Learning Robotic Manipulation Tasks via Task Progress based Gaussian Reward and Loss Adjusted Exploration

Sulabh Kumra, Shirin Joshi and Ferat Sahin

Abstract—Multi-step manipulation tasks in unstructured environments are extremely challenging for a robot to learn. Such tasks interface high-level reasoning that consists of the expected states that can be attained to achieve an overall task and low-level reasoning that decides what actions will yield these states. We propose a model-free deep reinforcement learning method to learn multi-step manipulation tasks. We introduce a Robotic Manipulation Network (RoManNet), which is a vision-based model architecture, to learn the action-value functions and predict manipulation action candidates. We define a Task Progress based Gaussian (TPG) reward function that computes the reward based on actions that lead to successful motion primitives and progress towards the overall task goal. To balance the ratio of exploration/exploitation, we introduce a Loss Adjusted Exploration (LAE) policy that determines actions from the action candidates according to the Boltzmann distribution of loss estimates. We demonstrate the effectiveness of our approach by training RoManNet to learn several challenging multi-step robotic manipulation tasks in both simulation and real-world. Experimental results show that our method outperforms the existing methods and achieves state-of-the-art performance in terms of success rate and action efficiency. The ablation studies show that TPG and LAE are especially beneficial for tasks like multiple block stacking. Code is available at: https://github.com/skumra/romannet

Index Terms—Robotic manipulation, reinforcement learning, deep learning.

I. INTRODUCTION

Robotic manipulation tasks have been the backbone of most industrial robotic applications, e.g. bin picking, assembly, palletizing, or machine tending operations. In structured scenarios, these tasks have been reliably performed by the methods used in the existing work [1], [2]. While, in unstructured scenarios, simple tasks such as pick only tasks have been successfully performed using grasping approaches such as [3]–[5], complex tasks that involve multiple steps such as clearing a bin of mixed items, and creating a stack of multiple objects, remain a challenge.

Training end-to-end manipulation policies that map directly from image pixels to joint velocities can be computationally expensive and time exhaustive due to a large volume of sample space and can be difficult to adapt on physical setups [6]–[10]. To solve this, many have tried pixel-wise parameterization of both state and action spaces, which enables the use of a neural network as a Q-function approximator [11]–[13]. However, these approaches have a low success rate, long learning time, and cannot handle intricate tasks consisting of multiple steps and long horizons.

In this work, we present a model-free deep reinforcement learning approach (shown in Fig. 1) to produce a deterministic policy that allows complex robot manipulation tasks to be effectively learned from pixel input. The policy directs a low-level controller to perform motion primitives rather than regressing motor torque vectors directly by learning a pixel-wise action success likelihood map. We propose a Task Progress based Gaussian (TPG) reward function to learn the coordinated behavior between intermediate actions and their consequences towards the advancement of an overall task goal. We introduce a Loss Adjusted Exploration (LAE) policy to explore the action space and exploit the knowledge, which helps in reducing the learning time and improving the action efficiency. The key contributions of this work are:

• We introduce Robotic Manipulation Network (RoManNet), an end-to-end model architecture to efficiently learn the action-value functions and generate accurate action candidates from visual observation of the scene.
• We propose a TPG reward function that uses a sub-task indicator function and an overall task progress function to compute the reward for each action in a multi-step manipulation task.
• We address the challenge of balancing the ratio of exploration/exploitation by introducing an LAE manipulation policy that selects actions according to the Boltzmann distribution of loss estimates.

We demonstrate the effectiveness of our approach in simula-
tion as well as in the real-world setting by training an agent to learn three vision-based multi-step robotic manipulation tasks. RoManNet trained with TPG reward and LAE policy performed significantly better than previous methods with only 2000 iterations in the real-world setting. We observed a pick success rate of 92% for a mixed item bin-picking task and 84% action efficiency for a block-stacking task.

II. RELATED WORK

Robotic manipulation has always been an essential part of research in the field of robotics. There has been significant progress in recent years in robotic grasping that leverages deep learning and computer vision for generating grasp candidates for specific tasks that can be used to select suitable grasp poses for novel objects in fairly structured environments [14]–[16]. While most prior manipulation methods focus on singular tasks, our work focuses on learning multi-step manipulation tasks which generalize to a wide range of objects and tasks.

