Imitation Learning for Visual Robotic Manipulation

Yunfan Jiang
Stanford University
yjiang05@stanford.edu

Agrim Gupta*
Stanford University

Jim Fan†
NVIDIA

Abstract

A milestone in robot learning is to learn policies that can manipulate objects precisely and reason about surrounding abstract concepts in the meanwhile. In this project, we step towards this goal by learning a language-conditioned policy for visual robotic manipulation through behavioural cloning. Concretely, conditioned on text description specifying target objects to manipulate, our model aligns RGB images input with learned object embedding and predicts movements of the end effector of a robot arm that can solve the task. Our learned model can solve the manipulation task of "put an object into another object" with a high success rate above 70%. We also identify the difficulty of recognizing the correct object to manipulate and propose the solution of aligning RGB image inputs with learned object embedding. Our experiment shows that this solution significantly improves the performance when there are distractor objects.

1. Introduction

The recent decade has witnessed the success of applying deep learning to solve robotic manipulation tasks [1,12,13,17,18,22–25,29,35]. Two main techniques behind these successes are reinforcement learning [30] and imitation learning [11]. Recently, with the success of large-scale foundation models [2] such as BERT [6] and GPT-3 [3], learned deep models can reason about complicated concepts in the form of images [26, 34], videos [31], decision-making [29], and so on. The milestone in robot learning would be combining both strengths from control and reasoning so that the learned agent can reason about surrounding abstract concepts (e.g., a red cup) and manipulate them precisely (e.g., drag the red cup).

In this project, we study how to apply imitation learning, especially behavioural cloning, to learn language-conditioned visual robotic manipulation tasks to achieve the goal described above. Specifically, we follow [29, 35] to develop a visual robotic manipulation task based on the simulator PyBullet [5], leverage an oracle agent to generate expert demonstrations, and learn a policy network with ResNet [10] as the perceiving module to solve this task. To augment our model with the reasoning ability, we describe tasks with natural language sentences and condition our model on them. We identified the difficulty of recognizing correct objects to be manipulated. Therefore, we propose the solution of learning object embedding from segmentation to alleviate the burden on object recognition so that the most compute can be used at the decision-making side. With inputs of RGB images, aligned segmentation masks, states of the end effector of the robot, and text descriptions of tasks, our model predicts actions that specify the movements of a UR5[1] robot arm to solve target tasks.

Our contributions are threefold:

- We extend the benchmark suite for robotic manipulation from [29, 35] by adding more observation view and modality and decomposing the primitive robotic action to make it easier for training.
- We developed a language-conditioned policy that takes visual inputs to solve a robotic manipulation task. It can achieve a high success rate above 70% with distractor objects in the workspace.
- We identified the difficulty of recognizing the correct objects to manipulate and proposed a solution of aligning RGB images with learned object embedding to alleviate the burden at detection side. Our experiment shows that it significantly improves the performance.

2. Related Work

A considerable amount of work in visual robotic manipulation makes object-centric assumption [8,9,16,20,33,36]. These efforts put huge burdens on data collection. The Transporter Network proposed in [35] instead makes no object-centric assumptions and preserves spatial structure

*CS231N TA
†Non-CS231N collaborator

[1] https://www.universal-robots.com/products/ur5-robot/

for object manipulation with visual inputs. Concretely, consider an example of executing pick-and-place actions, it can be thought as executing a sequence of *spatial displacements*. Each segment of displacement can be specified by a start pose and an end pose. This has been shown to improve sample efficiencies on several manipulation tasks such as stacking a pyramid, assembling kits, pushing piles, etc [35]. We adopt the same practices in our network models to learn policies that can complete a visual manipulation task.

Another problem in robotic manipulation is the difficulty to generalize to new goals or quickly transfer concepts from other tasks. To address, [29] proposes CLIPort, a two-stream architecture with semantic and spatial pathways for vision-based robotic manipulation. The semantic pathway contains a contrastively learned model, CLIP [26], to extract semantic information such that the learned policy is conditioned on language instructions. The spatial pathway is similar to Transporter Network for precises manipulation. We follow the similar idea to incorporate language instructions into our robotic manipulation task and learn an instruction-conditioned policy in favor of generalization to unseen objects.

Detecting and recognizing objects is hard. Detecting the right objects to manipulate is even harder in robotic manipulation tasks. The power of decision making in our policy network is hid if too much burden is put on detection side. To this end, inspired by [4], we propose to leverage the ground-truth segmentation of objects that can be queried from the simulator. Note that this does not break the rule because segmentation can be obtained from any off-the-shelf segmentation models [32]. Our experiment also shows that our method leads to better learned manipulation policies compared to not using segmentation masks.

