QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation

arXiv:1806.10293, Kalashnikov et al, 2018.

Sumamrized by Hyecheol (Jerry) Jang

Department of Computer Sciences
University of Wisconsin–Madison

RL Paper Study, Jun. 29. 2020

Table of Contents

Motivation: Why Robotics + Reinforcement Learning

- Usually, Robots are good at **repetitive tasks** (e.g. Assembly Line)

Motivation: Why Robotics + Reinforcement Learning

- Usually, Robots are good at **repetitive tasks** (e.g. Assembly Line)
- Want to make Robots that **identifies surroundings** and **behave accordingly**, but it is difficult

Motivation: Why Robotics + Reinforcement Learning

- Usually, Robots are good at **repetitive tasks** (e.g. Assembly Line)
- Want to make Robots that **identifies surroundings** and **behave accordingly**, but it is difficult
  - **Deep Learning**
    Provide ability to handling real-world scenarios
  - **Reinforcement Learning**
    Provide ability to make decision in long-term, using previous experiences in complex and robust scenarios

Motivation: Why Robotics + Reinforcement Learning

- Usually, Robots are good at **repetitive tasks** (e.g. Assembly Line)
- Want to make Robots that **identifies surroundings** and **behave accordingly**, but it is difficult
  - **Deep Learning**
    Provide ability to handling real-world scenarios
  - **Reinforcement Learning**
    Provide ability to make decision in long-term, using previous experiences in complex and robust scenarios
- Combining two techniques
  - Able to learn policy continuously from their experience
  - No need for manual engineering, use data they collects

- Varience in **visual and physical property of objects**

Motivation: Difficulites of Using RL in Robotics

- Varience in **visual and physical property of objects**
  - Hardness of object (Soft or Hard)
  - Surface Characteristics (Slippery, Sticky, ...)
  - Color Variation
  - Shape Variation
  - ...

Motivation: Difficulites of Using RL in Robotics

- Varience in **visual and physical property of objects**
  - Hardness of object (Soft or Hard)
  - Surface Characteristics (Slippery, Sticky, . . .)
  - Color Variation
  - Shape Variation
  - . . .
- **Noise** of sensors

Motivation: Difficulites of Using RL in Robotics

- Varience in **visual and physical property of objects**
  - Hardness of object (Soft or Hard)
  - Surface Characteristics (Slippery, Sticky, . . .)
  - Color Variation
  - Shape Variation
  - . . .
- **Noise** of sensors

$\Rightarrow$ Still hard to handle though we have sufficiently large training set

$\Rightarrow$ Collecting those training set is expensive (real experiments)

Motivation: Previous Works

- Focused on learning narrow, individual tasks
  - hitting a ball
  - opening door
  - throwing objects
  - ...

Motivation: Previous Works

- Focused on learning narrow, individual tasks
  - hitting a ball
  - opening door
  - throwing objects
  - ...

$\Rightarrow$ Use **Grasping** to achieve *generalization*

Motivation: Previous Works

- Focused on learning narrow, individual tasks
  - hitting a ball
  - opening door
  - throwing objects
  - ...

  $\Rightarrow$ Use **Grasping** to achieve *generalization*

- Approached the grasping task as predicting a *grasp pose*
  1. Observe the scene (*Normally, using a depth camera*)
  2. Choose best location to grasp
  3. Reach the location (open-loop setting)

Motivation: Previous Works

- Focused on learning narrow, individual tasks
  - hitting a ball
  - opening door
  - throwing objects
  - ...

  ⇒ Use **Grasping** to achieve *generalization*

- Approached the grasping task as predicting a *grasp pose*
  1. Observe the scene (*Normally, using a depth camera*)
  2. Choose best location to grasp
  3. Reach the location (open-loop setting)

  - Different with how humans and animals behave
  - Grasp is a **dynamical process** that sence and control at each stage

Motivation: Previous Works

- Focused on learning narrow, individual tasks
  - hitting a ball
  - opening door
  - throwing objects
  - ...