Deep reinforcement learning can be used to learn complex robotic manipulation tasks by using model-free deep policies. Kalashnikov et al. demonstrate this by proposing a QT-opt technique to provide a scalable approach for vision-based robotic manipulation applications [7]. Riedmiller et al. introduced a SAC-X method that learns complex tasks from scratch with the help of multiple sparse rewards where only the end goal is specified [17]. The agent learned these tasks by exploring its observation space and the results showed that this technique was highly reliable and robust. More recently, Xu et al. introduced a learning-to-plan method called Deep Affordance Foresight (DAF), which learns partial environment models of affordances of parameterized motor skills through trial-and-error [18]. Nematolahi et al. demonstrated that scene dynamics in the real-world can be learned for visuomotor control and planning [19].

Learning multi-step tasks with sparse rewards is particularly challenging because a solution is improbable through random exploration. Tasks such as clearing a bin with multiple objects [7], [20] do not include long-horizon and the likelihood of reverse progress is out of consideration. Many propose to use a model-free method through self-supervised learning. One such method proposed by Zeng et al. uses a visual pushing grasping (VPG) framework that can discover and learn to push and grasp through model-free deep Q learning [11]. Similarly, Jeong et al. performed a stacking task by placing a cube over another cube using a two-stage self-supervised domain adaptation (SSDA) technique [21]. Zhu et al. presented a framework in which manipulation tasks were learned by using a deep visuomotor policy (DVP) that uses a combination of reinforcement learning and imitation learning to map RGB camera inputs directly into joint velocities [22]. Hundt et al. developed the SPOT framework [12], that explores actions within the safety zones and can identify unsafe regions even without exploring and can prioritize its experience to only learn what is useful. Zeng et al. proposed Transporter Network trained using learning from demonstrations, which rearranges deep features to infer spatial displacements from visual input [23]. Kase et al. proposed a Deep Planning Domain Learning (DPDL) framework which learns a high-level model using sensor data to predict values for a large set of logical predicates consisting of the current symbolic world state and separately learns a low-level policy which translates symbolic operators into robust executable actions on a robot [24]. DPDL framework worked well on manipulation tasks in a photorealistic kitchen scenario. Similarly, Driess et al. proposed a network architecture consisting of a high-level reasoning network, adaptation network, and low-level controllers to learn geometrically precise manipulation tasks for a dual robot arm system where the parameters of early actions are tightly coupled with those of later actions [25]. Kit assembly [26], [27], and cloth manipulation [28], [29] are some other tasks that involve multiple steps.

In our work, we focus on multi-step tasks that involve long-horizon planning and considers progress reversal. More closely related to our work is the VPG framework introduced by Zeng et al. which utilized a Fully Convolutional Network (FCN) as a function approximator to estimate the action-value function. However, this method had a low success rate and was sample inefficient. We introduce a sample efficient RoManNet framework to learn multi-step manipulation tasks, which is trained using a TPG reward function and LAE policy to address these problems. In addition to the grasp and pre-grasp primitives explored by Zeng et al., we explore the placement primitive to evaluate our approach on multi-step manipulation tasks like building a stack of blocks.

III. APPROACH

We consider the problem of efficiently learning multi-step robotic manipulation for unknown objects in an environment with unknown dynamics. Each manipulation task can be formulated as a Markov decision process where at any given state $s_t \in \mathcal{S}$ at time $t$, the robot makes an observation $o_t \in \mathcal{O}$ of the environment and executes an action $a_t \in \mathcal{A}$ based on policy $\pi(s_t)$ and receives an immediate reward of $R_{a_t}(s_t, s_{t+1})$. In our formulation, $o_t \in \mathbb{R}^{4 \times h \times w}$ is the visual observation of the robot workspace from RGB-D camera, and we divide the action space $\mathcal{A}$ into two components: action type $\Phi$ and action pose $\Psi$. The underlying assumption is that the edges of $o_t$ are the boundaries of the agent’s workspace, and $o_t$ embeds all necessary state information, thus providing sufficient information of the environment to choose the correct action.

We introduce RoManNet to approximate the action-value function $Q_{\phi}$, which predict manipulation action candidates $A_{a_t}$ from the observation $o_t$ of the state $s_t$ at time $t$. The proposed LAE manipulation policy $\Pi_{\lambda_{AE}}(s_t, C_t)$ determines the action $a_t$ from the action candidates $A_{a_t}$ that maximizes the reward $R$. Once the agent executes $a_t$, the reward is computed using a TPG reward function. The parameters $\mu$ are updated by minimizing the loss function. An overview of the proposed learning framework is illustrated in Fig. 2.