3. Method

3.1. Problem Formulation

In this project, we study instruction-conditioned robotic manipulation from visual inputs. Concretely, given a robotic manipulation task $\mathcal{T}$ and its corresponding text instruction y (e.g., put the red cube into the green bowl), we learn a neural network policy parameterized by θ to make decisions $\pi_\theta(a_t|o_t, y) : \mathcal{O} \rightarrow \mathcal{A}$ such that it can successfully complete the task, where $\mathcal{O}$ represents the observation space and $\mathcal{A}$ represents the action space. $o_t \in \mathcal{O}$ is the observation at time step t. Similarly, $a_t \in \mathcal{A}$ is the action at time step t. The observation space $\mathcal{O}$ contains RGB images from two views (top view and front view) denoted as $I_{\text{top}} \in \mathbb{R}^{3\times H\times W}$ and $I_{\text{front}} \in \mathbb{R}^{3\times H\times W}$, the state of the end effector (i.e., the end effector grasps some object or not) $s_{ee} \in \{[1, 0], [0, 1]\}$ represented as one-hot vector, and segmentation masks M_{top} and M_{front} paired with I_{top} and I_{front}. Note that the observation of segmentation is not included in the baseline method. It is leveraged by our method only. RGB observations from two views are shown in Figure 1.

The action space $\mathcal{A}$ contains three parts and is same for both the baseline method and our method. The first part $pos = (x, y, z) \in \mathbb{R}^3$ is the position of the end effector the agent hopes to move to. The second part $rot = (a, b, c, d) \in \mathbb{R}^4$ specifies the rotation in quaternion of the end effector during movement. The third part $release \in \{0, 1\}$ tells if the agent wants to release the suction cup. Note that the suction cup end effector will automatically suck objects it contact, there is no need to define a such action.

3.2. Robot Learning Simulator

We extend upon the Ravens benchmark suite introduced in [35] and used in [29]. It is based on Pybullet [5] and includes 10 robotic manipulation tasks. We improve and modify it in three ways.

We first make the action space easier for training models. The original action space in [29, 35] is the primitive action of "pick and place". Therefore, it involves two poses, one for the pick pose and the other for place pose. The size of the original action space $|\mathcal{A}|$ is the Cartesian product of all possible two poses, leading to inefficient and difficult training [7]. This problem is exacerbated when compute is limited. To address, we decompose the original primitive action of "pick and place" into "move end effector" as described in Section 3.1. There is only one pose in our new action space. Hence, we halve the headroom required to learn a complicated mapping from observation space to a large action space.

We also develop another new task for this project under the framework of Ravens. In the new task, the agent is required to put an object into a container object. We can also vary the number of distractor objects. The instruction for this task is "put the `object A` into the `object B`", where `object A` and `object B` are placeholders for the object being manipulated and the container object, respectively. We instantiate specific objects and full-fill the instruction when a new task instance is created.

The third improvement includes more observation modality and better observation view. We query the simulator to get segmentation masks paired with RGB images. These segmentation masks are used in our method as described in Section 3.3.1. Furthermore, agents can only observe RGB images from the front view in [29, 35]. We add another top-down view such that our agent receives more information about its surrounding objects.

3.3. Learning Language-conditioned Robotic Manipulation from Visual Inputs

We learn a policy with visual inputs for language-conditioned robotic manipulation through behavioural cloning. An overview of our method is depicted in Figure 2.