$\Rightarrow$ Use **Grasping** to achieve *generalization*

- Approached the grasping task as predicting a *grasp pose*
  1. Observe the scene (*Normally, using a depth camera*)
  2. Choose best location to grasp
  3. Reach the location (open-loop setting)

  - Different with how humans and animals behave
  - Grasp is a **dynamical process** that sence and control at each stage

$\Rightarrow$ **Where this researches start!!**

Table of Contents

Goal

Use Reinforcement Learning with Deep Neural Network
to **perform pre-grasp manipulation**,
response to dynamic disturbances,
and **learn grasping in a generic framework**
that makes minimal assumptions about the task

Goal: Constraint/Condition + Literature Review

- **Closed-loop condition** (With feedback, *Morrison, et al.*)
  - For the other papers work on closed-loop grasping, they deals with servoing problems.
  - This paper focuses on making generalized RL algorithm
  - In practice, it makes Kalashnikov et al.'s method (this method) to autonomously acquire complicated grasping strategy

Goal: Constraint/Condition + Literature Review

- **Closed-loop condition** (With feedback, *Morrison, et al.*)
  - For the other papers work on closed-loop grasping, they deals with servoing problems.
  - This paper focuses on making generalized RL algorithm
  - In practice, it makes Kalashnikov et al.'s method (this method) to autonomously acquire complicated grasping strategy
- **Self-supervised** learning task
  - Compare to prevoius work(by Zeng et al.), Kalashnikov et al. utilize more general action space
  - Actions consist of end-effector **Cartesian motion** and **gripper opening/closing**

Goal: Constraint/Condition + Literature Review

- **Closed-loop condition** (With feedback, *Morrison, et al.*)
  - For the other papers work on closed-loop grasping, they deals with servoing problems.
  - This paper focuses on making generalized RL algorithm
  - In practice, it makes Kalashnikov et al.'s method (this method) to autonomously acquire complicated grasping strategy
- **Self-supervised** learning task
  - Compare to prevoius work(by Zeng et al.), Kalashnikov et al. utilize more general action space
  - Actions consist of end-effector **Cartesian motion** and **gripper opening/closing**
- Observation comes from **a single RGB camera** over the sholder
  - Many current grasping system utilizes depth sensing
  - Using wrist-mounted cameras

Table of Contents

Overview of Model Architecture: MDP

- General Formulation of Robotic Manipulation:
  Based on **Markov Decision Process (MDP)**
  - partially observed formulation (POMDP) is more general.
  - However, assuming current observation contains all necessary information for this task, it is sufficient to use MDP.

Overview of Model Architecture: MDP

- General Formulation of Robotic Manipulation:
  Based on **Markov Decision Process (MDP)**
  - partially observed formulation (POMDP) is more general.
  - However, assuming current observation contains all necessary information for this task, it is sufficient to use MDP.
- MDP have a **general and powerful formalism** for decision making problems.

  However, it is **hard to train**

Overview of Model Architecture: MDP

- General Formulation of Robotic Manipulation:
  Based on **Markov Decision Process (MDP)**
  - partially observed formulation (POMDP) is more general.
  - However, assuming current observation contains all necessary information for this task, it is sufficient to use MDP.
- MDP have a **general and powerful formalism** for decision making problems.
  However, it is **hard to train**
- For each step of MDP:
  1. Observes Image from robot's camera (see Fig. 1)
  2. choose a gripper command, Reward:
     - failed grasp: reward of 0
     - successful grasp: reward of 1
       Defined *success* when the robot holds the object above a certain height

Overview of Model Architecture: MDP

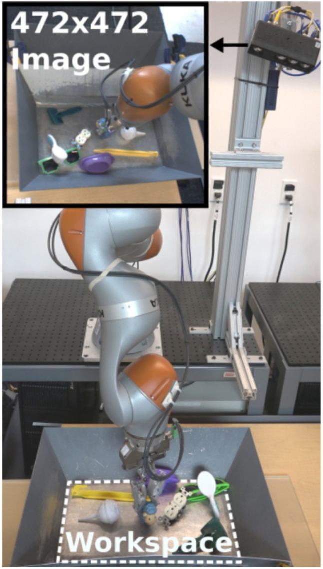

Figure 1: Configuration of robot cell, with a sample observation image on top-right box

Overview of Model Architecture: Algorithm Selection

- Usually, Generalization needs diverse data
  - However, recollecting experience on numerous objects after every policy update is impractical
  - Reason for **not using on-policy algorithm**