A. Manipulation Action Space

We parameterize each manipulation action $a_t$ as two components: action type $\Phi$, which consists of three high-level motion
primitives \{push, pick, place\}, and action pose \(\Psi\), which is defined by the pose at which the action is performed. Each manipulation action pose in the robot frame is defined as:

\[
\Psi_r = (P, \theta_r, Q)
\]

(1)

where, \(P = (x, y, z)\) is the gripper’s center position, \(\theta_r\) is the rotation of the gripper around the z-axis, and \(Q\) is an affordance score that depicts the “quality” of the action. The manipulation action pose in image frame \(\Psi_i\) is parameterized pixel-wise and defined as:

\[
\Psi_i = (x, y, \theta_i, Q)
\]

(2)

where \((x, y)\) is the center of action pose in image coordinates, \(\theta_i\) is the rotation in the image frame, and \(Q\) is the same affordance score as in equation (1).

The high-level motion primitive behaviors \(\Phi\) are defined as follows:

- **Pushing**: \(\Psi_{push} = (x, y, \theta, Q)\) denotes the starting pose of a 10cm push. A push starts with the gripper closed at \((x, y)\) and moves horizontally at a fixed distance along \(\theta\).

- **Picking**: \(\Psi_{pick} = (x, y, \theta, Q)\) denotes the middle position of a top-down grasp. During a pick attempt, both fingers attempt to move 3cm below \(\Psi_{pick}\) (in the \(-z\) direction) before closing the fingers.

- **Placing**: \(\Psi_{place} = (x, y, \theta, Q)\) denotes the middle position of a top-down placement. During a place attempt, both fingers open when the place pose \(\Psi_{place}\) is reached.

**B. Learning the Action-Value Functions**

The proposed algorithm to learn the action-value function is described in algorithm 1. We approximate the action-value function using a neural network architecture we call RoManNet. In contrast to the DenseNet-121 fully connected networks as in [11], [12], it is built using three lightweight Generative Residual Convolutional Neural Network (GR-ConvNet) [16] models: PushNet, PickNet, and PlaceNet, one for each motion primitive behavior (pushing, picking, and placing respectively). Each individual GR-ConvNet model takes as input the image representation \(o_t \in \mathbb{R}^{h \times w}\) of the state \(s_t\) and generates a pixel-wise map of Q values \(Q_{\phi}(s_t, a_t) \in \mathbb{R}^{h \times w}\) and corresponding rotation angle for each pixel \(\theta_{\phi}(s_t, a_t) \in \mathbb{R}^{h \times w}\) with the same image size and resolution as \(o_t\). In this work, \(h\) and \(w\) are same and \(\mathbb{R}^{h \times w}\) is represented as an image of
resolution 224 × 224. The overall Q-function $Q_{\mu}(s_t, a_t)$ selects the action $a_t$ that maximizes the Q-value over pixel-wise maps for all manipulation action poses: $\arg\max Q_{\mu}(s_t, a_t) = \arg\max (Q_{\text{push}}(s_t), Q_{\text{pick}}(s_t), Q_{\text{place}}(s_t))$, which gives the center position $(x, y)$ of the action in image coordinates and corresponding pixel in the rotation angle pixel-wise map $\theta_{tp}(s_t, a_t)$ gives the rotation of the gripper $\theta_t$.

The RoManNet is continuously trained to approximate the optimal policy with prioritized experience replay using stochastic rank-based prioritization and leverages future knowledge via a recursively defined expected reward function:

$$E_\pi(s_{t+1}, a_t) = R(s_t, a_t) + \gamma \mathbb{E}(\pi(s_{t+2}, a_{t+1}))$$

where, $\gamma \in [0, 1)$ is the discount factor and $\eta$ is a reward propagation factor. A value of $\gamma$ closer to 0 indicates that the agent will choose actions based only on the current reward and a value approaching 1 indicates that the agent will choose actions based on the long-term reward. The reward propagation factor $\eta$ ensures that future rewards only propagate across time steps where subtasks are completed successfully. The value of $\eta$ at time step $t$ equals to 1 if $R(s_t, a_t) > 0$, and 0 otherwise.

The loss of the network is computed using the Huber loss function at each time step $t$ as:

$$L_t = \begin{cases} \frac{1}{2} \delta(t)^2 & \text{for } |\delta(t)| \leq 1, \\ |\delta(t)| - \frac{1}{2} & \text{otherwise,} \end{cases}$$

where $\delta(t) = Q_{\mu}(s_t, a_t) - Y_t$ is the TD-error and $\mu_t$ are the weights of RoManNet at time $t$. We pass the gradients only through the network from which the predictions of the executed action $a_t$ were computed. For an antipodal gripper, the pick and place motion primitives are symmetrical around $0$-$180^\circ$, thus we can pass the gradients for the opposite orientation as well. The models are trained using Stochastic Gradient Descent with a fixed learning rate of $10^{-4}$, a momentum of 0.9, and weight decay of $2^{-5}$. The models are continuously trained on past trials using prioritized experience replay with future discount $\gamma = 0.5$.