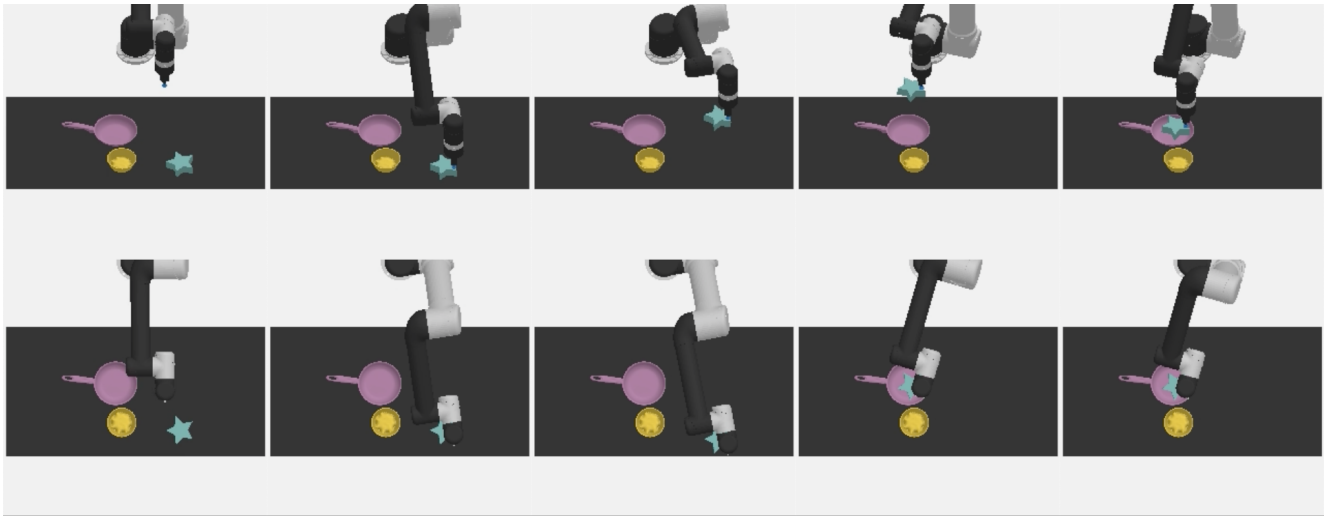

Figure 1. Visualizations of a successful trajectory completed by the oracle agent from both front and top-down views. The text instruction is "put the cyan star into the pink pan". There is a yellow bowl distractor.

The vision perceiving module is a ResNet [10] trained from scratch (specifically we use `resnet-18`). There are three critical components in our method, namely 1) aligned RGB and learned object embedding, 2) frame stacking, and 3) task descriptions conditioning through pre-trained language models. We elaborate them in following sections.

3.3.1 Aligning RGB Observations with Object Identities

Detecting and recognizing objects is hard. It becomes even harder when coupled with policy learning. Fail to detect and recognize the correct objects to be manipulated can lead to poor policy learning. We therefore propose to align RGB input images with learned object embedding from segmentation masks. Note that any off-the-shelf segmentation models can be used to obtain such segmentation masks. Here we use ground-truth segmentation masks queried from the simulator such that the most compute can be used for policy learning.

To this end, we introduce learned object embedding. Object embedding embeds object IDs to dense vectors $v_{\text{obj}} \in \mathbb{R}^D$ where D is the dimension of embedding. Recall that as described in Section 3.1, the segmentation masks M_{top} and M_{front} are paired with RGB images. They have the same spatial dimension and relationship. Because M_{top} and M_{front} only contain segmented objects represented as separate IDs, we pass M_{top} and M_{front} to object embedding to get embedded segmentation masks $M'_{\text{top}} \in \mathbb{R}^{D\times H\times W}$ and $M'_{\text{front}} \in \mathbb{R}^{D\times H\times W}$. We then concatenate M'_{top} and M'_{front} with I_{top} and I_{front} along the channel dimension, respectively. Denoting the concatenated results as $F_{\text{top}} \in \mathbb{R}^{(3+D)\times H\times W}$ and $F_{\text{front}} \in \mathbb{R}^{(3+D)\times H\times W}$, they now contain all visual information as the original RGB images have and also *aligned* object information. We then pass F_{top} and F_{front} to the ResNet. To handle these visual inputs, the input channel of the first layer of ResNet increases from 3 to $3 + D$.

3.3.2 Frame Stacking

We hypothesize that the policy can make better decisions when also conditioned on previous observations. An example is that when different degrees of occlusions occur at different time steps, the model can combine observed images from several previous time steps to make up an occlusion-free observation, which then helps predicting the correct action at the current time step.

To provide history to the model for better decision making, we stack previous x frames of F_{top} and F_{front}. Denoting the frame at time step t as $F_{\text{top},t}$ and $F_{\text{front},t}$, we stack a sequence of $\{F_{\text{top},t-x+1}, F_{\text{top},t-x+2}, \ldots, F_{\text{top},t}\}$ and $\{F_{\text{front},t-x+1}, F_{\text{front},t-x+2}, \ldots, F_{\text{front},t}\}$ to obtain $\mathbf{F}_{\text{top},t} \in \mathbb{R}^{n\times(3+D)\times H\times W}$ and $\mathbf{F}_{\text{front},t} \in \mathbb{R}^{n\times(3+D)\times H\times W}$. They are now temporally extended. We then pass them to the ResNet. Notice that this is similar to "early fusion" [14]. In our experiment detailed in Section 5.3.2, we find that stacking four frames ($n = 4$) achieves the best performance while keeps the entire computation manageable compared to stacking more.

