# SLOT DIFFUSION POLICY: OBJECT CENTRIC REPRESENTATION FOR MULTI-TASK IMITATION LEARNING

**Chahyon Ku, Miles Priebe, Ryan Diaz, Adam Imdieke**
Department of Computer Science
University of Minnesota
Minneapolis, MN 55414, USA
{ku000045,prieb038,diaz0329,imdie022}@umn.edu

## ABSTRACT

We present Slot Diffusion Policy – an approach for performing data efficient multi-task imitation learning from object-centric representations. While most existing imitation learning methods operate on scene-level representations (pixels and voxels of the scene), we propose learning policies on object-level representations produced by slot attention Locatello et al. (2020). By explicitly breaking down the scene into multiple objects and learning tasks from them, we are better able to recognize key landmarks present in manipulation tasks (reference objects, goal locations) and quickly learn a policy. We conduct experiments in popular simulated (RLBench James et al. (2020)) and real world (Push-T Chi et al. (2023)) tasks to provide insight into how slot attention can be used for imitation learning. Our code is available in GitHub.

## 1 INTRODUCTION

How can we teach robots to dexterously handle objects found in our everyday lives? The ability to perform various manipulation tasks in unstructured environments is among the fundamental goals of robot learning. One method of developing a policy for robotic manipulation is imitation learning, or "learning from demonstration", which gained popularity due to its simplicity and success on a variety of tasks. In short, imitation learning involves a "learner" (such as the robot) learning its policy through observing demonstrations by an "expert" (such as a human) who is assumed to already behave according to the optimal policy. This technique has proven very useful when trying to teach robotic agents complex and multimodal task for which a simple policy cannot be designed.

Most imitation learning methods for robotic manipulation operate on scene-level representations, producing actions directly from pixels or voxels. This requires a visual encoder backbone to identify each relevant object (red block, plate, cup) in the environment and its significance with regards to the specific task (pick, press, sweep) from demonstrations without any specific inductive bias. Furthermore, these scene encoders are often adapted to multitask settings by conditioning the scene representations on language instructions (ex. "pick up the red cup") through channel-wise linear transformation of image encodings (FiLM Perez et al. (2018)) or cross attention between language tokens and visual tokens (PerAct Shridhar et al. (2022)). Since our tasks focus on interacting with specific objects in the environment, is there a more meaningful way to extract information about these objects from observation?

Thus, we explore if it is possible to build an imitation learning framework which learns on object-centric representations rather than scene-level ones. To this end, we propose SlotDiffusionPolicy, combining the unsupervised object-centric representation learning capabilities of the slot attention mechanism with the powerful multi-modal policy learning capabilities of the diffusion policy formulation. We compare the data requirements, compute requirements, and task success rates of our method with state of the art methods on the RLBench framework. We perform real world experiments with the Push-T task to test if our method transfers well to real robots.

## 2 Related Works

### 2.1 Slot Attention

Slot attention is a mechanism that builds upon the attention module for learning unsupervised object-centric representations for object discovery and object property prediction, initially proposed by Locatello et al. (2020). Jiang et al. (2023) extended this framework by learning object-centric representations through diffusion for the purpose of image reconstruction. This is closely aligned with what we intend to implement for SlotDiffusionPolicy, though we aim to use these object-centric representations for reconstructing robot trajectories and object interactions rather than just images. In terms of training, Heravi et al. (2023) developed a slot attention module that can be frozen for downstream tasks such as object and end-effector localization and multi-object goal conditioned policy learning. In SlotDiffusionPolicy, we aim to train the slot attention module through end-to-end policy learning combined with the diffusion policy formulation.