### C. Task Progress based Gaussian Reward

The reward function $R(s_t, a_t) \in \mathbb{R}^{h \times w}$ operates on two principles: actions which advance overall task progress receive a reward proportional to the quantity of progress, but actions which reverse the progress receive 0 reward. The task progress is measured using: (i) a sub-task indicator function $X(s_{t+1}, a_t)$, which equals to 1 if $a_t$ leads to a successful primitive action and 0 otherwise, and (ii) an overall task progress function $P(s_{t+1}, a_t) \in [0, 1]$, which is proportional to the progress towards an overall goal. We define our task progress based reward function as:

$$R_{tp}(s_{t+1}, a_t) = \mathbb{W}(\Phi)X(s_{t+1}, a_t)P(s_{t+1}, a_t)$$

where $\mathbb{W}(\Phi)$ is a weighting function that depends on the primitive motion action type $\Phi$. In this work, we set $\mathbb{W}(\Phi)$ to 0.2 for pushing action, and 1 for picking and placing actions. In our experiments, to compute the sub-task indicator $X(s_{t+1}, a_t)$, a push action is successful if it perturbs an object, a pick action is successful if an object is grasped and raised from the surface, and a place action is successful only if it increases the stack height. The overall task progress function $P(s_{t+1}, a_t)$ is computed using the depth image. For item removal related tasks, it is the ratio of number of occupied pixels to the total number of pixels, and for the stacking task it is proportional to the height of the stack.

The task progress based reward function $R_{tp}$ is a single pixel in $\mathbb{R}^{h \times w}$ and to make the reward function more robust, we smooth it using an anisotropic Gaussian distribution $\mathbb{G}$ parameterized with standard deviations $\sigma_x$ and $\sigma_y$. We used an anisotropy ratio of 2 (i.e., $\sigma_x/\sigma_y = 2$), where $x$ and $y$ are the axis of the anisotropic Gaussian distribution and $x$ is aligned with the $x$-axis of the gripper. The proposed Task Progress based Gaussian (TPG) reward function is specified as follows:

$$G(x, y, \sigma_x, \sigma_y) = \frac{1}{2\pi\sigma_x\sigma_y}e^{-\frac{x^2}{2\sigma_x^2} - \frac{y^2}{2\sigma_y^2}}$$

$$R_{tp}(s_{t+1}, a_t) = R_{tp}(s_{t+1}, a_t) \odot G(x, y, \sigma_x, \sigma_y)$$

$$R_{tpg}(s_{t+1}, a_t) = \max(R_{tp}(s_{t+1}, a_t), R_{tpg}(s_{t+1}, a_t))$$

where $\odot$ is the convolution operator, and $G$ is the anisotropic Gaussian filter applied to $R_{tp}$. The intuition here is that action poses in the local neighborhood should yield similar reward. We experimentally show the advantages of using $R_{tpg}$ as compared to $R_{tp}$ in section IV-E.

### D. Loss Adjusted Exploration Policy

We introduce a Loss Adjusted Exploration (LAE) policy to reduce unnecessary exploration once knowledge about initial states has been sufficiently established. The LAE policy extends the $\epsilon$-greedy policy similar to the Value-Difference based Exploration [31], and eliminates the need for hand-tuning the exploration rate $\epsilon$ as in [11], [12]. We define the update steps for loss dependent exploration probability $\mathcal{E}(L_t)$ as the following Boltzmann distribution of loss estimates:

$$f(L_t, \sigma) = \frac{1 - e^{(-|\sigma - L_t|)}}{1 + e^{(-|\sigma - L_t|)}}$$

$$\mathcal{E}_{t+1}(L_t) = \beta \cdot f(L_t, \sigma) + (1 - \beta) \cdot \mathcal{E}_t(L_t)$$

where $L_t$ is the loss of the network computed using equation (4) at each time step $t$, $\sigma$ is a positive constant called inverse sensitivity, $\alpha$ is a constant set to 0.5 in our work, and $\beta \in [0, 1]$ is a parameter determining the influence of the selected action on the exploration rate. The LAE policy is defined as:

$$\Pi_{LAE}(s_t, \mathcal{L}_t) = \begin{cases} \mathcal{U}(\mathcal{A}_{s_t}) & \text{if } \xi < \mathcal{E}(L_t), \\ \arg\max_{a_t \in \mathcal{A}(s_t)} Q_{\mu}(s_t, a_t) & \text{otherwise.} \end{cases}$$

