“Good Robot!”:
Efficient Reinforcement Learning for Multi-Step Visual Tasks via Reward Shaping

Andrew Hundt¹, Benjamin Killeen¹, Heeyeon Kwon¹, Chris Paxton², and Gregory D. Hager¹

Abstract—In order to learn effectively, robots must be able to extract the intangible context by which task progress and mistakes are defined. In the domain of reinforcement learning, much of this information is provided by the reward function. Hence, reward shaping is a necessary part of how we can achieve state-of-the-art results on complex, multi-step tasks. However, comparatively little work has examined how reward shaping should be done so that it captures task context, particularly in scenarios where the task is long-horizon and failure is highly consequential. Our Schedule for Positive Task (SPOT) reward trains our Efficient Visual Task (EVT) model to solve problems that require an understanding of both task context and workspace constraints of multi-step block arrangement tasks. In simulation EVT can completely clear adversarial arrangements of objects by pushing and grasping in 99% of cases vs an 82% baseline in prior work. For random arrangements EVT clears 100% of test cases at 86% action efficiency vs 61% efficiency in prior work. EVT + SPOT is also able to demonstrate context understanding and complete stacks in 74% of trials compared to a baseline of 5% with EVT alone. To our knowledge, this is the first instance of a Reinforcement Learning based algorithm successfully completing such a challenge. Code is available at https://github.com/jhu-lcsr/good_robot.

I. INTRODUCTION

Multi-step tasks pose a significant challenge for robots, both because of their complexity and because it can be very easy to undo progress. Historically, this challenge required the use of a task planner, which represents the constraints of a task and searches for an efficient solution. Similarly, learning multi-step tasks requires an agent to understand each action within the context of a larger goal, but with limited knowledge of the state space. Whereas a motion planner constrains its search space according to a priori knowledge about the task space, an agent must learn whether each action contributes toward the end goal or reverses previous progress. We explore mechanisms for efficiently propagating this context information via Deep Reinforcement Learning (DRL) within the domain of block arrangement tasks.

We use the term context to denote information necessary to advance progress in a block stacking task (Fig. 1). For example, a learning algorithm has to discover that grasping a block from an existing stack is unproductive, and imprecisely grasping or placing a block entails a risk of toppling the stack. Likewise, an agent must learn about workspace constraints and develop and apply complex manipulation skills such as pushing or reorienting blocks (Fig. 2).

1Johns Hopkins University. {ahundt, killeen, hwon28, ghager1} @jhu.edu
2NVIDIA. cpaxton@nvidia.com

Discovering and effectively applying contextual knowledge is nontrivial, so behavior which demonstrates an understanding of context should be rewarded. Our proposed SPOT reward schedule (Sec. III-B) takes inspiration from a humane and effective approach to training pets sometimes called “Positive Conditioning”. Consider the goal of training a dog “Spot” to ignore an object or event she finds particularly interesting on command. In this practice, Spot is rewarded with treats whenever partial compliance with the desired end behavior is shown, and simply removed from harmful or regressive situations with zero treats (reward). One way to achieve this is to start with multiple treats in hand, place one treat in view of Spot, and if she eagerly jumps at the treat (a negative action) the human snatches and hides the treat immediately for zero reward on that action. With repetition, Spot will eventually hesitate, and so she is immediately praised with “Good Spot!” and gets a treat separate from the one she should ignore. This approach can be expanded to new situations and behaviors, plus it encourages exploration and rapid improvement once an initial partial success is achieved. As we describe in more detail below, our SPOT reward is likewise designed to provide neither reward nor punishment for actions which reverse progress.

No reward will be effective if the agent is unable to learn the task at hand in a reasonable amount of time. Thus, learning system design (in our case neural network design) goes hand in hand with reward design. In this work,
we introduce a novel Efficient Visual Task (EVT) deep convolutional architecture for perception-based manipulation tasks. Furthermore, we demonstrate a novel Schedule for Positive Task (SPOT) reward enabling new capabilities for multi-step robotic tasks with Deep Reinforcement Learning.

In summary, our contributions in this article are: (1) EVT, an efficient and accurate network model for visual tasks; (2) Reinforcement Learning of Multi-step robotic tasks combining model-based low level control, model-free learning of high level goals, and a progressive reward schedule; and (3) The Schedule for Positive Task (SPOT) reward for long-horizon robotic tasks. Combining the above enables DRL to train a network on “hard” multi-step tasks in a reasonable number of iterations.