### 2.2 Imitation Learning

Research in imitation learning applications to robotic manipulation has cultivated a vast ecosystem of methods, data, and performance benchmarks. For training and simulating tasks, James et al. (2020) developed RLBench, a comprehensive simulation environment for benchmarking tasks related to robotic perception and manipulation, complete with a wide array of simple and complex tasks. Using the RLBench framework, Shridhar et al. (2022) proposed PerAct, a method to perform language-conditioned imitation learning by unifying language instruction and voxel inputs with a transformer model to produce actions. We aim to use a diffusion policy to predict actions rather than an explicit policy. Building on this unified system of language-conditioned imitation learning, Gervet et al. (2023) used a 3D feature field scene representation of arbitrary resolution with features lifted from 2D multi-view observations. Goyal et al. (2023) addressed the scalability problem of processing voxel scene representations by utilizing a transformer network that acts on 2D multi-view observations of a 3D point cloud reconstruction of a scene. Rather than using these scene-level representations, our SlotDiffusionPolicy method aims to learn an object-centric representation of the scene through slot attention.

As a new formulation of robotic manipulation policies, Chi et al. (2023) proposed DiffusionPolicy, a method of using diffusion for visuomotor policy learning by denoising noisy actions, which showed superior performance on multimodal tasks such as Robomimic tasks proposed by Mandlekar et al. (2021). We will focus on this diffusion formulation to construct our policy. Ha et al. (2023) built upon the single-task diffusion policy formulation by integrating language-conditioning to enable multi-task capabilities, using an LLM for large-scale language-guided demonstration data collection. We may use these capabilities to enable language-conditioned multi-task learning for our method.

## 3 Method

We propose to use a slot-based attention mechanism in conjunction with a diffusion-based policy network to learn a visuomotor policy using an imitation learning setup. Specifically, we tackle multi-object 2D tabletop pushing/sliding tasks, conditioned on language instructions of multiple goals. By combining these two components, we hope to achieve data and compute efficient training of imitation learning agents.

**Real World Experiments.** We collected 80 demonstrations on the real-world UR5e robot for the Push-T task Chi et al. (2023), using a joystick teleoperation setup. This task requires the robot to use a peg end effector to push a 3D-printed T into the outlined goal marker and return to a home marker. The collected data contains multiple camera views and proprioceptive input for each sample. We show the top-down, side, and wrist camera views in Figure 1. The proprioceptive data consists of the 6D end-effector pose.

Table 1 shows the comparison between the real world push-t dataset we generated and the dataset provided from the original work. The image shapes reported are consistent between the two datasets and are downsampled to the same resolution. 2 views were used in the original work (wrist and front), compared to the 3 views we recorded in our dataset (wirst, side, and top-down). The action

horizon is defined as the number of actions the robot executes upon prediction. We keep the horizon consistent between the datasets, only changing from planar XY actions (2D), to XYZRPY actions (6D). This change was made due to mounting differences between our real world UR5 robot the original work's. Because we constrain all axis besides YZ in training and evaluation, this has no impact on performance. Compared to the original work, we decrease the number of demonstrations by over 50% due to significant data collection time investment. We can attribute the slight increase in demonstration length to non-intuitive teleoperation and higher difficulty of initial positions. The recording frequency of our observations from the D405 cameras remains consistent with the original work. Finally, we note that our dataset requires less storage due to the decreased number of demonstrations.

| | Diffusion Policy (Chi et al. (2023)) | Slot Diffusion Policy |
|---|---|---|
| Image shapes | 240x320 | 240x320 |
| Number of views | 2 | 3 |
| Action horizon | 8x2 | 8x6 |
| Number of Demos | 136 | 50 |
| Mean Demo Length (s) | 20.34 | 27.64 |
| Frames per Second | 10 | 10 |
| Total Size (GB) | 2.4 | 1.0 |

Table 1: Comparison of task/dataset between our work and original work.