where $\mathcal{U}$ is a uniform distribution over action candidates $\mathcal{A}_{s_t}$, and $\xi \in [0, 1]$ is a random number drawn at each time step $t$ from a uniform distribution.
Fig. 3: Visualization of various tasks being executed in simulation using our trained model. The first one is the dense clutter removal task, where the robot can be seen picking objects from the clutter of 30 objects in a bin and removing them from the bin one after another. We observe that the robot first picks any objects that are away from the clutter. The second is one of the challenging arrangements which has tightly packed objects. We observe that the robot first pushes the packed objects to de-clutter and then picks them up. The third is the block stacking task with a goal stack height of 4. We can see the robot performing consecutive pick and place actions to build a stack using 10 cubes randomly placed in the bin. The robot learns not to pick blocks from the stack being built as it leads to progress reversal and thus a lower reward.

IV. EXPERIMENTS

We conduct experiments in both simulated and real settings to evaluate the proposed method across various tasks. We design our experiments to investigate the following three questions: (i) How well does our method perform on different multi-step manipulation tasks? (ii) Does our method improve task performance as compared to baseline methods? (iii) What are the effects of the individual components of our framework in solving multi-step manipulation tasks, i.e. without the Loss Adjusted Exploration or Task Progress based Gaussian reward?

A. Experimental Setups

For the simulated environment, we expanded on the CoppeliaSim based setup used in [11] to provide a consistent environment for fair comparisons and ablations. The environment simulates the agent using a UR5 robot arm with an RG2 gripper. Bullet Physics 2.83 is used to simulate the dynamics and CoppeliaSim’s internal inverse kinematics module for robot motion planning. A statically mounted perspective 3D camera is simulated in the environment to capture the observations of the states. Color and depth images of size 640 × 480 are rendered with OpenGL using the simulated camera. For the real-world setup, we used a UR10 robot with a Piab suction cup as the end effector. The RGB-D images of the scene are captured using an Intel RealSense D415 mounted rigidly above the workspace.

B. Evaluation Metrics

For evaluating the trained model, the policy is greedy deterministic and the model weights are reset to the trained weights at the start of each new test run. For each of the test cases, we execute 30 runs with new random seeds and evaluate performance with the following metrics found in [11], [12]:

- Completion rate: the average percentage runs in which the policy completed the given task without 10 consecutive fail attempts.
- Pick success rate: the average percentage of object picking success per completion.
- Action efficiency: a ratio of the ideal to the actual number of actions taken to complete the given task.

C. Simulation Experiments

We design three manipulation tasks to evaluate our method in simulation: dense clutter removal, clearing 11 challenging test cases with adversarial objects, and stacking multiple objects. All tasks share the same MDP formulation in section III, while the object set, manipulation action space, and the reward function are different for each task. Three sets of objects are used in these tasks: (i) 9 different 3D toy blocks (same as VPG [11]), (ii) 49 diverse evaluation objects from EGAD [32], and (iii) cubes only for stacking task. To train our models for all tasks, objects with randomly selected shapes and colors are dropped in front of the robot at the start of each experiment. The robot then automatically performs data collection by trial and error. For object removal-related tasks in simulation, objects are removed from the scene after a successful pick action. The environment is reset at the termination of the task.

Dense Clutter Removal Task: We first evaluate the proposed method in the simulated environment where 30 objects are randomly dropped in a bin. The goal of the agent is to remove all objects from the bin in front of the robot by
executing pushing or picking actions. We trained the agent with only 10 objects instead of 30 objects. This helps to test the generalization of policies to more cluttered scenarios. The first experiment compares the proposed method to previous methods in simulation using the adversarial objects from [11]. Comparison of our results with Grasping-only [33], P+G Reactive [11], VPG [11] and SPOT [12] are summarized in Fig. 4(a). We see that with a pick success rate of 95.8%, the proposed method outperforms all other methods across all metrics. We observe that our proposed method learned to pick items from a dense clutter without having to declutter them before picking. We hypothesize that this led to the high action efficiency of 94.4%. The second experiment was with even more complex objects from the validation set of EGAD [32]. We observed a high success rate of 92.2% during testing on these complex objects, which suggests good generalizability of our proposed method.