Our grasping and placing Action Efficiency Error (Sec. IV-A) is 47% of the error found in previous work[1], while utilizing 49% of the computational resources. More importantly, we are able to complete long term multi-step tasks which are, to our knowledge, not viable with existing baseline algorithms.

II. RELATED WORK

The advent of deep learning has introduced novel approaches to robotic manipulation tasks like pushing and grasping. Within this space, our work confronts fundamental problems that arise in multi-step tasks like stacking or arranging blocks. We review these areas here in brief.

Deep Neural Networks (DNNs) in particular have enabled the use of raw images in robotic manipulation. In some approaches, the DNN’s output directly corresponds to motor commands, e.g. [2], [3]. Higher level methods, on the other hand, assume a simple model for robotic control and focus on bounding box or pose detection for downstream grasp planning [4], [5], [6], [7], [8], [9], [10], [11]. Increasingly, these methods benefit from the depth information provided by RGB-D sensors [1], [6], [11], which capture physical information about the workspace. However, the agent must still develop physical intuition, which recent work attempts in a more targeted setting. [12], [13], for instance, focus on block stacking by classifying simulated stacks as stable or likely to fall. Of these, the ShapeStacks dataset [13] includes a larger variety of objects such as cylinders and spheres, as opposed to blocks alone. Similarly, [14], [15] develop physical intuition by predicting push action outcomes. Our work diverges from these approaches by developing visual understanding and physical intuition simultaneously, in concert with understanding progress in multi-step tasks.

When paired with DNNs, reinforcement learning has proven effective at increasingly complex tasks in robotic manipulation. An early approach in this space [2] directly coordinates RGB vision with servo motor control, learning tasks like unscrewing a bottle or using a hammer. Other methods focus on transferring visuo-motor skills from simulated to real robots [16], [17]. Our work directs a low-level controller to perform actions rather than regressing torque vectors directly, following [1], [18] by learning a pixel-wise success likelihood map.

Notably, VPG[1] is state of the art for RL-based table clearing tasks which can be trained within hours on a single robot from images. It is frequently able to complete adversarial scenarios like those in Fig. 4, by first pushing a tightly packed group and then grasping the now-separated objects. We utilize the VPG V-REP simulation and models as our baseline for comparison. VPG assumes an instantaneous reward delivery is sufficient to complete the task at hand. By contrast, we tackle multi-step tasks with sparse rewards which cannot be represented by VPG.

Multi-step tasks with sparse rewards present a challenge in reinforcement learning generally because they are less likely to be discovered through random exploration. This suggests demonstration is an effective method for guiding exploration [19], [20]. Within robotic manipulation, one approach separates a multi-step task into many modular sub-tasks comprising a sketch [21], while another separates the learning architecture into robot- and task-specific modules [22]. Our SPOT Reward, meanwhile, combines a novel reward schedule with a frequent reset policy to make early successes both more likely as well as more instructive.

Finally, neural architecture search forms the basis for our hyperparameter choices[23], [24]. Neural networks are imperfect arbitrary function approximators, so a better choice of algorithm is an effective approach to improving deep learning based robotic manipulation algorithms, as we have detailed in past work[25].

III. APPROACH

We formulate the problem of visual picking, pushing, and placing to complete a structure as a Partially Observable Markov Decision Process (POMDP) \((S, O, A, P, R)\), with state space \(S\), observation space \(O\), action space \(A\), transition \(P\), and reward function \(R\). At every time step \(t\), the robot chooses an action to take according to its policy \(\pi(o_t) \rightarrow a_t\), which results in a state transition to \(s_{t+1}\).

As in VPG [1], our goal is to learn a deterministic policy \(\pi(s_t)\) via Q-learning, which chooses the action \(a_t\) at every \(t\) such that \(a_t = \arg \max_a Q(s_t, a)\). In our case, as in past work, we make the simplifying assumption that state can be
identified from a single observation, in which case we can instead frame this as an MDP over observations \( O \). Formally, we learn by iteratively minimizing the temporal difference error \( \delta_t \) of \( Q(o_t, a_t) \) to a target \( y_t \), where:

\[
\delta_t = ||Q(o_t, a_t) - y_t|| \\
a_{t+1}^* = \arg\max_{a \in A} Q(o_{t+1}, a) \\
y_t = R(o_{t+1}, a_t) + \gamma Q(o_{t+1}, a_{t+1}^*)
\]

In our case, \( O \) consists of RGB-D heightmap images, as shown in Fig. 3. We capture these from a fixed camera, which are first converted into a point cloud and then projected so that \( z \) is aligned with the direction of gravity. As per [1], our height maps cover an 0.448 m\(^2\) table space. Images have a resolution of 224 × 224, meaning that each pixel represents roughly 2 mm\(^2\).