3.3.3 Task Descriptions Conditioning

The ability to understand natural languages is the most powerful tool to help a trained agent to generalize to unseen scenes and objects. Consider two tasks with exactly the same workspace including a red cube, a green cube, and

a red bowl. The first task asks the agent to put the *red* cube into the bowl, while the second task asks to put the *green* cube into the bowl. If the agent can understand and differentiate phrases "red cube" vs "green cube", then it can apply the same control policy to successfully complete both tasks. Motivated by this, we condition our model on task descriptions by leveraging pre-trained language models. Specifically, given a text instruction y, we use a lightweight pre-trained DistillBERT [27] to extract a feature vector $v_y \in \mathbb{R}^{768}$ and then pass to our model. Note that the entire DistillBERT is *frozen* during training.

3.3.4 Learning through Behavioural Cloning

We learn policies parameterized by θ through behavioural cloning [11]. Concretely, we assume an offline dataset $\mathcal{D}^{\mathcal{T}} = \{\tau^{(1)}, \tau^{(2)}, \ldots, \tau^{(N)}\}$ containing N trajectories. Each trajectory $\tau^{(n)} = \{o_0, a_0, o_1, a_1, \ldots, o_T\}$ is a sequence of observations and actions. Section 4 elaborates how to obtain such an offline dataset. Given each transition pair (o_t, a_t) in a trajectory $\tau^{(n)} \in \mathcal{D}^{\mathcal{T}}$, denoting the predicted action from our policy $\pi_\theta(\cdot)$ as $\hat{a}_t$, we define the loss function $\mathcal{L}(a_t, \hat{a}_t) : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$ and update our policy network using stochastic gradient descent with gradient calculated from $\nabla_\theta \mathcal{L}(a_t, \hat{a}_t) = \nabla_\theta \mathcal{L}(a_t, \pi_\theta(o_t, y))$. The loss function $\mathcal{L}$ can be L2 loss for continuous action or cross-entropy loss for discrete action. In our case we find that predicting continuous actions and hence optimizing L2 loss works better than discrete action and cross-entropy loss. The final loss we optimize is a weighted combination of three parts, namely

$$\mathcal{L}_{pos} = \frac{1}{3} \sum_{dim \in \{x,y,z\}} \left(a_{pos,dim} - \hat{a}_{pos,dim}\right)^2, \quad (1)$$

$$\mathcal{L}_{rot} = \frac{1}{4} \sum_{dim \in \{a,b,c,d\}} \left(a_{rot,dim} - \hat{a}_{rot,dim}\right)^2, \quad (2)$$

and

$$\mathcal{L}_{release} = \left(a_{release} - \hat{a}_{release}\right)^2. \quad (3)$$

Note that we omit the timestep subscript t for simplicity.

4. Dataset Collection

The previous section implies that we can only learn a good policy only if the offline dataset we imitate has a high quality. To collect such a dataset, we leverage a stochastic oracle provided by [35] to generate successful trajectories. The oracle agent leverages ground-truth information such as noiseless top-down images from a near-orthographic camera and is programmed with task-specific trajectories (e.g., which object to manipulate, where to move the end effector). The action space of the oracle agent is same as our agents'. Therefore, to generate the dataset, we rollout the oracle agent π_{oracle} in task $\mathcal{T}$, record the observations and actions at each time step and construct the trajectory $\tau^{(n)}_{\pi_{\text{oracle}}}$, and append new trajectory into the dataset $\mathcal{D}^{\mathcal{T}}_{\pi_{\text{oracle}}} \leftarrow \mathcal{D}^{\mathcal{T}}_{\pi_{\text{oracle}}} \cup \tau^{(n)}_{\pi_{\text{oracle}}}$. As the oracle is a stochastic one, there is still a small probability that it will fail. We therefore exclude failed trajectories and only include successful trajectories such that $f^{\mathcal{T}}_{\text{eval}}(\tau) = 1, \forall \tau \in \mathcal{D}^{\mathcal{T}}_{\pi_{\text{oracle}}}$, where $f^{\mathcal{T}}_{\text{eval}}(\tau) \in \{0, 1\}$ is an evaluation function that takes an observation-action sequence and evaluates whether the task is successfully completed or failed.