Challenging Arrangements Task: We also compare the proposed method with other methods in simulation on 11 challenging test cases with adversarial clutter from [11]. These test cases are manually engineered to reflect challenging picking scenarios which allow us to evaluate the model’s robustness. Each test case consists of a unique configuration of 3 to 6 objects placed in a tightly packed arrangement that will be challenging even for an optimal picking-only policy as it will require de-cluttering them before picking. As a sanity check, a single isolated object is additionally placed separately from the main configuration. Similar to the dense clutter removal task, we trained a policy that learns to push and pick with 10 objects randomly placed in a bin and tested it on 11 challenging test cases. Performance comparison with previous work is shown in Fig. 4(b). Across the collection of test cases, we observe that our proposed method can successfully solve 10/11 cases with a 100% completion rate and an overall completion rate of 98.6%. Moreover, the action efficiency of 84.2% with our method indicates that our method is significantly better than VPG [11] and SPOT [12] methods that attained action efficiencies of only 60.1% and 38% respectively.

Block Stacking Task: To truly evaluate our multi-step task learning method, we consider the task of stacking multiple blocks on top of each other. This constitutes a challenging multi-step robotic task as it requires the agent to acquire several core abilities: picking a block from 10 blocks arbitrarily placed in the bin, precisely placing it on top of the second block, and repeating this process until a goal stack height is reached. We performed multiple experiments with a goal stack height in the range of 2 to 5. Fig. 4(c) summarises the performance of our method compared to prior work. Jeong et al. used self-supervised domain adaptation (SSDA) [21] and Zhu et al. used deep visuomotor policy (DVP) [22] and tested it with a goal stack height of 2. Hundt et al. used SPOT [12] and tested it with a goal stack height of 4. For this task with the highest complexity, our approach seems to perform significantly better with the action efficiency of 98% and 86% for a goal stack height of 2 and 4, respectively. A possible reason is that, although the task is very complex, the TPG reward function helps learn optimal multi-step manipulation policy.

D. Real-World Experiments

We validate our method in the real-world on two tasks: mixed item bin picking and block stacking. Fig. 5 illustrates the two tasks being executed by the UR10 robot using RoManNet trained for 2k iterations (~4 hours of robot run time).

Mixed Item Bin Picking: For the mixed item bin-picking task, the robot picks items from the source bin and places them into a destination bin until the source bin is empty, and then swaps the source and destination bin for the next run. 30 different items were used for training the model. To automate the training process, suction feedback is used to detect a successful pick and a binary classifier is used to detect

![Fig. 4: Quantitative comparisons with prior work for various tasks in similar simulation setups. Results for DVP and SSDA methods are borrowed from respective papers. All other results are reproduced in simulation using the open source code and models. (a) Performance for Dense Clutter Removal Task in terms of pick success rate and action efficiency. (b) Performance for Challenging Arrangements in terms of mean completion rate and action efficiency. (c) Performance for block stacking task in terms of mean action efficiency for various goal stack heights.](image-url)
Fig. 5: Illustration of mixed item bin picking and block stacking tasks being executed by the UR10 robot using the model trained for approximately 4 hours in the real-world. We provide additional qualitative results in the attached video.

Fig. 6: Ablation studies. (a) Comparison of network parameters and compute time of RoManNet with prior work (b) Learning curves for ablation of techniques used in conjunction with RoManNet for training agent on block stacking task. Solid lines indicate mean action success rates and dotted lines indicate mean action efficiency over training steps.

if the bin is empty after each manipulation action. We observe that the robot adopts diverse strategies to clear the clutter. The performance results in Table I shows that RoManNet performs consistently better than all previous methods. Pick success rate of 96% after training for only 2k iterations demonstrates the high sample efficiency of our method. Generalization is a key index for applications in industrial and logistics automation. Our experiments with 10 unseen items show that our method is extremely impressive as it can generalize to novel objects in a real-world setting and still achieve a success rate of 92%.

Real-World Block Stacking: The real-world block stacking task is similar to the one in simulation, where the robot performs manipulation actions until the goal stack height is reached. The depth measurements from the overhead camera are used to determine the current stack height. This task is particularly challenging as the agent needs to learn to align the position and orientation of the picked item with the stack during placement. Remarkably, we observed a 100% completion rate and 84% action efficiency for a goal stack height of 4, outperforming the state-of-the-art action efficiency of 61% reported by Hundt et al. in [12].

E. Ablation Studies

In order to assess the necessity and efficacy of the different components of our proposed method, described in section III, we provide ablation experimental results. Fig. 6(a) shows a comparison of the number of network parameters and compute time for calculating the Q-values for a pushing and placing task of RoManNet with network architectures used in VPG [11] and SPOT [12]. As RoManNet has significantly fewer network parameters, the compute time is significantly smaller (0.6 sec), which makes RoManNet more suitable for real-world manipulation tasks. Fig. 6(b) compares the learning curves for the underlying algorithm against the baseline approaches for the complex multi-step task of block stacking.