Our actions are represented as a 4-tuple \( a = \{\phi, x, y, \theta\} \). We will make two simplifying assumptions in order to handle pick and place tasks for assembly. First, we divide actions into set of high-level motion primitives \( \phi_t \), where \( \phi_t \in \{\text{grasp, push, place}\} \). \( x \) and \( y \) represent 2D planar coordinates that parameterize this action primitive, and \( \theta \) is an angle at which to position the gripper. Each \( \phi \) also defines the appropriate gripper behavior. In practice we use \( k = 16 \) discrete values for \( \theta \) by passing multiple discrete rotations of the input heightmap into \( Q(o_t, a_t) \), again following VPG [1].

### A. Network Architecture

Fig. 3 shows our overall Efficient Visual Task (EVT) solution, including choice of actions, how the robot acts on the world, and the model architecture. EVT utilizes 2 EfficientNet-B0 models, one for grasping and placing, with weights shared between color and depth data. We modify the EfficientNet-B0 [23] pretrained on ImageNet to an FCN [26] by loading the final stride 2 convolution as a stride 1 convolution with a dilation rate of 2. This dilated convolution ensures more fine grained action choices are possible by doubling the effective \((x, y)\) output resolution of the network.

The final dense grasp, push, and place blocks consist of a batchnorm and relu before each of two 1x1 convolutions with
2560 channels, i.e. [bn, relu, conv1x1, bn, relu, conv1x1], where a 1x1 convolution is equivalent to a dense layer at each pixel. These parameters are based on the final dense block structure optimized for accuracy via HyperTree Architecture Search [25] in our prior work. We note that efficiency was not considered in the HyperTree metric and as a result this pixel-wise dense block accounts for over 50% of the computation in EVT, so it is a good target for future efficiency gains. The push and grasp EVT architecture consumes 46 B FLOPS and holds 24 M parameters. The added dense block for placement actions brings this to 57 B FLOPS and 30 M parameters.

In our experiments we compare with VPG [1] which is the current state of the art DRL algorithm for clearing objects from a surface. It incorporates 4 separate DenseNet-121 models for grasp RGB, grasp Depth, push RGB, and push Depth with a total of 94 B floating point add-multiply operations (FLOPS). A push, grasp, and place configuration holds 24 M parameters. The added dense block for placement actions for stacking is evaluated more specifically by the placement action efficiency, and task completion rate. The completion rate is defined as the percentage of trials where the policy is able to successfully complete a task before an episode ends and in accordance with the metrics found in VPG[1]. These include the percentage of successful grasps, placement action efficiency, and task completion rate. The completion rate is defined as the percentage of trials where the policy is able to successfully complete a task before the grasp or push action fails 10 consecutive times. Success of a push is when more than 300 pixels have changed in a scene. A successful grasp is counted when the gripper is in a partially open state after executing the open loop action, indicating an object is present in the gripper, and the closed gripper state indicates grasp failure. A successful place for stacking is evaluated more specifically by the z height of object origins when object poses are known.

Our Schedule for Positive Task (SPOT) reward has two components: a linearly increasing sub-task “partial compliance” reward is delivered for actions which make progress on the task and a reward of 0 is delivered for actions which result in a reversal of progress. Formally, each reward $R$ is computed from $k$ different sub-tasks $[1..k] \in S_K$ with a number associated with the current active sub-task $S_{cur}$. Examples of possible values for $S_{cur}$ include one of [grasp=1, push=2, place=3] at a specific action $i$ in the sequence of actions during the trial, so $S_{cur}$ varies depending on the action taken at a given time $i$, and $S_{max}$ is max($S$). A current task progress depth $T_{cur} = [1..d] \in T_D$ indicates linear progress through a task, such as stack height. The range of possible reward values is defined between $[\eta_{min}, \eta_{max}]$. First, we wish to expand the skill set, and each mastered skill provides exponential increases in overall capability. For this reason we ensure rewards grow faster as more parts of a curriculum are mastered in the Positive Reward (PR) Component:

$$R_{PR,i} = \eta_{min} + (\eta_{max} - \eta_{min}) \frac{(\frac{T_{cur,i}}{T_{max}}) p_{\text{compl}} - 1}{p - 1} \quad (2)$$