Overview of Model Architecture: Algorithm Selection

- Usually, Generalization needs diverse data
  - However, recollecting experience on numerous objects after every policy update is impractical
  - Reason for **not using on-policy algorithm**
- Using **scalable off-policy algorithm** based on Q-learning
  - actor-critic algorithm are popular for handling continuous actions
  - However, Kalashnikov et al. found **scalable and more stable ways** to train only Q-function

Overview of Model Architecture: Algorithm Selection

- Usually, Generalization needs diverse data
  - However, recollecting experience on numerous objects after every policy update is impractical
  - Reason for **not using on-policy algorithm**
- Using **scalable off-policy algorithm** based on Q-learning
  - actor-critic algorithm are popular for handling continuous actions
  - However, Kalashnikov et al. found **scalable and more stable ways** to train only Q-function
- Large Dataset and Network (See Fig. 2)
  - Kalashnikov et al. devised **distributed** training system (with 7 robots)
  - **Asynchronously update** target values, collect **on-policy data**, reloads **off-policy data** from previous experiences, and train network on both data stream.

Overview of Model Architecture: Algorithm Selection

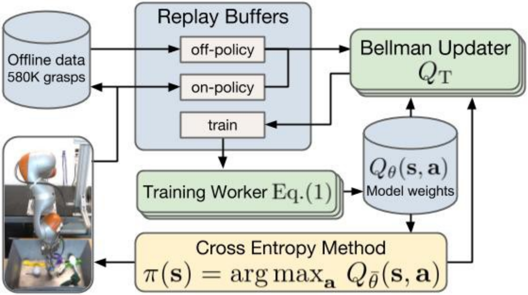

Figure 2: Distributed Reinforcement Learning infrastructure for QT-Opt.

- **On-policy Learning** learns the value of the policy being **carried out by the agent**, including the exploration steps.
  e.g. SARSA(State-Action-Reward-State-Action) (See Fig. 3)

Optional: On-policy vs Off-policy *Poole et al.*

- **On-policy Learning** learns the value of the policy being **carried out by the agent**, including the exploration steps.
  e.g. SARSA(State-Action-Reward-State-Action) (See Fig. 3)
- **Off-policy Learning** learns the value of the optimal policy **independently of the agent's action**
  e.g. Q-Learning (See Fig. 4)

Optional: On-policy vs Off-policy *Poole et al.*

controller SARSA(S,A,γ,α)

inputs:

S is a set of states

A is a set of actions

γ the discount

α is the step size

internal state:

real array $Q[S,A]$

previous state s

previous action a

begin

initialize $Q[S,A]$ arbitrarily

observe current state s

select action a using a policy based on Q

repeat forever:

carry out an action a

observe reward r and state s'

select action a' using a policy based on Q

$Q[s,a] \leftarrow Q[s,a] + \alpha(r + \gamma Q[s',a'] - Q[s,a])$

$s \leftarrow s'$

$a \leftarrow a'$

end-repeat

end

Figure 3: SARSA Algorithm

controller *Q-learning*(S,A,γ,α)

Inputs

S is a set of states

A is a set of actions

γ the discount

α is the step size

Local

real array $Q[S,A]$

previous state s

previous action a

initialize $Q[S,A]$ arbitrarily

observe current state s

repeat

select and carry out an action a

observe reward r and state s'

$Q[s,a] \leftarrow Q[s,a] + \alpha(r + \gamma \max_{a'} Q[s',a'] - Q[s,a])$

$s \leftarrow s'$

until termination

Figure 4: Q-Learning Algorithm

Table of Contents

QT-Opt

- Continuous action version of Q-Learning
  - For **scalable** learning and optimized for **stability**
  - To handle **large amount of off-policy image data** for complex tasks

QT-Opt: Revisit Q-Learning

- **state**: $\mathsf{s} \in \mathcal{S}$ (*Image Observations*)
- **action**: $\mathsf{a} \in \mathcal{A}$ (*Robot Arm Motions and Gripper commands*)
- at **each time step** t
  1. Choose an action
  2. transition to new state
  3. receive reward $\gamma(\mathsf{s}_t, \mathsf{a}_t)$

