JUICER: Data-Efficient Imitation Learning for Robotic Assembly
Supplementary Materials

Lars Ankile[1,3], Anthony Simeonov[2,3], Idan Shenfeld[2,3], Pulkit Agrawal[2,3]
[1]Harvard University, [2]Massachusetts Institute of Technology, [3]Improbable AI Lab

APPENDIX

A. Implementation Details

1) Training Hyperparameters: We present a comprehensive set of hyperparameters used for training. Hyperparameters shared for all models are shown in Table I, hyperparameters specific to the diffusion models in Table II, and hyperparameters specific to the MLP baseline in Table III.

TABLE I: Training Hyperparameters Shared for All Models

Parameter	Value
Control Mode	Position
Action Space Dimension	10
Observation Space Dimension	16
Orientation Representation	6D
Max LR	10^{-4}
LR Scheduler	Cosine
Weight Decay	10^{-6}
Warmup steps	500
Batch Size	256
Max Epochs	400
Steps per Epoch	400
Image Size Input	$2 \times 320 \times 240 \times 3$
Image Size Encoder	$2 \times 224 \times 224 \times 3$
Vision Encoder Model	ResNet18
Encoder Weights	ImageNet 1k
Encoder Parameters	2×11 million
Runs per Condition	5
Encoder Feature Projection Dim	128

TABLE II: Diffusion Model Hyperparameters

Parameter	Value
U-Net Down dims	[256, 512, 1024]
U-Net Parameters	69 million
Policy total parameters	91 million
Observation Horizon T_o	1
Prediction Horizon T_p	32
Action Horizon T_a	8
DDPM Training Steps	100
DDIM Inference Steps	8

TABLE III: MLP Baseline Hyperparameters

Parameter	Value
Residual Blocks	5
Residual Block Width	1024
Parameters	10 million
Policy total parameters	32 million
Observation Horizon T_o	1
Prediction Horizon T_p (Chunked)	32
Action Horizon T_a (Chunked)	8
Prediction Horizon T_p (No Chunking)	1
Action Horizon T_a (No Chunking)	1

2) Normalization: All 10 and 16 dimensions of the action and proprioceptive state, respectively, were independently normalized to lie in the range [-1, 1]. The normalization limits were calculated across all the demonstration data across all 4 tasks to ensure consistent action and state spaces across tasks. This follows the normalization used in, e.g., [1], [2] and is also the generally accepted normalization used for diffusion models. [1] standardized the input to have mean 0 and standard deviation 1 instead of min-max scaling to [0,1], which is something we did not test in our experiments.

3) Rotation Representation: We represent all orientations and rotations with the 6D representation, both for the predicted action and proprioceptive end-effector pose orientation [3], [4]. The end-effector rotation angular velocity was still encoded as roll, pitch, and yaw values. This representation of rotations contains redundant dimensions but is continuous in that small changes in orientation also lead to small changes in the values in the representation, which is not generally the case for Euler angles and quaternions, which can enable easier learning.

4) Image Augmentation: We apply image augmentation to both images during training. We apply random cropping only to the front camera view. In addition, we apply color jitter with hue, contrast, brightness, and saturation set to 0.3 and Gaussian blur with a kernel of size 5 and sigma between 0.1 and 5 to both camera views.

At inference time, we statically center-crop the front camera image from 320×240 to 224×224 and resize the wrist camera view with the same dimensions. For both the random crop and center crop, we resize the image to 280×240 to ensure we are not moving the image around so much that essential parts of the scene are cropped out.

The specific choice of the above values was made by eye to find a level that was sufficiently adversarial while still keeping all essential features discernible. We show examples of augmentations below.