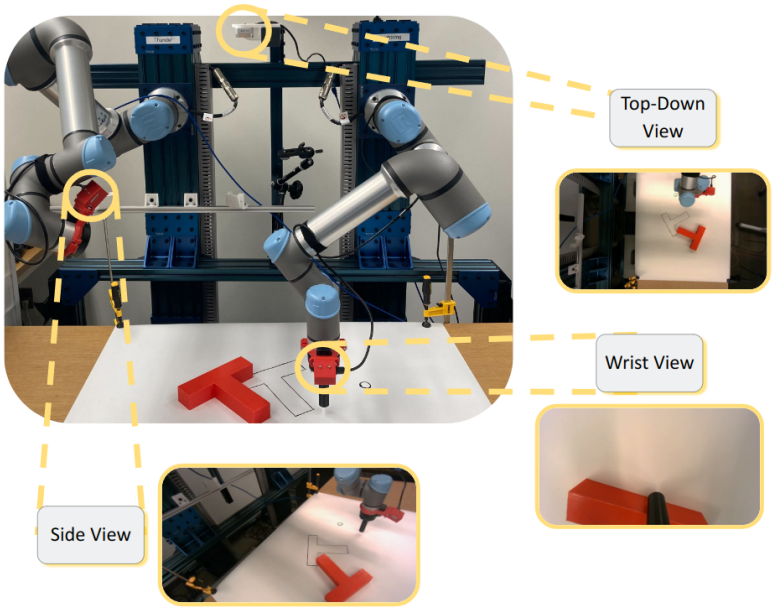


Figure 1: Real world single-arm setup with Push T task environment, along with top-down view, wrist view, and side view fed into the policy as observations.

**Slot Attention.** We can first train our Slot Attention module on frames obtained from the side camera view, allowing for the least occlusion of the 3D printed T, the object we wish to learn a representation of.

To create an object-aware representation, typically a CNN is used to extract visual features from images, followed by iterations of Slot Attention (Locatello et al. (2020)). The Slot Attention mechanism works similarly to cross attention, where the keys and values are generated from the input features, and the queries are generated from a learned set of weights. After the query-key multiplication, a SoftMax operation is applied to the result across the slot dimension. Each slot attends

to every input token, so the SoftMax should cause one slot to dominate the query-key score. The output of the SoftMax is used to do a weighted average of the values for the GRU update of the slots.

After $T$ iterations of Slot Attention, a L2 reconstruction loss is used to produce gradients. Each slot works to add to the reconstruction of the image as slots are separately reconstructed through deconvolutional layers. The output of each reconstructed slot contains three color channels, with a mask channel. The mask channel is normalized across the slots and used as a weight for the slot's contribution to that pixel.

Slot Attention theoretically offers benefits when used as a visual encoder for our policy. First, the competition between slots causes each slot to dominate the reconstruction of a particular aspect of the image. When Slot Attention is trained on a dataset of images with moving objects, each slot should describe an object. With each slot describing an object, it is hypothesized that the policy learns the interactions between these objects better.

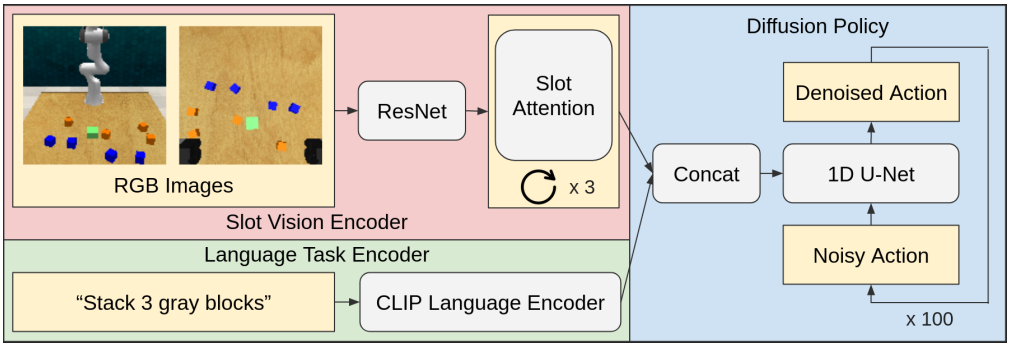


Figure 2: Architecture diagram for Slot Diffusion. The observations are first passed through a ResNet, and then the features have slot attention applied. In this project the language encoder is not used, and the U-Net is conditioned only on the image features.

**Policy Training.** For the behavior cloning portion of this work, we will build upon Chi et al. (2023). That work implements a DDPM (Denoising Diffusion Probabilistic Model) on 6 DoF end effector trajectory. The model starts with a sample of Gaussian noise for each point in the trajectory, and Denoises the points conditioned on the robot's current position and images, following a set noise schedule. We can then replace the visual encoders (ResNet-18) with Slot Attention modules.