QT-Opt: Revisit Q-Learning

- Need to solve for Optimal Q-function: Minimize Bellman Error

$$\mathcal{E}(\theta) = \mathbb{E}_{(\mathsf{s},\mathsf{a},\mathsf{s}') \sim p(\mathsf{s},\mathsf{a},\mathsf{s}')}[\mathcal{D}(\mathcal{Q}_\theta(\mathsf{s},\mathsf{a}), \mathcal{Q}_\mathcal{T}(\mathsf{s},\mathsf{a},\mathsf{s}'))]$$

  Where $\mathcal{Q}_\mathcal{T}(\mathsf{s},\mathsf{a},\mathsf{s}') = r(\mathsf{s},\mathsf{a}) + \gamma V(\mathsf{s}')$ (*target value*)
- $\mathcal{D}$: divergence metric (squared difference for Q-Learning)
- Expectation is taken under the distribution over all previously observed transition

QT-Opt: Q-Learning Implementation

- Use **two target network** to improve stability
  - Maintaining two lagged version of the parameter vector $\theta, \bar{\theta}_1, \bar{\theta}_2$
  - $\bar{\theta}_1$: exponential moving averaged version of θ,
    averaging constant: 0.9999
  - $\bar{\theta}_2$: lagged version of $\bar{\theta}_1$,
    lagged by 6000 gradient steps

QT-Opt: Q-Learning Implementation

- Use **two target network** to improve stability
  - Maintaining two lagged version of the parameter vector $\theta, \bar{\theta_1}, \bar{\theta_2}$
  - $\bar{\theta_1}$: exponential moving averaged version of θ,
    averaging constant: 0.9999
  - $\bar{\theta_2}$: lagged version of $\bar{\theta_1}$,
    lagged by 6000 gradient steps
- Compute target value by
  $V(\mathsf{s}') = \min_{i=1,2} \mathcal{Q}_{\bar{\theta_i}}(\mathsf{s}', \mathit{arg}\max_{\mathsf{a}'} \mathcal{Q}_{\bar{\theta_1}}(\mathsf{s}', \mathsf{a}'))$

QT-Opt: Q-Learning Implementation

- Use **two target network** to improve stability
  - Maintaining two lagged version of the parameter vector $\theta, \bar{\theta_1}, \bar{\theta_2}$
  - $\bar{\theta_1}$: exponential moving averaged version of θ,
    averaging constant: 0.9999
  - $\bar{\theta_2}$: lagged version of $\bar{\theta_1}$,
    lagged by 6000 gradient steps
- Compute target value by
  $V(\mathsf{s}') = \min_{i=1,2} \mathcal{Q}_{\bar{\theta_i}}(\mathsf{s}', \mathit{arg}\max_{\mathsf{a}'} \mathcal{Q}_{\bar{\theta_1}}(\mathsf{s}', \mathsf{a}'))$
- the policy is recovered by $\pi(\mathsf{s}) = \mathit{arg}\max_{\mathsf{a}} \mathcal{Q}_{\bar{\theta_1}}(\mathsf{s}, \mathsf{a})$

QT-Opt: Q-Learning Implementation

- Use **two target network** to improve stability
  - Maintaining two lagged version of the parameter vector $\theta, \bar{\theta}_1, \bar{\theta}_2$
  - $\bar{\theta}_1$: exponential moving averaged version of θ,
    averaging constant: 0.9999
  - $\bar{\theta}_2$: lagged version of $\bar{\theta}_1$,
    lagged by 6000 gradient steps
- Compute target value by
  $V(s') = \min_{i=1,2} \mathcal{Q}_{\bar{\theta}_i}(s', \arg\max_{a'} \mathcal{Q}_{\bar{\theta}_1}(s', a'))$
- the policy is recovered by $\pi(s) = \arg\max_a \mathcal{Q}_{\bar{\theta}_1}(s, a)$
- After collects samples from environment interaction, then perform off-policy training on all samples collected
  - Problem: Large-scale learning (Technically Difficult)
  - Solution: Use **Parallel Asynchronous** version
    (which leads ability to scale up the process)

QT-Opt: Problem of Q-Learning

- Difficult to deal with continuous actions, e.g. continuous gripper motion

QT-Opt: Problem of Q-Learning

- Difficult to deal with continuous actions, e.g. continuous gripper motion
- *Previous Solutions:*
  - using a second network that "amortize" the maximization
  - constraining the Q-function to be convex in a, so that makes the function to easily find maximum ananlytically

