Machine Learning with Robotic Manipulation Papers

Jacob Taylor Cassady

The following papers attempt to present frameworks for using deep neural networks and reinforcement learning to perform robotic manipulation tasks not explicitly trained on by the robot. The first paper, Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates, presents an algorithm for learning tasks such as door opening, bin picking, and random reaching. The second paper, Learning Deep Policies for Robot Bin Picking by Simulating Robot Grasping Sequences, purposes a Partially Observable Markov Decision Process for bin picking from cluttered bins. The third paper, QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation, presents a general formulation of robotic manipulation as a Markov Decision Process to perform bin picking from cluttered bins. The fourth and final paper, Segmenting Unknown 3D Objects from Real Depth Images using Mask R-CNN Trained on Synthetic Data, attempts to show that Mask R-CNN networks can be trained using synthetic depth images and hopes to demonstrate its usefulness in a robotics application through grasping.

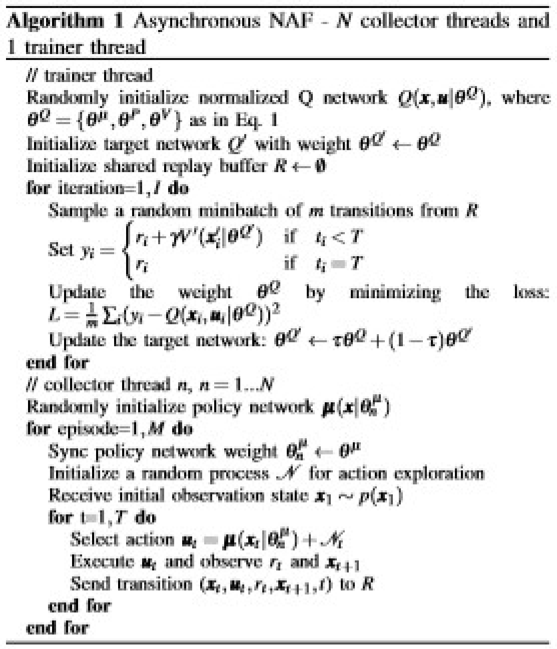
1 Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates (Google [2016])

1.1 Goals

This paper aims to demonstrate that deep reinforcement learning can learn policies efficiently enough to train on real physical robots. Additionally, they aim to reduce the algorithm's training time by training multiple robots which pool their policy updates asynchronously. Furthermore, they hope these robots will be able to learn complex manipulation policies without user-provided demonstrations using only neural network representations that do not require task-specific domain knowledge.

1.2 Methods

The goal of their algorithms is to find the optimal policy that maximizes the expected sum of returns from the initial state distribution. They investigate two Q-function based methods: Deep Deterministic Policy Gradient (DDPG) and Normalized Advantage Functions (NAF) in simulation. They then choose the best performing algorithm, NAF, to test on real physical systems. To test their algorithm, they perform random target reaching, door pushing & pulling, and pick & place in simulation. On real robots they run experiments performing random target reaching and door opening.

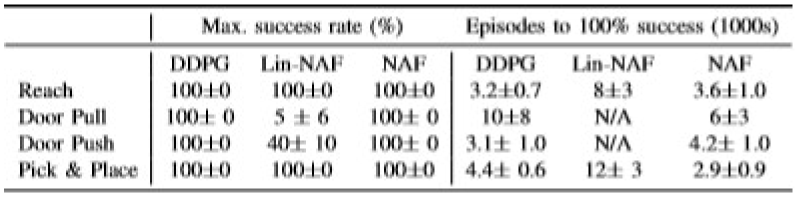


```
Algorithm 1 Asynchronous NAF - $N$ collector threads and 1 trainer thread
  // trainer thread
  Randomly initialize normalized Q network $Q(\boldsymbol{x},\boldsymbol{u}|\theta^Q)$, where
  $\theta^Q = \{\theta^\mu, \theta^P, \theta^V\}$ as in Eq. 1
  Initialize target network $Q'$ with weight $\theta^{Q'} \leftarrow \theta^Q$
  Initialize shared replay buffer $R \leftarrow \emptyset$
  for iteration=1, $I$ do
     Sample a random minibatch of $m$ transitions from $R$
     Set $y_i = \begin{cases} r_i + \gamma V'(\boldsymbol{x}'_i|\theta^{Q'}) & \text{if } t_i < T \\ r_i & \text{if } t_i = T \end{cases}$
     Update the weight $\theta^Q$ by minimizing the loss:
     $L = \frac{1}{m}\sum_i (y_i - Q(\boldsymbol{x}_i, \boldsymbol{u}_i|\theta^Q))^2$
     Update the target network: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
  end for
  // collector thread $n$, $n = 1...N$
  Randomly initialize policy network $\boldsymbol{\mu}(\boldsymbol{x}|\theta^\mu_n)$
  for episode=1, $M$ do
     Sync policy network weight $\theta^\mu_n \leftarrow \theta^\mu$
     Initialize a random process $\mathcal{N}$ for action exploration
     Receive initial observation state $\boldsymbol{x}_1 \sim p(\boldsymbol{x}_1)$
     for t=1, $T$ do
        Select action $\boldsymbol{u}_t = \boldsymbol{\mu}(\boldsymbol{x}_t|\theta^\mu_n) + \mathcal{N}_t$
        Execute $\boldsymbol{u}_t$ and observe $r_t$ and $\boldsymbol{x}_{t+1}$
        Send transition $(\boldsymbol{x}_t, \boldsymbol{u}_t, r_t, \boldsymbol{x}_{t+1}, t)$ to $R$
     end for
  end for
```