We generate $1,000$ successful trajectories for the task we are interested in (i.e., the task "put the `object A` into the `object B`"). Note that $1,000$ successful trajectories are not insufficient because [29] already shows that this order of magnitude of data are enough to learn good manipulation policies. Each trajectory contains 5 observations (including the terminal observation) and paired 4 oracle actions. Therefore, we have roughly $4,000$ successful transitions to imitate. We thus use all of them as the training data and evaluate our model through online evaluation. In other words, we test our trained model by letting it attempt to complete the task and record success rates. One successful oracle trajectory is shown in Figure 1. Algorithm 1 illustrates our dataset collection pipeline.

We use an image size of 128×256 for both RGB images and segmentation masks. We normalize RGB pixel values to be within $[0, 1]$ by dividing 255 before passing to our model. We follow the procedure described in Section 3.3.1 to process segmentation masks. We do not include any data augmentation now.

5. Experiments, Results, and Discussion

In this section, we start with instantiation details of our robotic manipulation task. We also briefly present important hyper-parameters used during training. We then introduce the primary metric we care about and elaborate the experiment results.

5.1. Task Instantiation Details

We instantiate manipulation tasks from a single template "put the `object A` into the `object B`". Both object placeholders contain two part, a color adjective and a noun. All colors and objects are listed in Table 1. Every time we sample a specific combination. We also add a distractor into the workspace. We sample a combination of color and object for the distractor as well. Positions of all objects are randomly sampled from the workspace.

5.2. Hyper-parameters

During training we use a mini-batch size of 20 due to our GPU memory limitation. We use Adam optimizer [15]

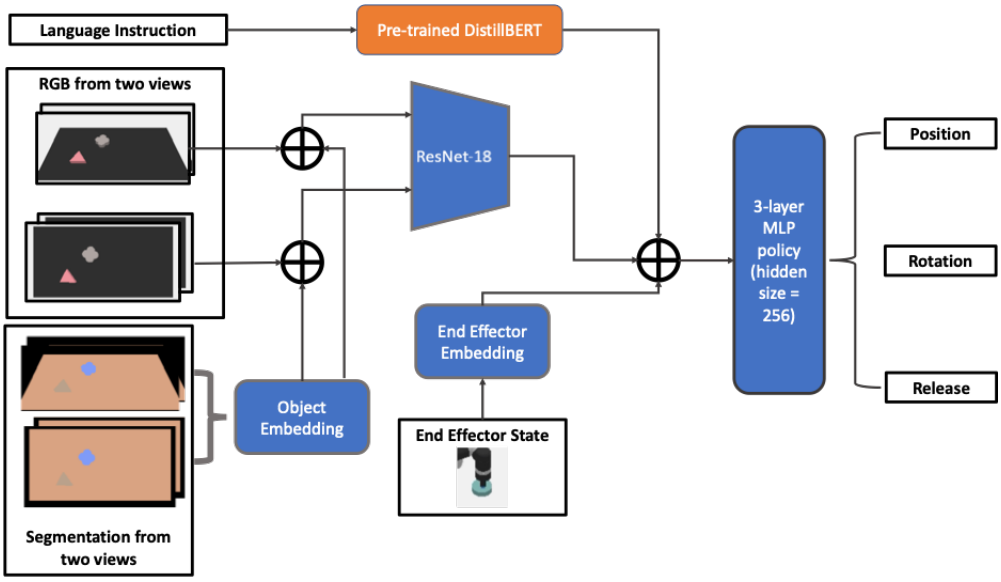

Figure 2. Schematic of our model architecture. Symbol $\oplus$ denotes concatenation.

with an initial learning rate of 5×10^{-4}. We decay the initial learning rate to 1×10^{-5} following a Cosine annealing scheme [19] because it is empirically proven to be able to stabilize the training. We train all models for 450 epochs. Each trial takes approximately 8 hours on a RTX 2080 Ti GPU.

5.3. Main Results

The primary metric we care about is (averaged) success rate. Specifically, we alternate between training and evaluation every 10 epochs. During evaluation, we let the (partially) trained model rollout in the simulator for 40 episodes and then compute the average success rate.

Table 2 shows the main experimental results. Figure 3 shows the curve of success rate vs training epoch of our method described in Section 3.3. Our model is able to achieve an average success rate around 70% after training for 200 epochs. The best success rate it achieved is 85%. This result is promising and suggests that our model is able to reason about its complicated surrounding environment meanwhile complete manipulation tasks precisely. We then ablate our different design components to investigate each part's contribution to the performance.