QT-Opt: Problem of Q-Learning

- Difficult to deal with continuous actions, e.g. continuous gripper motion
- *Previous Solutions:*
  - using a second network that "amortize" the maximization
  - constraining the Q-function to be convex in a, so that makes the function to easily find maximum ananlytically
- problem of previous solutions
  - Unstable: problematic for large-scale RL tasks where running hyperparameter sweeps is very expensive
  - Action-convex value functions are poor fit for complex manipulation tasks

QT-Opt: Stable Continuous-Action Q-Learning

- To **maintain the generality of non-convex Q-function** while **not using a second maximizer network**
- the Bellman equation is evaluated with stocastic optimization
  - handles non-convex and multimodal optimization landscapes

QT-Opt: Stable Continuous-Action Q-Learning

- To **maintain the generality of non-convex Q-function** while **not using a second maximizer network**
- the Bellman equation is evaluated with stocastic optimization
  - handles non-convex and multimodal optimization landscapes

- QT-Opt
  - $\pi_{\bar{\theta}_1}(s)$ is evaluaed by running stochastic optimization over a, using $\mathcal{Q}_{\bar{\theta}_1}(s, a)$ as the objective value

QT-Opt: Stable Continuous-Action Q-Learning

- To **maintain the generality of non-convex Q-function** while **not using a second maximizer network**
- the Bellman equation is evaluated with stocastic optimization
  - handles non-convex and multimodal optimization landscapes

- QT-Opt
  - $\pi_{\bar{\theta}_1}(\mathrm{s})$ is evaluaed by running stochastic optimization over a, using $\mathcal{Q}_{\bar{\theta}_1}(\mathrm{s}, \mathrm{a})$ as the objective value
  - Use Cross-Entropy Method
    1. samples a batch of N at each iteration
    2. fits a Gaussian distribution to the best $M < N$ samples
    3. Samples next batch of N from the Gaussian

QT-Opt: Stable Continuous-Action Q-Learning

- To **maintain the generality of non-convex Q-function** while **not using a second maximizer network**
- the Bellman equation is evaluated with stocastic optimization
  - handles non-convex and multimodal optimization landscapes

- QT-Opt
  - $\pi_{\bar{\theta}_1}(\mathrm{s})$ is evaluaed by running stochastic optimization over a, using $\mathcal{Q}_{\bar{\theta}_1}(\mathrm{s}, \mathrm{a})$ as the objective value
  - Use Cross-Entropy Method
    1. samples a batch of N at each iteration
    2. fits a Gaussian distribution to the best $M < N$ samples
    3. Samples next batch of N from the Gaussian

    - easy to parallelize
    - moderately robust to local optima

QT-Opt: Distributed Asynchronous QT-Opt

- Requires large amount of diverse data to generalize over new scenes and objects for the image-based policy

QT-Opt: Distributed Asynchronous QT-Opt

- Requires large amount of diverse data to generalize over new scenes and objects for the image-based policy
- Needs of Distributed System (See Fig. 2)

QT-Opt: Distributed Asynchronous QT-Opt

- Requires large amount of diverse data to generalize over new scenes and objects for the image-based policy

- Needs of Distributed System (See Fig. 2)
  1. Replay buffer stores both **off-line data** (from disk) and on-going experiments' data (**on-line data**)

QT-Opt: Distributed Asynchronous QT-Opt

- Requires large amount of diverse data to generalize over new scenes and objects for the image-based policy

- Needs of Distributed System (See Fig. 2)
  1. Replay buffer stores both **off-line data** (from disk) and on-going experiments' data (**on-line data**)
  2. Data in buffer labeled with target Q-value using a set of 1000 "bellman updater" jobs

QT-Opt: Distributed Asynchronous QT-Opt

- Requires large amount of diverse data to generalize over new scenes and objects for the image-based policy

- Needs of Distributed System (See Fig. 2)
  1. Replay buffer stores both **off-line data** (from disk) and on-going experiments' data (**on-line data**)
  2. Data in buffer labeled with target Q-value using a set of 1000 "bellman updater" jobs
  3. Store the labeled sample in the training buffer
     - Some samples in the training buffer are **labeled with lagged version** of the Q-network

QT-Opt: Distributed Asynchronous QT-Opt

- Requires large amount of diverse data to generalize over new scenes and objects for the image-based policy