RoManNet Baseline: We establish a baseline using the primary reward function found in VPG [11] and ϵ-greedy exploration policy. We define the baseline reward function as function of sub-task indicator only, i.e $R_{base}(s_{t+1}, a_t) = \lambda'(s_{t+1}, a_t)$. The only case where this baseline model is on par with the full model is the clutter removal task, in which both the baseline and the full model achieved similar levels of performance. We hypothesize that this is due to the short length of the task, where the task progress-based reward did not play a significant role.

RoManNet + TPGR: The second baseline is a combination of the TPG reward function and ϵ-greedy exploration policy. We observe that TPG reward helped the agent learn to avoid task progress reversal for the multi-step block stacking task, thus improving the success rate by more than 25%. We also observed a significant improvement in action efficiency for the challenging arrangements and block stacking tasks. Moreover, we observed that anisotropic gaussian smoothing improved manipulation action stability by removing maxima that were close to regions of low Q values. For example, RoManNet generated more stable action poses when trained with $R_{tpgr}$ as compared to $R_{tp}$. As seen in Fig. 6(b), training with $R_{tpgr}$ reward function improved the action success rate by 9% compared to training with $R_{tp}$ for block stacking task.

RoManNet + TPGR + LAE: For the complete proposed model, the proposed LAE policy is used instead of the ϵ-greedy exploration. We observe an 11% improvement in action efficiency for the block stacking task when RoManNet is trained with LAE policy instead of the baseline ϵ-greedy policy. This suggests that LAE policy improves the efficiency of exploration, which is critical in real-world applications as training for a large number of iterations on a physical robot is expensive.
V. CONCLUSIONS

In this work, we present a vision-based deep reinforcement learning framework to effectively learn complex manipulation tasks that consist of multiple steps and long-horizon planning. The experimental results indicate that our TPG reward which computes reward based on the actions that lead to successful motion primitives and progress toward the overall task goal can successfully handle progress reversal in multi-step tasks. Moreover, we showed that our LAE policy can be used to curb unnecessary exploration which can occur after the initial states have already been explored. Compared to the previous work in multi-step manipulation, empirical results demonstrate that our method outperformed all previous methods in various multi-step tasks in simulation as well as real-world settings. We demonstrate a high generalization ability of our method by showing that our method achieves state-of-the-art results in performing multi-step tasks on previously unseen objects.