Figure 1 : Asynchronous Learning Algorithm

1.3 Conclusions

The simulated experiments showed that both DDPG and NAF could learn reach, door push/pull and pick & place in a simulated environment. Please see **Figure 2** below.



	Max. success rate (%)			Episodes to 100% success (1000s)		
	DDPG	Lin-NAF	NAF	DDPG	Lin-NAF	NAF
Reach	100±0	100±0	100±0	3.2±0.7	8±3	3.6±1.0
Door Pull	100± 0	5 ± 6	100± 0	10±8	N/A	6±3
Door Push	100±0	40± 10	100± 0	3.1± 1.0	N/A	4.2± 1.0
Pick & Place	100±0	100±0	100±0	4.4± 0.6	12± 3	2.9±0.9

Figure 2 : Simulation Experimental Results

They conclude having two or four workers attempting a task significantly improves the learning speed over 1 worker; although, there are significant diminishing returns past 2 workers. They found that two workers learning simultaneously could achieve a 100% success rate of door opening evaluated across 20 consecutive trials after 2.5 hours of training while one worker required 4 hours to achieve the 100% success rate.

2 Learning Deep Policies for Robot Bin Picking by Simulating Robust Grasping Sequences (Ken Goldburg [2017])

2.1 Goals

This paper attempts to present a method for universal bin picking from cluttered heaps of objects using parallel-jaw grippers by purposing a Partially Observed Markov Decision Process (POMDP) model. They hope to train a GQ-CNN to predict grasps with a high reward through training in simulations.

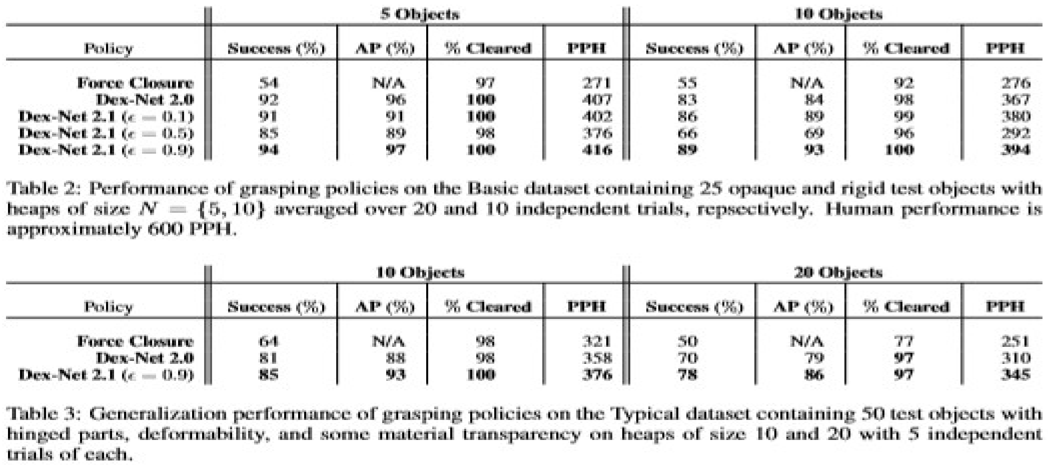
2.2 Methods

They use a pybullet dynamic simulator to develop their test environment. In this simulator they uniformly sample *m* 3D CAD object models and drop them into the environment with random poses. They then generate demonstrations of the robot grasping using Dex-Net 2.0 and aggregate synthetic point cloud observations and collected rewards to form a dataset for training using imitation learning. The training data is then preprocessed by transforming the point cloud to align the grasp center and axis with the center pixel and middle row to improve classification performance.