- Needs of Distributed System (See Fig. 2)
  1. Replay buffer stores both **off-line data** (from disk) and on-going experiments' data (**on-line data**)
  2. Data in buffer labeled with target Q-value using a set of 1000 "bellman updater" jobs
  3. Store the labeled sample in the training buffer
     - Some samples in the training buffer are **labeled with lagged version** of the Q-network
  4. Training workers pull the labeled sample from the training buffer **randomly** to update the Q-function
     - each training workers compute gradient that sent to parameter server asynchronously

QT-Opt: Distributed Asynchronous QT-Opt

- Requires large amount of diverse data to generalize over new scenes and objects for the image-based policy

- Needs of Distributed System (See Fig. 2)
  1. Replay buffer stores both **off-line data** (from disk) and on-going experiments' data (**on-line data**)
  2. Data in buffer labeled with target Q-value using a set of 1000 "bellman updater" jobs
  3. Store the labeled sample in the training buffer
     - Some samples in the training buffer are **labeled with lagged version** of the Q-network
  4. Training workers pull the labeled sample from the training buffer **randomly** to update the Q-function
     - each training workers compute gradient that sent to parameter server asynchronously
  - Empirically, it requires up to **15M gradient steps** to train effective Q-function due to **complexity of the task** and **the size of dataset and model**

Table of Contents

Environment Settings: Overview

- Policy
  - locate object
  - position for grasping (pre-grasping manipulation, if needed)
  - pick up (regrasping, if needed)
  - Pull up the object
  - signal thermination

Environment Settings: Overview

- Policy
    - locate object
    - position for grasping (pre-grasping manipulation, if needed)
    - pick up (regrasping, if needed)
    - Pull up the object
    - signal thermination
- Reward
    - only indicates whether or not an object was successfully picked up

Environment Settings: Overview

- Policy
  - locate object
  - position for grasping (pre-grasping manipulation, if needed)
  - pick up (regrasping, if needed)
  - Pull up the object
  - signal thermination
- Reward
  - only indicates whether or not an object was successfully picked up
- End-to-End approach of grasping!!
  - no prior knowledge about object, physics, or motion planning
  - model itself autonomously extract the knowledges from data

Environment Settings: MDP for grapsing

- state observation $s \in \mathcal{S}$ includes:
  - robot's current camera observation (RGB image, res: 472×472) from over-the-sholder single-lens camera
  - current status of gripper (binary)
  - vertical position of the gripper relative to the floor

Environment Settings: MDP for grapsing

- state observation $s \in \mathcal{S}$ includes:
  - robot's current camera observation (RGB image, res: 472 $\times$ 472) from over-the-sholder single-lens camera
  - current status of gripper (binary)
  - vertical position of the gripper relative to the floor
- action $a = (t, r, g_{open}, g_{close}, e) \in \mathcal{A}$ includes:
  - vector in Cartesian space $t \in \mathbb{R}^3$ (desired change in the gripper position)
  - change in azimuthal angle via sine-cosine encoding, $r \in \mathbb{R}^2$
  - binary gripper open and close command, g_{open}, g_{close}
  - termination command, e

Environment Settings: Reward Function

- 1: if gripper carries the object up above certain height at the end of the episode
- 0: otherwise

Environment Settings: Reward Function

- 1: if gripper carries the object up above certain height at the end of the episode
- 0: otherwise
- penalty (-0.05): for all time steps prior to termination
  - emits termination action
  - exceed the maximum number of time steps(20)

Environment Settings: Reward Function

- 1: if gripper carries the object up above certain height at the end of the episode
- 0: otherwise
- penalty (-0.05): for all time steps prior to termination
  - emits termination action
  - exceed the maximum number of time steps(20)

- delayed and sparse reward function is challenging, but more practical for automated self-supervision