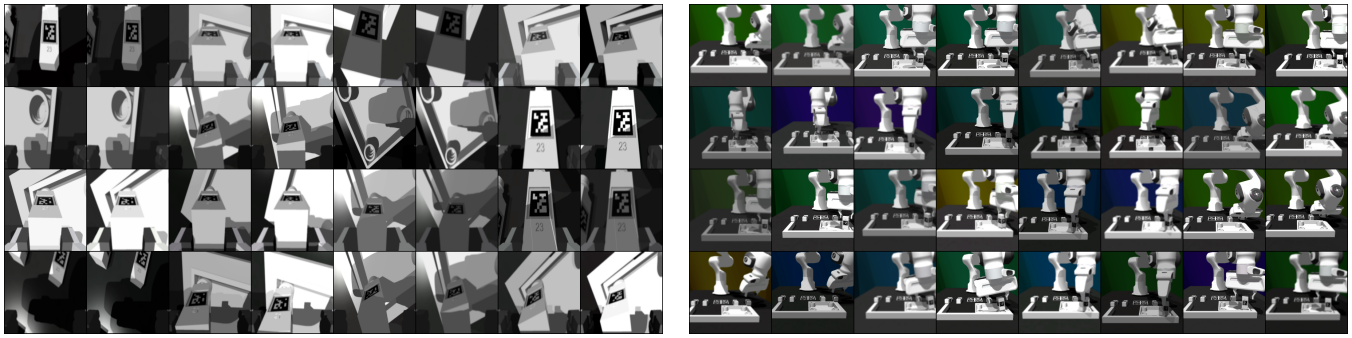


Fig. 1: **Left:** Examples of augmentations of the wrist camera view, consisting of color jitter and Gaussian blur. **Right:** Examples of augmentations for the front view also consist of color jitter and Gaussian blur augmentations, as well as random cropping.

B. Data Collection, Training, and Evaluation Details

The full pipeline for training the final models with JUICER involved several steps and different code files. In the below list, we give a brief summary of the steps on a high level and what files the relevant code and command-line arguments are found in. Please follow the installation instructions in the code repository (github.com/ankile/imitation-juicer) to ensure all required packages are installed.

1) Teleoperation Demonstration and Annotation Effort: In Table IV, we present the approximate time the teleoperator spent collecting and annotating 50 demos for each of the 4 tasks in this work. The labeling time includes all time spent on the labeling task, i.e., all idle time resulting from waiting for files to load and write and any mistakes that were undone and redone. Particularly, the loading of the data took a meaningful amount of time, and the whole process could be made significantly faster by optimizing the read-write speeds (which we did not do).

TABLE IV: Approximate Demo Collection and Annotation Times in Minutes

Task	Critical States	Collect 50 Demos (min)	Annotate 50 Demos (min)
`one_leg`	1	∼70	∼10
`round_table`	3	∼85	∼20
`lamp`	5	∼95	∼30
`square_table`	4	∼210	∼30

2) Simulator Teleoperation Assistive Improvements: One crucial difficulty with collecting data in a simulator is the lack of depth perception. This impediment is, on the surface, very limiting. However, two things helped alleviate the difficulties. First is our experience that the brain is quite adaptable and very readily learned to rely on cues other than depth to infer the relative positions of objects in the depth direction (presumably shadows and relative sizes). Second, we added a "laser

light" that protrudes out from the end-effector that more explicitly indicates the position and orientation of the end-effector in the scene, as shown in Figure 2.

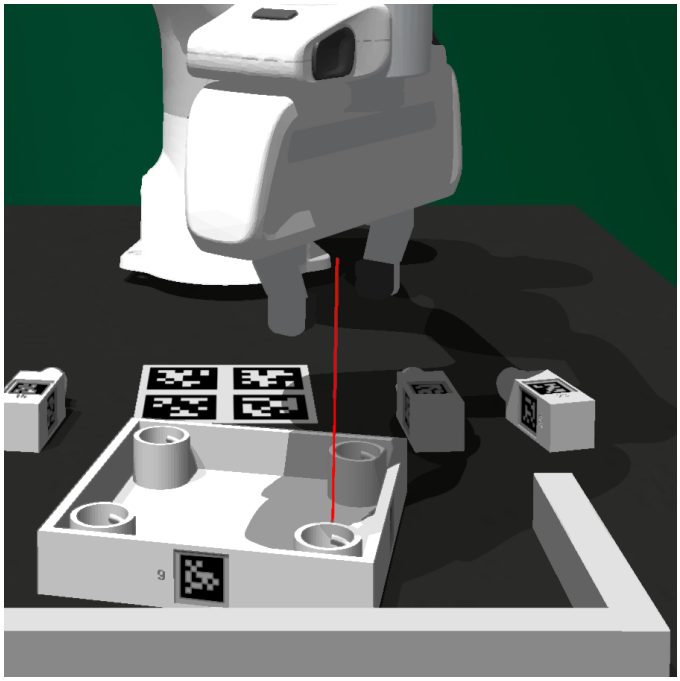


Fig. 2: The red light is added as a visual aid for the teleoperator to help alleviate the difficulties introduced by the lack of depth perception. The red line is only visible to the operator and is not rendered to the image observations that are stored while teleoperating. In the image, in the absence of the line, it is quite hard to tell if the end-effector is placed right above the tabletop or not. With the red line, however, it becomes much more obvious.