Diffusion Policy handles multi-modal trajectory predictions well, along with scaling to high dimensional data (Chi et al. (2023)). Assuming $D$ dimensional actions, the forward process takes a ground truth trajectory and adds noise according to the noise schedule. The reverse policy predicts the noise in $\mu$ for each denoising step, starting with $D \times Horizon$ dimensional Gaussian noise. The reverse denoising steps are conditioned on the image embeddings and proprioceptive information. Our implementation uses the previous two observations and predicts the next 16 steps of actions, executing only the first 8. In this project, we will evaluate the feasibility of using Slot Attention based image encoders and compare to the standard Resnet-18 trained from scratch.

**Evaluation.** We will evaluate our network on Reconstruction Mean Squared Error (MSE), Task Success Rate, and Task Completion Time. We will use these metrics as these are what the original paper used to compare training methods for the ResNet18 (ImgNet, R3M, From Scratch).

We will consider a policy rollout a success if the T is at a higher IoU than the lowest seen in training, as the authors of Diffusion Policy (Chi et al. (2023)) did. Our training data had a minimum IoU of about 80% so we will use that for the cutoff. This value was determined through visual inspection and was not numerically calculated through segmentation as in the original work. The time will stop when the policy returns home, or after 60 seconds, so it is possible that a success is achieved, but the policy runs out of time.

## 4 RESULTS

**Slot attention for reconstruction.** Before jointly training a policy with slot attention, we first conduct experiments to determine the best hyperparameters for slot attention. To specify, we compare the backbones (5-layer CNN from Slot Attention (Locatello et al. (2020)) vs. ResNet-34 from Invariant Slot Attention (Biza et al. (2023))), the tasks from which the training data is collected (1 vs. 5 vs. 18), the views (front vs. front, wrist, top, 2 shoulders), and the number of slots (3 vs. 5 vs. 8). Our results show that slot attention benefits from training on multiple tasks, likely due to more visual variation, but fails to generalize to multiple views, as these have drastically different scale and backgrounds. We observe best performance with a ResNet-34 backbone, 18 tasks, 1 view, and 5 slots when evaluated on unseen frames from the close jar task.

| | Train MSE | Val MSE |
|---|---|---|
| 1-task 1-view (Random Slots) | 0.0011 | 0.0045 |
| 1-task 1-view | **0.0010** | 0.0044 |
| 5-task 1-view | 0.0016 | 0.0028 |
| 1-task 5-view | 0.0021 | 0.0049 |
| 5-task 5-view | 0.0033 | 0.0039 |
| 18-task 1-view | 0.0019 | **0.0018** |
| 18-task 1-view (8 Slots) | 0.0018 | 0.0023 |

Table 2: Reconstruction quality of slot attention on RLBench data. All models trained on specified task/view until convergence of validation loss on front-view close jar images with 5 learned slots unless indicated otherwise.

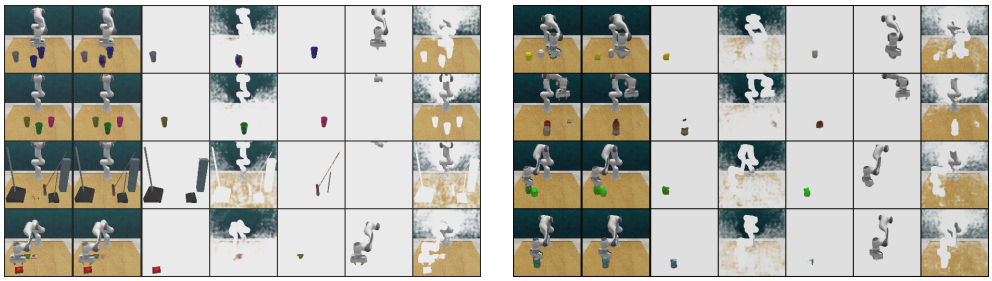

Figure 3: **Left:** Sampled training images from 18 tasks in RLBench. **Right:** Sampled validation images for our target task (Close Jar). **Columns:** From the left, ground truth image, combined reconstructed image, and reconstructed images from each of the 5 slots.

Figure 4: Sampled validation images of the trained slot attention the real-world data. **Columns:** From the left, ground truth image, combined reconstructed image, and reconstructed images from each of the 5 slots.