Environment Settings: Q-function

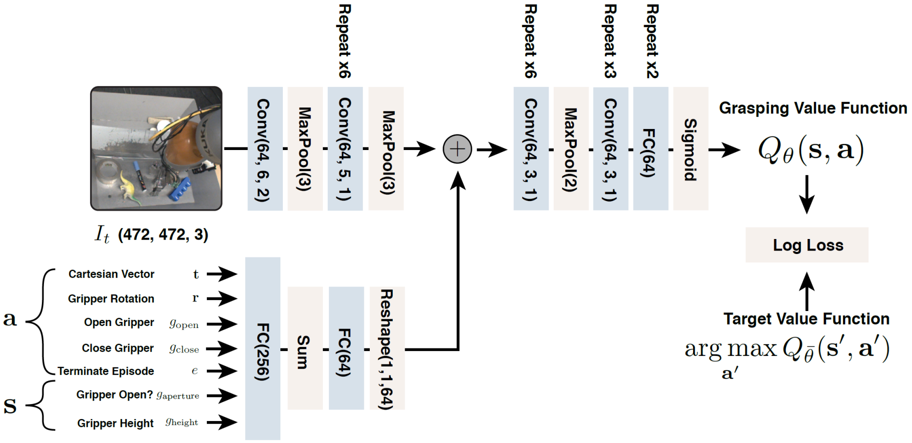

Figure 5: Neural Network Architecture for Q-Fuction.
It has total of 1.2M parameters. The image is processed with convolution filters, and the other inputs are processed by fully connected layer, then concatenated with the image

Environment Settings: Data Collection

- Need to collect data on sufficiently large and diverse set of objects
- Use multiple robots and multiple experiments to collect such data
  - Took four months, 800 Robot hrs
  - Collected during multiple separated experiments, and each experiment reused the data from the previous one

Environment Settings: Data Collection

- Need to collect data on sufficiently large and diverse set of objects
- Use multiple robots and multiple experiments to collect such data
  - Took four months, 800 Robot hrs
  - Collected during multiple separated experiments,
    and each experiment reused the data from the previous one

- Initialize policy
  - Used weak scripted exploration policy to bootstrap data collection
  - Still random, but biased toward reasonable grapsing
  - Success Rate: Around 15-30%
  - Switching to QT-Opt once it reach 50% of success rate

Environment Settings: Data Collection

- Need to collect data on sufficiently large and diverse set of objects
- Use multiple robots and multiple experiments to collect such data
  - Took four months, 800 Robot hrs
  - Collected during multiple separated experiments, and each experiment reused the data from the previous one

- Initialize policy
  - Used weak scripted exploration policy to bootstrap data collection
  - Still random, but biased toward reasonable grapsing
  - Success Rate: Around 15-30%
  - Switching to QT-Opt once it reach 50% of success rate

- Other Conditions
  - Using seven LBR IIWA robots
  - with 4-10 objects per robots
  - objects were replaced every 4 hours during business hours
  - Use different objects during test

Table of Contents

Experiments

1. Performance on unseen object (quantitative)
2. Performance comparision with other self-supervised grasping system (quantitative)
3. Manipulation strategies which carried out meaningful pre-grasp manipulation (quality)

Experiments: Quantitative Analysis

- Used two evaluation protocol
  - Each robots make 102 grasp attemps on test objects.
    - grasp attemps last for up to 20 times steps
    - any grasped objects returned back to the bin
    - Experimently, the robot made grasp attemps on a various objects, not picking one objects
    - Might have confounding effects

Experiments: Quantitative Analysis

- Used two evaluation protocol
  - Each robots make 102 grasp attemps on test objects.
    - grasp attemps last for up to 20 times steps
    - any grasped objects returned back to the bin
    - Experimently, the robot made grasp attemps on a various objects, not picking one objects
    - Might have confounding effects
  - bin emptying
    - single robot unload a bin with 28 test objects, using 30 grasp attemps
    - Repeated for five times
    - Success rate is reported over the first 10, 20, and 30 grasp attemps

Experiments: Quantitative Analysis (Result)

Method	Dataset	Test	Bin emptying		
			first 10	first 20	first 30
QT-Opt (ours)	580k off-policy + 28k on-policy	**96%**	**88%**	**88%**	**76%**
Levine et al. [27]	900k grasps from Levine et al. [27]	78%	76%	72%	72%
QT-Opt (ours)	580k off-policy grasps only	87%			
Levine et al. [27]	400k grasps from our dataset	67%			

Figure 6: Result of Quantitative Analysis Result.
Left half indicates the result of re-deposit grasping tasks, the right half indicates the results of bin emptying experiments

(See Fig. 6)

- Usage of on-policy training
  - on-policy joint finetuning provides better Performance
  - Kalashnikov et al. analyzed that on-policy helps the model to remove "hard negatives"