Experiments were performed both in simulation and on a physical ABB YuMi. For physical experiments, a subset of N objects was randomly dropped into a bin; half of these objects were rigid and opaque while the other half have transparency, moving parts, or deformable material. The robot's goal was to clear the bin of all objects.

2.3 Conclusions

They find they can achieve 94% success rate and 96% average precision on heaps of 5-10 objects with their POMDP model. Furthermore, it achieves up to 416 successful picks per hour (PPH) and was shown to clear heaps of 10 objects in under 3 minutes.



	5 Objects				10 Objects			
Policy	Success (%)	AP (%)	% Cleared	PPH	Success (%)	AP (%)	% Cleared	PPH
Force Closure	54	N/A	97	271	55	N/A	92	276
Dex-Net 2.0	92	96	**100**	407	83	84	98	367
Dex-Net 2.1 ($\epsilon = 0.1$)	91	91	**100**	402	86	89	99	380
Dex-Net 2.1 ($\epsilon = 0.5$)	85	89	98	376	66	69	96	292
Dex-Net 2.1 ($\epsilon = 0.9$)	**94**	**97**	**100**	**416**	**89**	**93**	**100**	**394**

Table 2: Performance of grasping policies on the Basic dataset containing 25 opaque and rigid test objects with heaps of size $N = \{5, 10\}$ averaged over 20 and 10 independent trials, repsectively. Human performance is approximately 600 PPH.

	10 Objects				20 Objects			
Policy	Success (%)	AP (%)	% Cleared	PPH	Success (%)	AP (%)	% Cleared	PPH
Force Closure	64	N/A	98	321	50	N/A	77	251
Dex-Net 2.0	81	88	98	358	70	79	**97**	310
Dex-Net 2.1 ($\epsilon = 0.9$)	**85**	**93**	**100**	**376**	**78**	**86**	**97**	**345**

Table 3: Generalization performance of grasping policies on the Typical dataset containing 50 test objects with hinged parts, deformability, and some material transparency on heaps of size 10 and 20 with 5 independent trials of each.

Figure 3 : Experimental Results

3 QT-OPT: SCALABLE DEEP REINFORCEMENT LEARNING FOR VISION-BASED ROBOTIC MANIPULATION (GOOGLE [NOV 2018])

3.1 GOALS

This paper aims to present a scalable self-supervised vision-based reinforcement learning framework to train a neural network to perform general object grasping. Their goal is to develop a grasping system that can pick up unseen objects with both reliable and effective grasps based on a Markov Decision Process.

3.2 METHODS

They have seven real robots fitted with RGB cameras performing self-supervision and training of the reinforcement learning framework. Observations come from the camera and actions consist of Cartesian motion of the end-effector and opening/closing of the gripper. The reinforcement algorithm is based on a continuous-action generalization of Q-learning. The robot receives a reward when it successful lifts an object and no reward when it doesn’t.

They had the robots make grasp attempts at objects continuously. During training, they would replace objects that were successfully grasped into the same bin. Additionally, they ran tests where they would empty the bin out as the robot grasped objects, forcing it to focus on the tougher objects.

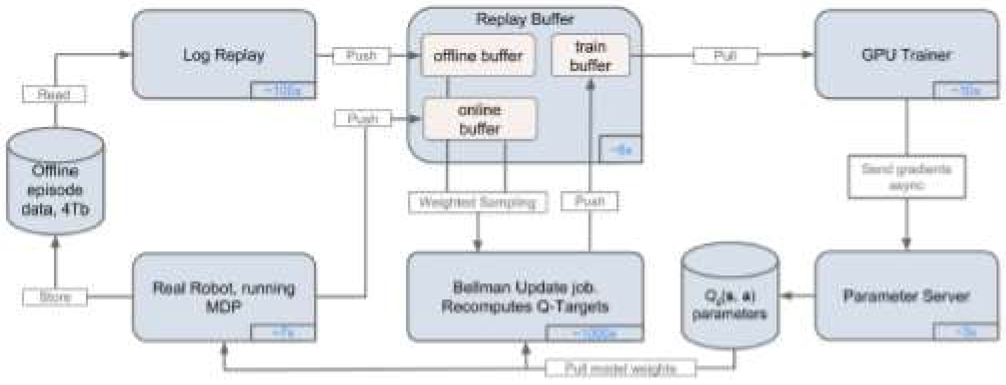


Figure 14: Architecture of the QT-Opt distributed reinforcement learning algorithm.