Second, as in our story of Spot the dog (Sec. I), we also wish to minimize disincentive for exploration without rewarding mistakes. This second component is called Situation Removal (SR), and is applied separately at each action time step:

$$R_{SR,i} = \begin{cases} R_{PR,i} & \text{if } T_{cur,i} \geq T_{prev,i-1} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Here $R_{PR,i}$ is from eq. 2 and subtask rewards $s_i \in S$ are given if the “partial compliance” subtask is successfully completed, and 0 otherwise. For stacking tasks we choose $p = 2$ and the actions for which subtask rewards are delivered include $S = \{\text{none}, \text{successful scene change}, \text{grasp success}, \text{place success}\}$ with values $[0, 1, 2, 3]$. Task depth $T_{cur}$ is the stack height with possible values of $[1, 2, 3, 4]$. In addition to the instantaneous version of SPOT above, we also define a recursive SPOT Trial Reward for use during experience replay of previous full trials:

$$R_{SRT,i} = \begin{cases} R_{SR,i} + \gamma R_{SR,i+1} & \text{if final timestep} \\ R_{SR,i} + \gamma R_{SR,i+1} + \gamma R_{SR,i+2} & \text{if } R_{SR,i} > 0 \end{cases} \quad (4)$$

These values are recursively rolled out from the final action to the first action of a trial. The effect of this trial reward is that future rewards only propagate across time steps where subtasks are completed successfully.

IV. Experiments

We conducted simulated experiments in the two baseline scenarios provided by VPG [1] with the same simulator and settings, as well as in two multi-step tasks of our own design. We present the results of these experiments in Table II, with descriptions and analysis below.

A. Evaluation Metrics

We evaluate our algorithms in test cases with new random seeds and in accordance with the metrics found in VPG[1]. These include the percentage of successful grasps, placement action efficiency, and task completion rate. The completion rate is defined as the percentage of trials where the policy is able to successfully complete a task before the grasp or push action fails 10 consecutive times. Success of a push is when more than 300 pixels have changed in a scene. A successful grasp is counted when the gripper is in a partially open state after executing the open loop action, indicating an object is present in the gripper, and the closed gripper state indicates grasp failure. A successful place for stacking is evaluated more specifically by the z height of object origins when object poses are known.
TABLE I: Pushing and grasping results. The first task is to clear 10 toys for 100 trials with random arrangements, and the second is to clear 10 toys each across 11 adversarial arrangements with 110 total trials. Bold entries highlight our key algorithm improvements over the baseline.

TABLE II: Multi-Step task test success rates measured out of 100% for simulated tasks involving push, grasp and place actions. A lower Floating Point Operations (FLOPS) count indicates improved neural network computational efficiency. Bold entries highlight our key algorithm improvements over the baseline.

Alternately it can be awarded when height the highest vertical z height of a scene has increased by a minimum threshold. Ideal Action Efficiency is 100% and calculated as the $\text{IdealActionCount}/\text{ActualActionCount}$. The Ideal Action Count is 1 action per object for grasping tasks, and for tasks which involve placement it is against 2 actions per object, where one object is assumed to remain stationary. This means 6 total actions for a stack of height 4 since only 3 objects must move.

B. Baseline Scenarios

**Clear 10 Toys:** We establish a baseline via the primary simulated experiment found in VPG [1], where 10 toys with varied shapes must be grasped to clear the robot workspace. EVT reduces the Action Efficiency Error ($1 - \text{Action Efficiency}$) from 39% with VPG to 14% with EVT. We validate on 100 trials of 10 novel random object positions, where $\gamma = 0.5$.

Table I(top) shows the results. As this is a fairly straightforward task, all methods were eventually able to complete it in the basic case, but we see that our proposed EVT model has the highest grasp success rate. In this case, we can also see SPOT does not make a meaningful difference – which makes intuitive sense, as the task structure is very simple.

**Clear Toys Adversarial:** Our second more challenging baseline scenario (Fig. 5) contains 11 adversarial arrangements from prior work [1] where toys are placed in tightly packed configurations. We use the pretrained weights from the “Clear 10 Toys” task in scenarios the algorithm has never previously seen. The evaluation algorithm defined by VPG[1] is designed such that when no successful grasp is made for 10 consecutive actions the task is considered incomplete. We validate on 10 trials for each of the 11 challenge arrangements, and our model was able to clear the scene in 109 out of 110 trials. Table I(bottom) details the results: EVT performed substantially better. In EVT’s lone failure case the model had successfully separated the tightly packed blocks on the final 10th action without a successful grasp. It is reasonable to expect it would have finished clearing the scene in the next few actions had it not hit the incomplete task limit.

