0±2.0
cupboards_open_all	5.3±4.2	<b>17.3±21.4</b>	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
cupboards_close_all	<b>63.3±7.0</b>	0.7±1.2	1.3±2.3	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
take_cups	0.0±0.0	<b>26.0±2.0</b>	5.3±2.3	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
put_cups	3.3±2.3	<b>30.0±7.2</b>	0.7±1.2	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
pick_box	20.7±1.2	<b>40.7±1.2</b>	22.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
store_box	8.7±3.1	<b>13.3±3.1</b>	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
saucepan_to_hob	21.3±4.6	<b>88.0±2.0</b>	34.7±3.1	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
store_kitchenware	0.0±0.0	2.7±2.3	0.7±1.2	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
sandwich_toast	5.3±1.2	<b>30.7±6.1</b>	10.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
sandwich_flip	0.0±0.0	<b>32.0±2.0</b>	4.7±1.2	<b>0.7±1.2</b>	0.0±0.0	0.0±0.0	0.0±0.0
sandwich_remove	40.7±8.3	<b>55.3±7.0</b>	48.0±0.0	<b>0.7±1.2</b>	0.0±0.0	0.0±0.0	0.0±0.0
store_groceries_lower	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
store_groceries_upper	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
Average	18.0±1.1	<b>46.3±1.4</b>	20.8±0.8	<b>13.9±0.2</b>	13.0±1.2	12.4±0.0	10.5±0.4

tasks with little interaction with the objects, e.g., `reach_target_single` and `top_drawer_close`, and completely fail to solve all the other tasks. We hypothesise this is because (i) the presence of mobile base makes it more difficult for agents to explore meaningful state. e.g. an erroneous base turning action can easily cause robot to lose view of the objects. and (ii) RL agents struggle to learn value-functions on long-horizon BiGym tasks with sparse reward.

## References for Appendix

- [50] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- [51] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [52] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [53] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

- 604 [54] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*,  
605 2016.
- 606 [55] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint*  
607 *arXiv:1412.6980*, 2014.
- 608 [56] Y. Lu, K. Hausman, Y. Chebotar, M. Yan, E. Jang, A. Herzog, T. Xiao, A. Irpan, M. Khansari,  
609 D. Kalashnikov, et al. Aw-opt: Learning robotic skills with imitation and reinforcement at scale.  
610 *arXiv preprint arXiv:2111.05424*, 2021.