Figure 4 : Reinforcement Learning Algorithm Architecture

Algorithm 1 Grasping control-loop

```
 1: Pick a policy $policy$.
 2: Initialize a $robot$.
 3: while $step < N$ and not $terminate\_episode$ do
 4:     $\mathbf{s}$ = $robot$.CaptureState()
 5:     $\mathbf{a}$ = $policy$.SelectAction($\mathbf{s}$)
 6:     $robot$.ExecuteCommand($\mathbf{a}$)
 7:     $terminate\_episode = e$ {Termination action $e$ is either learned or decided heuristically.}
 8:     r = $robot$.ReceiveReward()
 9:     emit($\mathbf{s}$, $\mathbf{a}$, r)
10: end while
```

Figure 5 : Algorithm Implementation in Practice

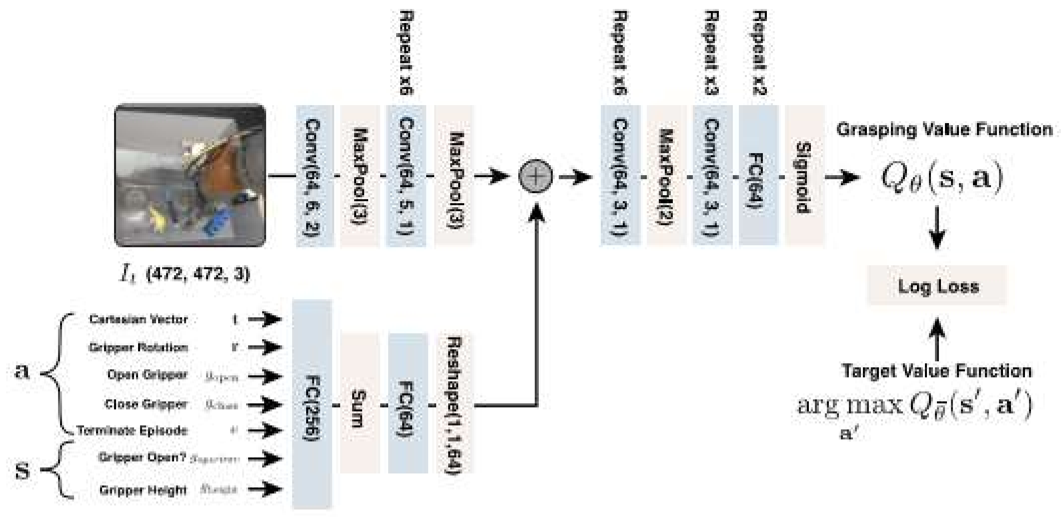


Figure 6 : Q-Function Neural Network Design

3.3 Conclusions

Initially they worried replacing objects in the bin would cause the robots to fixate on objects but found that not to be true in practice. Their framework performed very well on the test set achieving 87% accuracy on the test set after 580k training grasps and 96% accuracy after additional 28,000 on-policy training grasps for joint finetuning. When emptying the bins, they found the algorithm to perform worse as it got to the final objects. Below is a table describing the success rate of grasping objects within 10, 20, and 30 grasp attempts.

Bin emptying		
first 10	first 20	first 30
88%	**88%**	**76%**
76%	72%	72%

Figure 7 : Grasping Success Rates During Bin Emptying

4 SEGMENTING UNKNOWN 3D OBJECTS FROM REAL DEPTH IMAGES USING MASK R-CNN TRAINED ON SYNTHETIC DATA (KEN GOLDBURG [2019])

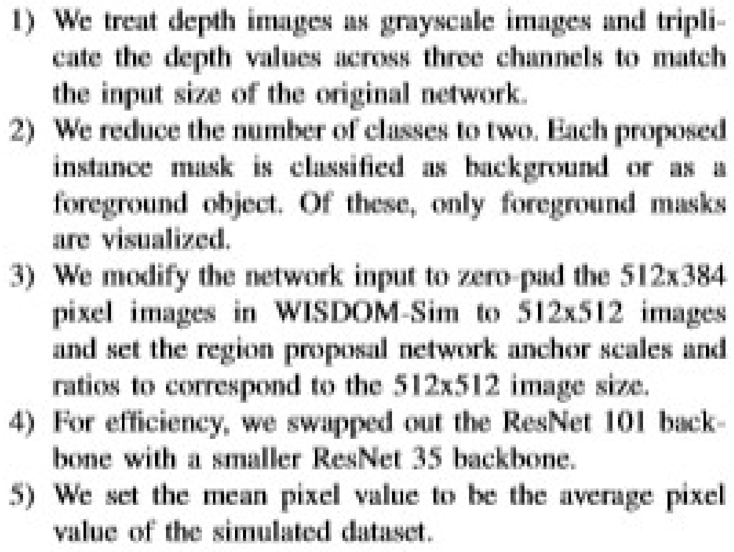
4.1 GOALS

This paper attempts to show that Mask R-CNN networks can be trained using synthetic depth images and hopes to demonstrate its usefulness in a robotics application through grasping. Furthermore, they purpose a method for rapidly generating synthetic depth images using domain randomization and present a hybrid real/sim dataset, WISDOM, for training and evaluating segmentation methods in the context of robot bin picking.