**Simulation policy.** The policy training results in the simulation setup was largely unsuccessful, resulting in 0% success rates for both slot and non-slot models. Upon close inspection of the failure modes, we observe behaviors suggesting the model lacks the precision required to perform manipulation tasks. To specify, we observe that the policy fails to reach the jar cap and repeatedly

make attempts to grasp the jar. While it is unclear exactly what is contributing to this failure, we have ruled out a few differences between robomimic (simulation environment originally used for evaluating diffusion policy) and RLBench (a multitask simulation environment). First, the rotation representations were diferent, with robomimic using the first two columns of the rotation representation Hempel et al. (2022) and RLBench using quaternions, so we tried manually converting the regression targets to ortho6d, but observed no improvement. Second, the distribution of motion is different, with the jittery and slow movement of robomimic's human demonstrators vs. the linear and fast movement of RLBench's scripted demonstrations, so we tried capping the magnitude of the movement but observed no clear improvement. In conclusion, we hypothesize that there is a critical hyperparameter that needs to be tuned to move foward with evaluating policies on the RLBench simulation environment.

| | Recons. MSE | Success Rate | Mean Time |
|---|---|---|---|
| PerAct (Shridhar et al. (2022)) | - | 60.00 % | - |
| Act3D (Gervet et al. (2023)) | - | 92.00 & | - |
| Image Diffusion Policy | - | 0.00 % | - |
| Slot Diffusion Policy | 0.0018 | 0.00 % | - |

Table 3: Simulation results on the close jar task of RLBench. Trained on 100 demonstrations and evaluated on 20 random initialization.

**Real world policy.** On the real setup, the performance of our Diffusion Policy is worse than that of the original paper's. We attribute the fewer number of training epochs (200 vs. 600) and less amount of data (80 demos vs. 136 demos) to the degradation of performance. Additionally, the hyperparameters for policy training such as learning rate, data augmentation (random cropping), and preprocessing (imagenet normalization) may have been optimized for their real world setup.

| | Recons. MSE | Success Rate | Mean Time |
|---|---|---|---|
| Diffusion Policy (Chi et al. (2023)) | - | 95.00 % | 22.90 s |
| Diffusion Policy (ours) | - | 65.00 % | 51.62 s |
| Slot Diffusion Policy | 0.0007 | 50.00 % | 46.49 s |

Table 4: Real world results on the push-t task. Trained on 50 demonstrations and evaluated on 20 random initializations.

Contrary to our original hypothesis, we observe that Slot Diffusion Policy performs worse than vanilla diffusion policy. To specify, we observe a 15% decrease in success rate when we evaluate on 20 random initailizations. We have two theories as to why this is the case: shorter training run caused by the extra amount of compute (200 epochs for non-slot vs. 100 epochs for slot) and the smaller image resolution fed into the slot policy (320x240 for non-slot vs. 192x128 for slot).

## 5 LIMITATIONS

**Indecisive behavior.** While the multimodal nature of diffusion policy excels at performing complicated tasks and recovering from local failures, we observed indecisive behavior where the policy fails to make progress and jitters in place. This is likely due to there being a local minima in which the trajectory gets stuck at. This was expecially evident when the policy was presented with a multimodal setup, in which there were two equally likely solutions to solve the problem (the gripper is in the exact center of the T and attempts to move to the other side). Through the evaluation process, it was unclear how these behaviors were learned and if the data or the training was to blame. We hypothesize it is a combination of low quality demonstrations, inefficient task design, and a sub-optimal quantity of demonstrations.

**Sensitivity to change in environment.** It was observed through real world experiments that the trained policy is relatively robust to the variation in task (location and orientation of the T), but extremely sensitive to any external variations (lighting, position of table, exposure of camera, the goal location).

# 6 CHALLENGES

## 6.1 SIMULATION

Although simulation provided an ideal environment for rolling out and evaluating our policy, a non-trivial amount of effort had to be devoted to getting it up and running.