3) Dataset Sizes Across Tasks and Methods: This section details more about what mix of data from the different sources the different models across normal behavior cloning (BC), trajectory argumentation (TA), Collect-Infer (CI), and a mix of all of them for the models for which we report results in the main results figure and Table VI.

TABLE V: Dataset Size for Task and Method

		Teleop		Rollout		Augmentation		
Task	Method	Demos	Timesteps	Demos	Timesteps	Demos	Timesteps	Total Timesteps
one_leg	BC	50	29k	–	–	–	–	29k
	Traj. Aug	50	29k	–	–	50	3k	32k
	Col. Inf.	50	29k	50	27k	–	–	56k
	TA + CI	50	29k	50	27k	150	8k	64k
round_table	BC	50	47k	–	–	–	–	47k
	Traj. Aug	50	47k	–	–	500	12k	59k
	Col. Inf.	50	47k	50	55k	–	–	102k
	TA + CI	50	47k	50	55k	700	17k	119k
lamp	BC	50	42k	–	–	–	–	42k
	Traj. Aug	50	42k	–	–	100	5k	47k
	Col. Inf.	50	42k	50	45k	–	–	87k
	TA + CI	50	42k	50	45k	200	10k	97k
square_table	BC	50	130k	–	–	–	–	130k
	Traj. Aug	50	130k	–	–	150	8k	138k
	Col. Inf.	50	130k	50	126k	–	–	256k
	TA + CI	50	130k	50	126k	400	22k	278k

C. Results in Tabular Form

In Table VI, we present the same results as in the main results table, but in tabular form with the exact numbers.

TABLE VI: Main Results Table

Task	One leg		Round table		Lamp		Square table	
Method	Avg	Max	Avg	Max	Avg	Max	Avg	Max
MLP-NC	0	0	0	0	0	0	0	0
MLP-C	40	52	9	15	3	4	4	9
DP-BC	59	68	18	21	6	7	6	9
State noise [5]	64	70	12	19	6	11	13	16
Traj. Aug.	64	65	28	31	8	12	9	17
Col.-Inf.	**75**	79	24	27	18	**29**	12	**20**
TA & CI	**74**	**85**	**32**	**35**	**28**	**30**	**15**	17
Multi-task	58	67	25	**34**	12	18	7	10

D. Further Analysis

1) Data Efficiency: In Figure 3, we see how the performance for the `one_leg` task changes with increasing demo dataset size. We calculate success rates for policies trained with [10, 20, 30, 40, 50, 100, 200, 300, 500, 1000] demos. We find the success rate to follow an exponential curve closely up to 1000 demonstrations but seems to saturate at around ∼300-500 demonstration trajectories in the training data.