The evaluation performance also demonstrates that our method does not overfit. Although at the first glance it seems that ours can easily overfit because we only have $4,000$ observation-action pairs to imitate. However, it indeed does not because the entire state space is very large. Roughly, the state space is at the order of the area of the workspace to the power of number of objects. This huge amount of possible task instances prevents our model to overfit.

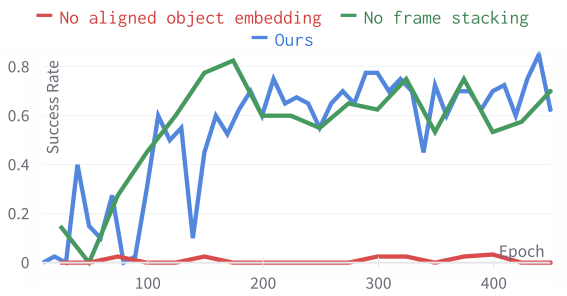

Figure 3. Success rate of our method and ablated methods

5.3.1 Ablation on Aligned RGB and Object Embedding

We ablate the aligned RGB and object embedding as introduced in Section 3.3.1. Surprisingly, we find that using aligned RGB and object embedding contributes the mostly to the final performance because the model without it cannot even solve the task (success rate around 3%). This finding supports our hypothesis that we cannot learn a good policy for decision making if the burden on object detection is too heavy. Indeed, aligning RGB images with learned object embedding helps to alleviate the learning difficulty at the perceiving side, which then fully showcases the ability to do complicated reasoning.

5.3.2 Ablation on Frame Stacking

We then ablate the effects of stacking multiple frames to provide historical information to the model as described in

Algorithm 1: Imitation Dataset Collection Pipeline.

Input: $\pi_{\text{oracle}}(a|o)$: policy of the oracle agent
Input: $env(o, \text{done}|a)$: simulator environment function that accepts an action, forwards one step, and returns a new observation and a termination signal
Input: $f^{\mathcal{T}}_{\text{eval}}(\tau)$: an evaluation function that outputs if the task is successfully completed or not

Initialize an empty dataset buffer $\mathcal{D}^{\mathcal{T}}_{\pi_{\text{oracle}}} \leftarrow \emptyset$;
while *have not collected N successful trajectories* **do**
  Initialize an empty buffer to store the current trajectory $\tau^{(n)}_{\pi_{\text{oracle}}} \leftarrow \emptyset$;
  Reset the simulator to get the initial observation o and append to $\tau^{(n)}_{\pi_{\text{oracle}}}$;
  while *the current episode is not terminated* **do**
    Obtain oracle action $a = \pi_{\text{oracle}}(o)$ and append a into $\tau^{(n)}_{\pi_{\text{oracle}}}$;
    Obtain new observation and termination signal from the simulator $o, \text{done} = env(a)$;
    Append the new observation to $\tau^{(n)}_{\pi_{\text{oracle}}}$. Terminate the episode if done.
  end
  if $f^{\mathcal{T}}_{eval}(\tau^{(n)}_{\pi_{oracle}}) = 1$ **then**
    $\mathcal{D}^{\mathcal{T}}_{\pi_{\text{oracle}}} \leftarrow \mathcal{D}^{\mathcal{T}}_{\pi_{\text{oracle}}} \cup \tau^{(n)}_{\pi_{\text{oracle}}}$
end

Object to be manipulated	
Colors:	Red, blue, orange, cyan, purple
Objects:	Block, L-shaped block, round, star, flower
Container object	
Colors:	Green, pink, yellow, brown, white
Objects:	Bowl, frame, box, pallet, pan

Table 1. All colors and objects used to instantiate tasks.

Method	Success Rate
Ours	85%
Without aligned object embedding	3%
Without frame stacking	82.5%

Table 2. Best success rates of our method, without aligned object embedding, and without frame stacking.

Section 3.3. We find that it has minimal impacts on the final performance as shown in Figure 3. We compare our method with 4 frames stacked and a model without frame stacked. Both of them achieve a similar sample efficiency. The best success rate of the model without frame stacking is slightly lower than ours (82.5% vs 85%). Nevertheless, frame stacking is a common practice used in deep reinforcement learning (RL) to provide historical context to RL agents such that they can have better estimation of state (or state-action) values and better policy [21, 28]. We hypothesize that the reason that frame stacking does not offer significant helps in our case is our training paradigm is much easier and more stable than deep RL. Our agent does not require much history to complete the manipulation tasks.