REFERENCES

[1] J. Bohg, A. Morales, T. Asfour, and D. Kragic, “Data-driven grasp synthesis—a survey,” IEEE Transactions on Robotics, vol. 30, no. 2, pp. 289–309, 2014.
[2] M. Kopicki, R. Detry, M. Adjibigle, R. Stolkin, A. Leonardis, and J. L. Wyatt, “One-shot learning and generation of dexterous grasps for novel objects,” The International Journal of Robotics Research, vol. 35, no. 8, pp. 959–976, 2016.
[3] P. Schmidt, N. Vahrenkamp, M. Wächter, and T. Asfour, “Grasping of unknown objects using deep convolutional neural networks based on depth images,” in 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 6831–6838.
[4] S. Kumra and C. Kanan, “Robotic grasp detection using deep convolutional neural networks,” in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017, pp. 769–776.
[5] L. Yin-Chen, A. Zeng, S. Song, P. Isola, and T.-Y. Lin, “Learning to see before learning to act: Visual pre-training for manipulation,” in 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020, pp. 7286–7293.
[6] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection,” The International Journal of Robotics Research, vol. 37, no. 4-5, pp. 421–436, 2018.
[7] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Otto, M. Kalakrishnan, V. Vanhoucke et al., “Scalable deep reinforcement learning for vision-based robotic manipulation,” in Conference on Robot Learning, 2018, pp. 651–673.
[8] S. Joshi, S. Kumra, and F. Sahin, “Robotic grasping using deep reinforcement learning,” in 2020 IEEE 16th International Conference on Autonomous Science and Engineering (CASE). IEEE, 2020, pp. 1461–1466.
[9] P. Florence, L. Manuelli, and R. Tedrake, “Self-supervised correspondence in visuomotor policy learning,” IEEE Robotics and Automation Letters, vol. 5, no. 2, pp. 492–499, 2019.
[10] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, “Learning complex dexterous manipulation with deep reinforcement learning and demonstrations,” in Proceedings of Robotics: Science and Systems, Pittsburgh, Pennsylvania, June 2018.
[11] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, “Learning synergies between pushing and grasping with self-supervised deep reinforcement learning,” in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018, pp. 4238–4245.
[12] A. Hundt, B. Killeen, N. Greene, H. Wu, H. Kwon, C. Paxton, and G. D. Hager, “‘good robot!’: Efficient reinforcement learning for multi-step visual tasks with sim to real transfer,” IEEE Robotics and Automation Letters, vol. 5, no. 4, pp. 6724–6731, 2020.
[13] L. Berscheid, P. Meffert, and T. Kröger, “Robot learning of shifting objects for grasping in cluttered environments,” in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019, pp. 612–618.
[14] I. Lenz, H. Lee, and A. Saxena, “Deep learning for detecting robotic grasps,” The International Journal of Robotics Research, vol. 34, no. 4-5, pp. 705–724, 2015.
[15] L. Pinto and A. Gupta, “Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours,” in 2016 IEEE International Conference on Robots and Automation (ICRA). IEEE, 2016, pp. 3406–3413.
[16] S. Kumra, S. Joshi, and F. Sahin, “Antipodal robotic grasping using generative residual convolutional neural network,” in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020.
[17] M. Riedmiller, R. Hafner, T. Lampe, M. Neuen, J. Degrave, T. Wiele, V. T. T. Heess, and T. J. Springenberg, “Learning by playing solving sparse reward tasks from scratch,” in International Conference on Machine Learning, PMLR, 2018, pp. 4344–4353.
[18] D. Xu, A. Mandlekar, R. Martín-Martín, Y. Zhu, S. Savarese, and L. Fei-Fei, “Deep affordance foresight: Planning through what can be done in the future,” in 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021.
[19] I. Nematollahi, O. Mees, L. Hermann, and W. Burgard, "Hindsight for foresight: Unsupervised structured dynamics models from physical interaction," in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 5319–5326.
[20] J. Mahler and K. Goldberg, “Learning deep policies for robot bin picking by simulating robot grasping sequences,” in Conference on Robot Learning. PMLR, 2017, pp. 515–524.
[21] R. Jeong, Y. Aytar, D. Kosmid, Y. Zhou, J. Kay, T. Lampe, K. Bousmalis, and F. Nori, “Self-supervised sim-to-real adaptation for visual robotic manipulation,” in 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020, pp. 2718–2724.
[22] Y. Zha, Z. Wang, J. Merel, A. Rusu, T. Perez, S. Cabi, S. Tunsuyuva, C. Kramar, R. Hadsell, N. de Freitas et al., “Reinforcement and imitation learning for diverse visuomotor skills,” Robotics: Science and Systems XIV, vol. 14, 2018.
[23] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, and J. Lee, “Translator networks: Rearranging the visual world for robotic manipulation,” Conference on Robot Learning (CoRL), 2020.
[24] K. Kase, C. Paxton, H. Mazar, T. Ogata, and D. Fox, “Transferable task execution from pixels through deep planning domain learning,” in 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020, pp. 10 459–10 465.
[25] D. Driess, J.-S. Ha, R. Tedrake, and M. Toussaint, “Learning geometric reasoning and control for long-horizontal tasks from visual input,” in 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021.
[26] K. Zakka, A. Zeng, J. Lee, and S. Song, “Form2fit: Learning shape priors for generalizable assembly from disassembly,” in Proceedings of the IEEE International Conference on Robotics and Automation, 2020.
[27] S. V. Luong, J. Ichnowski, M. Danieleczuk, D. S. Brown, A. Balakrishna, S. Joshi, E. Rocha, E. Solowjow, and K. Goldberg, “Kit-net: Self-supervised learning to kit novel 3d objects into novel 3d cavities,” in 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE). IEEE, 2021.
[28] J. Borràs, G. Alenya, and C. Torras, “A grasping-centered analysis for cloth manipulation,” IEEE Transactions on Robotics, vol. 36, no. 3, pp. 924–936, 2020.
[29] D. Seita, A. Ganapathi, R. Hoque, M. Huang, E. Cen, A. K. Tanwani, A. Balakrishna, B. Thananjeyan, J. Ichnowski, N. Jamali et al., “Deep imitation learning of sequential fabric smoothening from an algorithmic supervisor,” 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).
[30] C. Tsotsios and M. Petrou, “On the choice of the parameters for anisotropic diffusion in image processing,” Pattern recognition, vol. 46, no. 5, pp. 1369–1381, 2013.
[31] M. Tomik, “Adaptive $\varepsilon$-greedy exploration in reinforcement learning based on value differences,” in Annual Conference on Artificial Intelligence Springer, 2010, pp. 203–210.
[32] D. Morrison, P. Corke, and J. Leitner, “Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation,” IEEE Robotics and Automation Letters, vol. 5, no. 3, pp. 4368–4375, 2020.