Experiments: Quantitative Analysis (Result)

(See Fig. 6)

- Usage of on-policy training
  - on-policy joint finetuning provides better Performance
  - Kalashnikov et al. analyzed that on-policy helps the model to remove "hard negatives"
- Bin emptying
  - successfully empty all objects within 30 grasps for 2/5 trials (Kalashnikov et al.)
  - Previous method (Levine et al.) successfully empty for 1/5 trials
  - lower success rate for 30 grasp due to the algorithm tend to grasp easy one first

Experiments: Quantitative Analysis (Result)

(See Fig. 6)

- Compare with Levine et al.
  - greedly optimized for grasp success at the next grasp
  - does not control the opening and closing of gripper
  - does not explain about pregrasp manipulation

Experiments: Quantitative Analysis (Result)

(See Fig. 6)

- Compare with Levine et al.
  - greedly optimized for grasp success at the next grasp
  - does not control the opening and closing of gripper
  - does not explain about pregrasp manipulation
  - action representation is different, making the format of dataset different

(See Fig. 6)

- Compare with Levine et al.
  - greedly optimized for grasp success at the next grasp
  - does not control the opening and closing of gripper
  - does not explain about pregrasp manipulation
  - action representation is different, making the format of dataset different
  - tested on both Levine et al.'s data format and this article's format
  - either way, it is worse than Kalashnikov et al's work

Experiments: Qualitative Analysis

Video: `https://sites.google.com/view/qtopt`

Experiments: Qualitative Analysis (Explaination)

- Singulation and pregrasp manipulation
    - Change position of object to make them easier to grasp

Experiments: Qualitative Analysis (Explaination)

- Singulation and pregrasp manipulation
  - Change position of object to make them easier to grasp
- Regrasping
  - open and close the gripper at any time
  - detect failed or unstable grasp earlier
    so that let the robots to grisp more securely

Experiments: Qualitative Analysis (Explaination)

- Singulation and pregrasp manipulation
  - Change position of object to make them easier to grasp
- Regrasping
  - open and close the gripper at any time
  - detect failed or unstable grasp earlier
    so that let the robots to grisp more securely
- Handling disturbance and Dynamic objects
  - grasp object that moving dynamically (e.g. balls)
  - though the sequence intentionally disturbed,
    it still able to grasp the object

Experiments: Qualitative Analysis (Explaination)

- Singulation and pregrasp manipulation
  - Change position of object to make them easier to grasp
- Regrasping
  - open and close the gripper at any time
  - detect failed or unstable grasp earlier
    so that let the robots to grisp more securely
- Handling disturbance and Dynamic objects
  - grasp object that moving dynamically (e.g. balls)
  - though the sequence intentionally disturbed,
    it still able to grasp the object
- Grapsing in clutter
  - Note that there exists upto 10 objects during training time
  - The policy still able to grasp object in dense clutter

Experiments: Qualitative Analysis (Explaination)

- Singulation and pregrasp manipulation
  - Change position of object to make them easier to grasp
- Regrasping
  - open and close the gripper at any time
  - detect failed or unstable grasp earlier
    so that let the robots to grisp more securely
- Handling disturbance and Dynamic objects
  - grasp object that moving dynamically (e.g. balls)
  - though the sequence intentionally disturbed,
    it still able to grasp the object
- Grapsing in clutter
  - Note that there exists upto 10 objects during training time
  - The policy still able to grasp object in dense clutter
  - Some failure
    - prone to regrasp repeatedly
    - often produce successful graps, but time consuming

Table of Contents

Table of Contents

Bibliography I

- **Kalashnikov, Dmitry, et al. QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation. 28 Nov. 2018, arxiv.org/abs/1806.10293.**
- Irpan, Alex, and Peter Pastor. Scalable Deep Reinforcement Learning for Robotic Manipulation. 28 June 2018, ai.googleblog.com/2018/06/scalable-deep-reinforcement-learning.html.
- Morrison, Douglas, et al. "Closing the Loop for Robotic Grasping: A Real-Time, Generative Grasp Synthesis Approach." Robotics: Science and Systems XIV, 2018, doi:10.15607/rss.2018.xiv.021.
- Poole, David Lynton, and Alan K. Mackworth. Artificial Intelligence: Foundations of Computational Agents. Cmabridge University Press, 2018.