4.2 METHODS

They define category-agonistic instance segmentation as finding the set of visible foreground object masks, M, from a given depth image y where M contains the sets of pixels generated by the surface of some object Oi. The synthetic data is generated from sampling two distributions: a diverse set of object geometries, poses, and camera parameters, and an observation distribution that models sensor operation and noise. Real objects were sampled from a diverse set of 50 commonly found, novel objects with highly varied geometry. Objects used in WISDOM-Real have no corresponding CAD model in WISDOM-Sim. After dataset collection, all images were hand labeled; they estimate the labeling of 800 real images took over 35 hours of effort.

Training of the Mask R-CNN was based on Matterport's open-source Keras and TensorFlow implementation from GitHub which closely follows the original Mask R-CNN paper with the following modifications:



1) We treat depth images as grayscale images and triplicate the depth values across three channels to match the input size of the original network.
2) We reduce the number of classes to two. Each proposed instance mask is classified as background or as a foreground object. Of these, only foreground masks are visualized.
3) We modify the network input to zero-pad the 512x384 pixel images in WISDOM-Sim to 512x512 images and set the region proposal network anchor scales and ratios to correspond to the 512x512 image size.
4) For efficiency, we swapped out the ResNet 101 backbone with a smaller ResNet 35 backbone.
5) We set the mean pixel value to be the average pixel value of the simulated dataset.

Figure 8 : Modifications to the original Mask R-CNN Implementation

To test how SD Mask R-CNN performed compared to baseline methods, they placed a subset of ten objects from WIDSOM-Real's test set into a bin. They had SD Mask R-CNN output a labeled mask for each object. The mask with the highest predicted probability of being the target was fed to Dex-Net 3.0 for grasp planning. Each iteration was considered a success if the target object was successfully grasped, lifted, and removed from the bin.

4.3 CONCLUSIONS

After testing 50 iterations of picking 10 objects from WISDOM-Real, they found SD Mask R-CNN had achieved a success rate of 74% which is higher than the baseline method of PCL Euclidean Clustering which had a success rate of 56%. With further fine tuning, they were able to improve Mask R-CNN to have a success rate of 76%. They state WISDOM-Sim's dataset component enabled transfer to real images without expensive hand-labeling; consequently, suggesting that depth alone can encode segmentation cues. Furthermore, their experiments suggest more data could continue to increase performance.

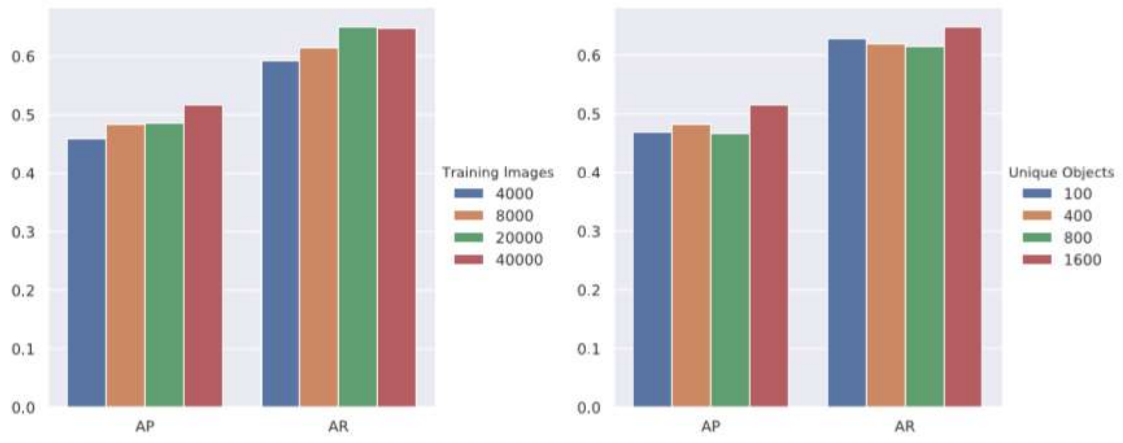


Figure 9 : Effect of increasing dataset size and number of unique objects on performance of R-CNN on the WISDOM-Real test set.