5.4. Failure Case Analysis

We now qualitatively analyze failure cases of our method. We find that in most failed cases, our agent can successfully pick up the correct object. However, it fails to put it into the correct container. One representative trajectory is visualized in Figure 4. We hypothesize that this is because we use continuous actions. The policy head predicts position in $\mathbb{R}^3$ and rotation in $\mathbb{R}^4$. Any small disturbances can cause the output values change. A potential solution is to discretize the action space and use discrete actions. However, our preliminary results (not included here) show that training with discrete actions is worse than training with continuous actions. We leave the investigation as future work to improve the robustness of the learned policy.

6. Conclusion and Future Work

Learning agents that can reason about their complicated surrounding environments and do precise control is a long-standing goal in robot learning and embodiment learning. In this project, we step towards this goal by learning a language-conditioned model to solve robotic manipulation tasks from visual inputs. Our learned model is not only able to follow specific instructions to complete tasks but also can precisely manipulate objects. We verified our hypothesis that the difficulty to learn object detection can obstacle the learning of decision making. We therefore propose the key component behind our method to align RGB images with learned object embedding. In our experiments, we also ablate the effects of frame stacking and do not find a huge difference, which is probably due to the fact that imitating noiseless successful trajectories already stabilizes the training process. Meanwhile, we qualitatively study failure cases of our learned model and identify possible directions for further improvements.

One direction of future work is to improve the robustness of the learned model. For example, can the model still work reliably under different workspace with different illumination conditions? Other promising directions include to scale up the richness and quantity of manipulation tasks such that

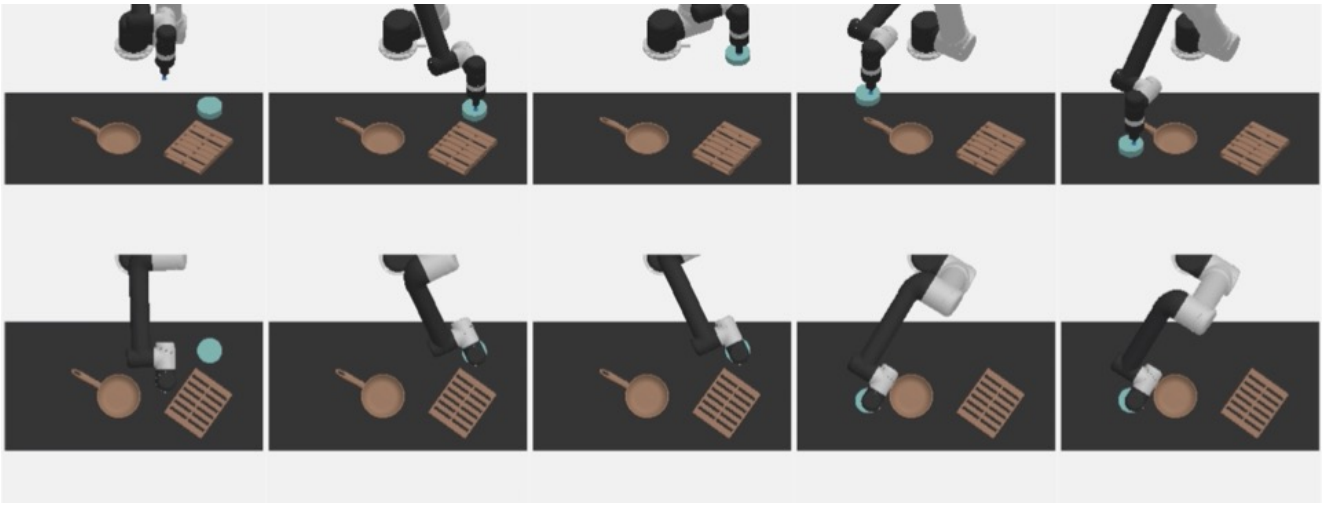

Figure 4. Visualizations of our method's failed trajectory. The text instruction is "put the cyan round into the brown pan". There is a brown pallet distractor. Our model successfully picks up the correct object but fails to put it into the container.

we can use the same framework to solve more diverse problems.