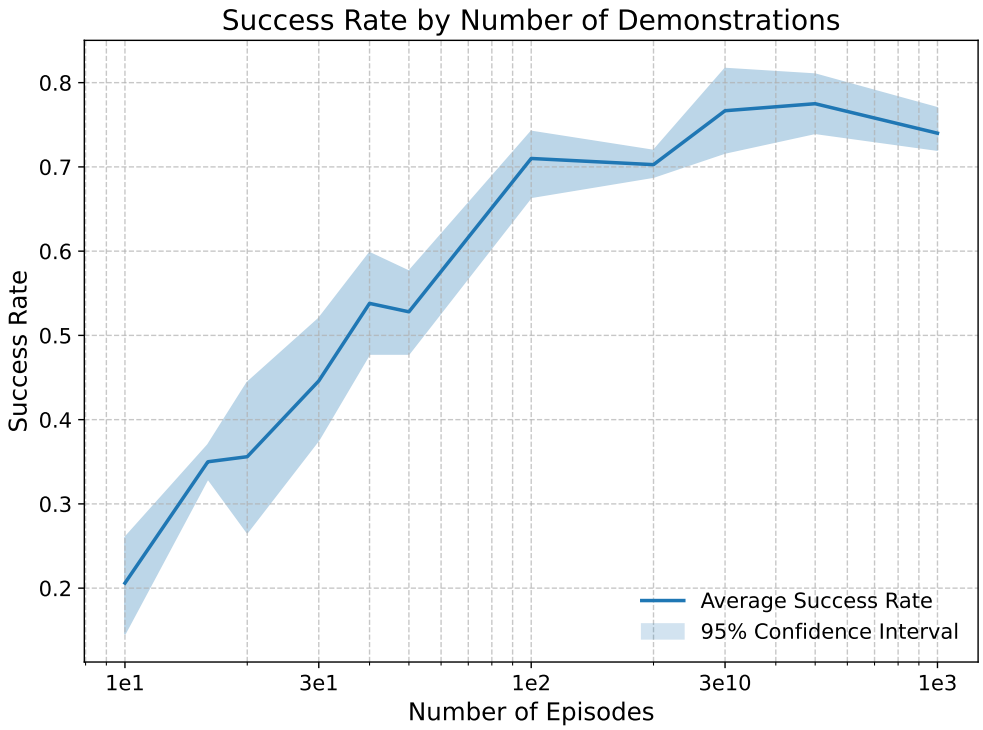
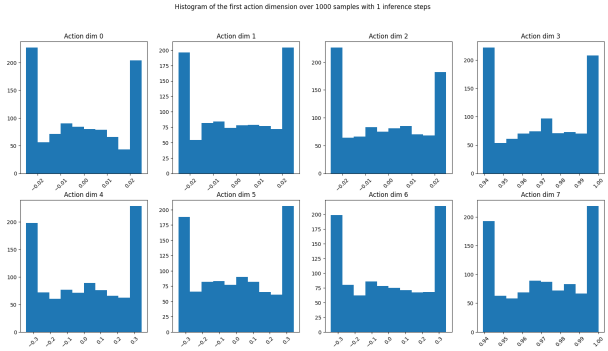


Fig. 3: Data efficiency graph for the ‘one_leg‘ task.

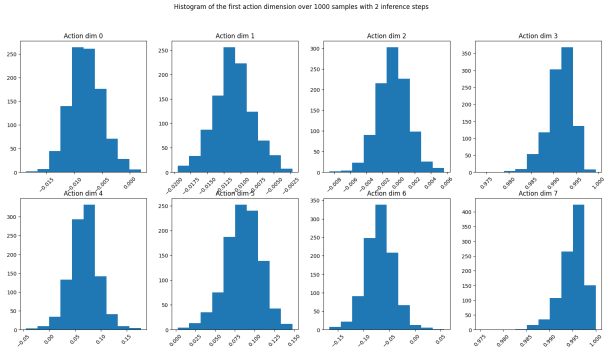
2) Number of Denoising Steps in DDIM Sampler: In Figure 4, we present histograms over samples of actions for a given state observation for different numbers of DDIM inference sampling steps of 1, 2, 4, and 100. In particular, we chose an arbitrary state during the `one_leg` task (the robot was grasping a leg and close to the point of insertion in this example). We sampled 1000 random Gaussian noise vectors that we fully denoised using the DDIM sampler with the given number of denoising steps.

In each of the four figures in Figure 4, we show the distribution as a histogram for each action independently, here as delta actions for the 3D position and rotation as roll, pitch, and yaw. The 8th action distribution is the gripper action. These representations are used in this analysis because it is easier to intuitively interpret than the absolute position actions and 6D rotation representations.

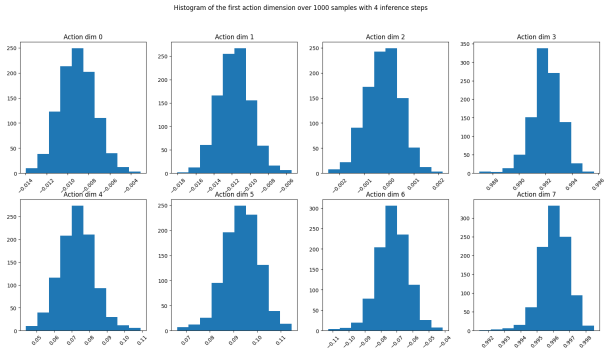
We observe that the distributions converge quickly to the “final” distribution with the number of denoising steps. After 4 steps, it is already hard to distinguish the resulting distribution from the one using 100 steps.