**Adaptation of data.** The off-the-shelf codebase for diffusion policy had no native way for taking in demonstration data from RLBench in its original format. In order to evaluate our SlotDiffusion-Policy setup on RLBench data, we first had to ensure that the diffusion policy was able to run using this data. This involved implementing a dataloader that was compatible with the inputs required by the policy. Although there was a cursory overview of the codebase that the diffusion policy runs off of, there was little to no documentation present in the actual code. Thus, significant amounts of time had to be invested combing through much of the code and piecing together the components needed to be able to convert the third-party RLBench data into a format usable by the diffusion policy.

**Adaptation of environment.** Like with the data, the diffusion policy repository did not come with a simulation environment that was able to simulate the RLBench task setup. Thus, we had to implement the connection between the RLBench simulation environment and the diffusion policy ourselves. This implementation step also suffered from the lack of documentation present in the code, and the functioning of the evaluation and policy rollout code in the simulation was a bottleneck for debugging the dataloading and training code, and ultimately running our simulation experiments.

## 6.2 REAL WORLD

Task design and data generation each presented us a unique sets of challenges in the real world environment.

**Physical task setup.** The real world data generation process relies on an effective physical task setup. For the Push T task, this consists of precise placement of the cameras, goal location, and start location, to include the least amount of self-occluded views in the dataset. Given our task setup, the top-down camera consistently provided unreliable views of the demonstration, with the arm typically self-occluding the scene. Only having one side camera could also be a constraint on the performance, given that there are no observations of the robot pushing the T from the other side.

**Teleoperation.** The real world data generation process, specifically for imitation learning approaches, also depends on reliable and intuitive teleoperation hardware for collecting expert demonstrations. Virtual reality controllers, joysticks, and scaled-down robotic arms have all been used as such hardware. The joystick used in this project was low-cost, drifted when not in operation, and required some increased cognitive load to operate. Given the position of the operator when controlling the robot, it was sometimes hard to observe both the interaction with the object and the goal state. Upgrades to the joystick, such as a 3dconnexion SpaceMouse as used in the original work, could have improved the quality of the demonstrations.

# 7 CONCLUSION

To our knowledge, our work is first to present results of training slot attention on popular imitation learning tasks in simulation (RLBench James et al. (2020)) and in the real world (Push-T Chi et al. (2023). We show that while slot attention can successfully reconstruct both the simulation and the real setup within the small-scale data provided for imitation learning, merely adding slot attention in imitation-learned policies offers no significant benefit to the policy's performance. Additionally, we highlight that the slot attention's extra compute requirement, which increases the trainig time by a factor of 5, renders the standard slot attention module infeasible for imitation learning purposes and motivate simpler object-centric modules for policy learning.

## REFERENCES


Ondrej Biza, Sjoerd van Steenkiste, Mehdi S. M. Sajjadi, Gamaleldin F. Elsayed, Aravindh Mahendran, and Thomas Kipf. Invariant slot attention: Object discovery with slot-centric reference frames. In *ICML*, 2023.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3d: 3d feature field transformers for multi-task robotic manipulation, 2023.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. *CoRL*, 2023.

Huy Ha, Pete Florence, and Shuran Song. Scaling up and distilling down: Language-guided robot skill acquisition. In *Proceedings of the 2023 Conference on Robot Learning*, 2023.

Thorsten Hempel, Ahmed A. Abdelrahman, and Ayoub Al-Hamadi. 6d rotation representation for unconstrained head pose estimation. In *2022 IEEE International Conference on Image Processing (ICIP)*. IEEE, October 2022. doi: 10.1109/icip46576.2022.9897219. URL http://dx.doi.org/10.1109/ICIP46576.2022.9897219.

Negin Heravi, Ayzaan Wahid, Corey Lynch, Pete Florence, Travis Armstrong, Jonathan Tompson, Pierre Sermanet, Jeannette Bohg, and Debidatta Dwibedi. Visuomotor control in multi-object scenes using object-aware representations. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9515–9522. IEEE, 2023.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

Jindong Jiang, Fei Deng, Gautam Singh, and Sungjin Ahn. Object-centric slot diffusion. *NeurIPS*, 2023.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538, 2020.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *CoRR*, abs/2108.03298, 2021. URL https://arxiv.org/abs/2108.03298.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.