7. Contributions and Acknowledgements

In this project, we extend upon codebases from Transporter[2] and CLIPort[3]. As non-CS231N collaborators, Jim and Agrim supervise this project at a high level. This work has not been submitted for any peer-reviewed conferences or journals. Part of experiments in this project is conducted using GPUs from Stanford Vision and Learning Lab.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, and Mengyuan Yan. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv: Arxiv-2204.01691*, 2022. 1

[2] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *arXiv preprint arXiv: Arxiv-2108.07258*, 2021. 1

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language mod-

[2] https://github.com/google-research/ravens

[3] https://github.com/cliport/cliport

els are few-shot learners. *arXiv preprint arXiv: Arxiv-2005.14165*, 2020. 1

[4] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. *arXiv preprint arXiv: Arxiv-1912.12294*, 2019. 2

[5] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019. 1, 2

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv: Arxiv-1810.04805*, 2018. 1

[7] Gregory Farquhar, Laura Gustafson, Zeming Lin, Shimon Whiteson, Nicolas Usunier, and Gabriel Synnaeve. Growing action spaces. *arXiv preprint arXiv: Arxiv-1906.12266*, 2019. 2

[8] Peter Florence, Lucas Manuelli, and Russ Tedrake. Self-supervised correspondence in visuomotor policy learning. *arXiv preprint arXiv: Arxiv-1909.06933*, 2019. 1

[9] Peter R. Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. *arXiv preprint arXiv: Arxiv-1806.08756*, 2018. 1

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. 1, 3

[11] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2), apr 2017. 1, 4

[12] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv: Arxiv-1806.10293*, 2018. 1

[13] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv: Arxiv-2104.08212*, 2021. 1

[14] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014. 3

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv: Arxiv-1412.6980*, 2014. 4

[16] Tejas Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. *arXiv preprint arXiv: Arxiv-1906.11883*, 2019. 1

[17] Youngwoon Lee, Joseph J. Lim, Anima Anandkumar, and Yuke Zhu. Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization. *arXiv preprint arXiv: Arxiv-2111.07999*, 2021. 1

[18] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016. 1

[19] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv: Arxiv-1608.03983*, 2016. 5

[20] Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. *arXiv preprint arXiv: Arxiv-1903.06684*, 2019. 1

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv: Arxiv-1312.5602*, 2013. 6

[22] Anusha Nagabandi, Kurt Konoglie, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. *arXiv preprint arXiv: Arxiv-1909.11652*, 2019. 1

[23] Suraj Nair, Eric Mitchell, Kevin Chen, Brian Ichter, Silvio Savarese, and Chelsea Finn. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. *arXiv preprint arXiv: Arxiv-2109.01115*, 2021. 1

[24] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand. *arXiv preprint arXiv: Arxiv-1910.07113*, 2019. 1

[25] OpenAI OpenAI, Matthias Plappert, Raul Sampedro, Tao Xu, Ilge Akkaya, Vineet Kosaraju, Peter Welinder, Ruben D'Sa, Arthur Petron, Henrique P. d. O. Pinto, Alex Paino, Hyeonwoo Noh, Lilian Weng, Qiming Yuan, Casey Chu, and Wojciech Zaremba. Asymmetric self-play for automatic goal discovery in robotic manipulation. *arXiv preprint arXiv: Arxiv-2101.04882*, 2021. 1

[26] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 2021. 1, 2

[27] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv: Arxiv-1910.01108*, 2019. 4

[28] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv: Arxiv-1707.06347*, 2017. 6

[29] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. *arXiv preprint arXiv: Arxiv-2109.12098*, 2021. 1, 2, 4

[30] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. 1

[31] Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer. Videoclip: Contrastive pre-training for zero-shot video-text understanding. *arXiv preprint arXiv: Arxiv-2109.14084*, 2021. 1

[32] Pavel Yakubovskiy. Segmentation models. https://github.com/qubvel/segmentation_models, 2019. 2

[33] Youngrock Yoon, G.N. DeSouza, and A.C. Kak. Real-time tracking and pose estimation for industrial objects using geometric features. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 3, pages 3473–3478 vol.3, 2003. 1

[34] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv: Arxiv-2205.01917*, 2022. 1

[35] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Ayzaan Wahid, Vikas Sindhwani, and Johnny Lee. Transporter networks: Rearranging the visual world for robotic manipulation. *arXiv preprint arXiv: Arxiv-2010.14406*, 2020. 1, 2, 4

[36] Menglong Zhu, Konstantinos G. Derpanis, Yinfei Yang, Samarth Brahmbhatt, Mabel Zhang, Cody Phillips, Matthieu Lecce, and Kostas Daniilidis. Single image 3d object detection and pose estimation for grasping. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3936–3943, 2014. 1