Curiously, while the rate of Tasks Completed rises, Action Efficiency is reduced in these challenging scenarios. Subjectively, this is due to the higher priority placed on grasping when compared to pushing as EVT attempts grasps in 89% of actions in these scenarios on average. In many cases the algorithm attempts a push only after several failed attempts at grasping, which finally frees up the blocks to complete the task, while in other cases it can separate blocks using
grasps alone.

C. Multi-Step Task Scenarios

We attempted to evaluate a direct extension of the VPG algorithm in which we simply add new DenseNet-121 place action models for RGB and Depth but the architecture exceeds the memory limits of our GTX 2080Ti GPUs (and the Titan X GPUs from VPG), illustrating limitations in the scalability of VPG when compared to our EVT architecture for new tasks.

Stack 4 Cubes: Our primary test task is to stack 4 cubes randomly placed within the scene. We ensure that workspace constraints are strictly observed by deeming any action in which a partial stack the robot has already assembled is subsequently toppled returns a reward of 0 and immediately ends the trial with the failure condition. This strict progress evaluation criteria ensures the scores indicate an understanding of the context surrounding the stack. A block can also occasionally tumble out of the workspace after a missed place action, which leaves no opportunity for recovery.

EVT is evaluated under the basic exponential reward schedule (eq. 1) and our SPOT Reward given in eq. 2, which accounts for task progress. Table II shows the results. As evidenced by the huge difference between SPOT and the baseline reward schedule, the SPOT Reward proves essential to task completion. EVT trained with SPOT succeeds 74% of the time, versus only 5% without it. This is because it is impossible to differentiate between placing one block on another single block vs stack of height 2 with the exponential reward curve.

Row of 4 Cubes: Our third test task evaluates the ability of the algorithm to generalize across tasks. Curiously, while making a row of 4 blocks appears the same as stacking, it is in fact much more difficult to train to complete efficiently. In particular, whereas with stacking optimal placement occurs on top of a strong visual feature—another block—the arrangement of blocks in rows depends on non-local visual features, i.e. the rest of the row. Additionally, every block in each row is available for grasping, which may reverse progress, as opposed to stacks where only the top block is readily available. This requires significant understanding of context, as we have described it, to accomplish. Table II shows significant progress on this challenging environment, succeeding 92% of the time. The higher overall task completion rate for rows when compared to stacks is in part due to reduced risk of a block tumbling out of the robot workspace.

V. Conclusion

In spite of its obvious importance to results in reinforcement learning, reward shaping is one of the most under-explored areas in deep learning research for robotics. We have demonstrated an effective approach for training long-horizon tasks which present a high risk for reversing progress: the SPOT reward. To our knowledge, this is the first instance of reinforcement learning applied toward such a challenge. First, our EVT neural network model far exceeds existing methods’ computational efficiency for manipulation tasks while simultaneously providing a 20% increase in action efficiency and a 15% higher perfect completion rate for adversarial pushing and grasping scenarios. Our results show the continued importance of neural network architecture design choices for Robotics and Reinforcement Learning algorithms. Second, our SPOT Reward quantifies an agent’s progress within multi-step tasks while also providing zero-reward guidance, which we found necessary to achieve a 74% completion rate on a block stacking task and a 92% completion rate in a row creation task. Our work does assume some mechanism for intermediate rewards. Related methods which could help relax this assumption include inverse reinforcement learning of reward signals necessary to complete tasks, learning from demonstration, and metalearning. We expect that recent reinforcement learning algorithms beyond a Q function could also improve the efficiency of our algorithm. Nonetheless, we believe these principles are an effective approach for efficient learning on complex multi-step problems.

Finally, we note that our simulation executes in real time to ensure the viability of task transfer to a real robot as they did in the baseline VPG [1] experiments. It is our hope to demonstrate similar results on a physical testbed in the near future.

Acknowledgment

This material is based upon work supported by the NSF NRI Grant Award #1637949.

References

[1] 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.
[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” The Journal of Machine Learning Research, vol. 17, no. 1, pp. 1334–1373, 2016.
[3] S. Levine, P. Pastor, A. Krizhevsky, I. Barz, 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, dataset:https://sites.google.com/site/brainrobotdata/home. [Online]. Available: https://doi.org/10.1177/0278364917710318