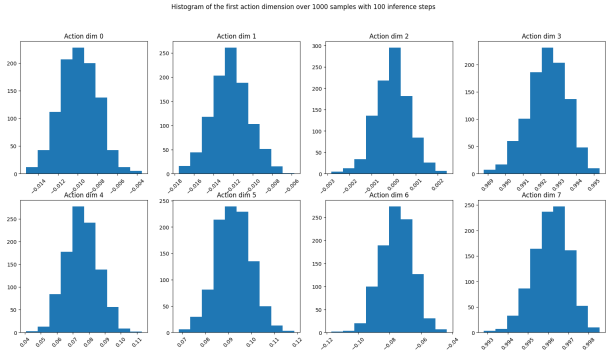
(a) Action prediction using 1 DDIM sampling step.



(b) Action prediction using 2 DDIM sampling steps. The samples are already getting relatively good.



(c) Action prediction using 4 DDIM sampling steps. The samples are very hard to distinguish from the predictions using 100 DDIM steps.



(d) Action prediction using 100 DDIM sampling steps.

Fig. 4: Comparison of samples of predicted actions for the same state with a varying number of DDIM sampling steps. The four plots show the resulting prediction sample histograms for 1000 action samples with 1, 2, 4, and 100 DDIM sampling steps, respectively. Each of the 8 histograms in each of the 4 subplots shows the distribution for a specific dimension of the action space: $[\Delta x, \Delta y, \Delta z, \Delta\text{roll}, \Delta\text{pitch}, \Delta\text{yaw}, \text{gripper-width}]$. Increasing the sampling steps from 1 to 2 produces a vast difference in the distribution, but further increases do not change the predicted action distributions much.

3) Ratio of Synthetic to Demonstration Data: The optimal number of synthetic trajectory snippets was not a question of more being necessarily better, and we experimented with different ratios. Across our tasks, we typically got the best results with the augmentation timesteps constituting somewhere between 8-20% of the total number of timesteps in the training data (see Table V).

4) Importance of Action-Chunking: In our experiments with implementing the baseline MLP model, we see a drastic difference in performance between the same architecture trained to predict the next action $T_p = 1$ versus ones trained to predict a chunk $T_p = 32$. In particular, not a single full task completion was observed for any rollout for any of the 5 models trained for any of the 4 tasks. Qualitatively, when observing the rollouts, the behavior is also a lot more erratic and less smooth with a prediction horizon of 1.

5) Comparison of Different Vision Encoders: We found a standard ResNet [6] from the `torchvision` package with the `IMAGENET1k` [7] pretraining weights to work better than both the same ResNet with no pretraining and the ResNet with the spatial softmax pooling layer from `Robomimic` [8] that was used in [2]. We also see very similar performance when using a Vision Transformer [9] model ViT-Small with Dino V1 weights [10], a ViT-Base with pertaining weights from pertaining on ImageNet with the MAE [11] objective and a ResNet50 with weights from pertaining on the Ego4D [12] dataset with the VIP [13] objective. The best-performing model, though, was a ResNet18 with pertaining weights from the R3M [14] objective on the Ego4D dataset, achieving an average success rate of 77% for the `one_leg` task versus 59% for the ResNet18 pretrained on ImageNet. Surprisingly, the R3M models using the ResNet 34 and 50 performed slightly worse than the ResNet18.

Results for the `one_leg` tasks for different vision encoder architectures and pretrained weights are summarized in Table VII.

TABLE VII: Comparison of Vision Encoder Success Rates on the `one_leg` task

Model	Training Objective	Supervision	Pretraining Dataset	Success Rate
ResNet18 [6]	Classification	Supervised	ImageNet [7]	59%
ResNet18	No Pretraining	—	—	49%
ResNet18 (Spatial Softmax) [15]	No Pretraining	—	—	30%
ViT-Small [9]	DINO [10]	Self-supervised	ImageNet	55%
ViT-Base	MAE [11]	Self-supervised	ImageNet	63%
ResNet18	**R3M [14]**	**Self-supervised**	**Ego4D [12]**	**77%**
ResNet34	R3M	Self-supervised	Ego4D	75%
ResNet50	R3M	Self-supervised	Ego4D	73%
ResNet50	VIP [13]	Self-supervised	Ego4D	54%

6) Empirical Mode Sampling: During development, we observed that the model can sample actions in low-density parts of the distribution, which could be actions that take the agent out of the distribution, causing a failure. One interesting finding is that as soon as the model was OOD, it tended to widen its action prediction distributions by approximately 10 times.

In experiments with forcing the model to sample only actions near the mode, we implemented a simple Kernel Density Estimation method for estimating the mode of an empirical distribution over a sample of continuous actions. In our experience, this did not improve success rates for the preliminary experiments on `one_leg`.

These experiments were carried out before reading the results of [16]. We hypothesize that one of the reasons we did not find KDE mode sampling to help in our case is that our joint action-chunk space is of $|a| \cdot T_a = 10 \cdot 8 = 80$ dimensions, which can be a very sparsely populated space, even with 5000 samples. This is subject to further investigations in future work.

REFERENCES

[1] M. Reuss, M. Li, X. Jia, and R. Lioutikov, "Goal-Conditioned Imitation Learning using Score-based Diffusion Policies," June 2023, arXiv:2304.02532 [cs]. [Online]. Available: http://arxiv.org/abs/2304.02532

[2] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," June 2023, arXiv:2303.04137 [cs]. [Online]. Available: http://arxiv.org/abs/2303.04137

[3] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5745–5753.

[4] J. Levinson, C. Esteves, K. Chen, N. Snavely, A. Kanazawa, A. Rostamizadeh, and A. Makadia, "An analysis of svd for deep rotation estimation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 554–22 565, 2020.

[5] L. Ke, J. Wang, T. Bhattacharjee, B. Boots, and S. Srinivasa, "Grasping with Chopsticks: Combating Covariate Shift in Model-free Imitation Learning for Fine Manipulation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE, May 2021, pp. 6185–6191. [Online]. Available: https://ieeexplore.ieee.org/document/9561662/

[6] 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*, 2016, pp. 770–778.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, 2009.

[8] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," in *arXiv preprint arXiv:2108.03298*, 2021.

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[10] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," 2021.

[11] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," 2021.

[12] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, M. Martin, T. Nagarajan, I. Radosavovic, S. K. Ramakrishnan, F. Ryan, J. Sharma, M. Wray, M. Xu, E. Z. Xu, C. Zhao, S. Bansal, D. Batra, V. Cartillier, S. Crane, T. Do, M. Doulaty, A. Erapalli, C. Feichtenhofer, A. Fragomeni, Q. Fu, A. Gebreselasie, C. Gonzalez, J. Hillis, X. Huang, Y. Huang, W. Jia, W. Khoo, J. Kolar, S. Kottur, A. Kumar, F. Landini, C. Li, Y. Li, Z. Li, K. Mangalam, R. Modhugu, J. Munro, T. Murrell, T. Nishiyasu, W. Price, P. R. Puentes, M. Ramazanova, L. Sari, K. Somasundaram, A. Southerland, Y. Sugano, R. Tao, M. Vo, Y. Wang, X. Wu, T. Yagi, Z. Zhao, Y. Zhu, P. Arbelaez, D. Crandall, D. Damen, G. M. Farinella, C. Fuegen, B. Ghanem, V. K. Ithapu, C. V. Jawahar, H. Joo, K. Kitani, H. Li, R. Newcombe, A. Oliva, H. S. Park, J. M. Rehg, Y. Sato, J. Shi, M. Z. Shou, A. Torralba, L. Torresani, M. Yan, and J. Malik, "Ego4d: Around the world in 3,000 hours of egocentric video," 2022.

[13] Y. J. Ma, S. Sodhani, D. Jayaraman, O. Bastani, V. Kumar, and A. Zhang, "VIP: Towards Universal Visual Reward and Representation via Value-Implicit Pre-Training," Mar. 2023, arXiv:2210.00030 [cs]. [Online]. Available: http://arxiv.org/abs/2210.00030

[14] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3M: A Universal Visual Representation for Robot Manipulation," Nov. 2022, arXiv:2203.12601 [cs]. [Online]. Available: http://arxiv.org/abs/2203.12601

[15] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 512–519.

[16] T. Pearce, T. Rashid, A. Kanervisto, D. Bignell, M. Sun, R. Georgescu, S. V. Macua, S. Z. Tan, I. Momennejad, K. Hofmann, and S. Devlin, "Imitating Human Behaviour with Diffusion Models," Mar. 2023, arXiv:2301.10677 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2301.10677