Reinforcement learning applied to a Universal Robot with a vacuum gripper

Álvaro Burgos Madrigal

ICAM, Universidad Pontificia Comillas – ICAI

E-mail: 201603823@alu.comilllas.edu

Abstract

In collaboration with the ICAM university, this project approaches the problem of teaching a robot with a vacuum gripper to pick and place, from a box full of the same objects, one of the objects, by identifying the best position to place its vacuum gripper and without any major information about the object and only using a depth and color camera. In the project, a digital twin of the scenario is developed and a reinforcement learning algorithm is deployed, performing a sensitivity analysis and comparing the optimized results with different approaches of sampling the batch of experiences used for training.

Key Words: *vacuum gripper, Resnet, CNN, NN, Replay Memory, Reinforcement Learning*

Introduction

Artificial intelligence is one of angular pieces of the smart industry, or industry 4.0, which will disrupt the present paradigm of the day-to-day industrial processes as they are known.

The smart industry focuses on two main objectives. The first objective is to improve the efficiency and efficacity of the current industrial processes. When a robot learns to do a task by his own, he will be often capable of delivering the same result as a human but in less time, with fewer defects and consuming less resources. These improvements will drive long term rentability on the enterprises and thus it is destined to arrive. The second important objective of the smart industry, and specially in reinforcement learning, is to place the humanity where the true value of the enterprise is. A robot will mostly serve to eliminate the repetitive tasks that a worker is forced to do and that do not contribute to giving meaning to his life, and to assist the human providing enhanced visibility where the human skills don't keep up.

The purpose of this project follows these objectives. The project and report that will be presented represent the next steps of a Research & Development team of the university of ICAM, in Toulouse. The team started developing this project two years ago, and its final objective is to teach a robot to pick up objects using vision. The scenario which the robot is set to resolve is formed by two boxes, one camera and the robot itself. One of the boxes is full of the same objects, and the other one is empty. Then, the objective is that the robot detects the point with the highest

probability of being apt for the pick. Then, the robot will then place the vacuum gripper in the point selected and successfully pick the object and place it in the other box.
The main complexity of the problem is due to the lack of information about the object that is going to be picked. The algorithm will only have access to a color camera, a depth camera, and will know the depth of the floor.
This project, could ultimately have various applications. Among them, this robot, trained correctly, would be able to place some objects, such as components of a product, screws, tools etc, from the inventory of a factory to the plate of an AGV which could, once the robot has finished approach these items to the operator, save him the time and the discomfort of having to stand up to go from the workspace to the storage and carry all the needed items with himself.

Overview of the technologies

Reinforcement Learning is a subfield of Machine Learning, but is also a general-purpose formalism for automated decision-making and AI. A reinforced learning algorithm benefits from a continuous experience on the environment, enlarging the available dataset and continuously learning from the mistakes to improve recurrently the overall accuracy.

One of the bigger differences between a reinforcement learning algorithm and a supervised algorithm is the existence of consequent decision-making processes that depend on each other. In a classic reinforcement learning system the algorithm will interact with a scenario, starting from an initial position and making a decision based on the scenario. This decision will drive a new position in which the algorithm will have to make a decision. This process will be repeated until the scenario gets to an end and the scenario is reset, starting from the starting position again. The group of consequent decisions and positions until the scenario finishes is called an episode.

The process of continuously learning is defined using the concepts of state, action and reward, present in Figure 1. Firstly, a state gives the algorithm a full overview of the current scenario of the robot. Secondly, the algorithm chooses, from all the possible actions, one action. Thirdly, the action implemented in the scenario results in a reward and the next state. With this information in mind, the reinforced learning algorithm will learn to understand, for a given state, which actions are the ones that will maximize the total reward on the episode, which not only includes the reward resulted after an action, but also all the predicted rewards after the consequent actions on the states resulting from that action.

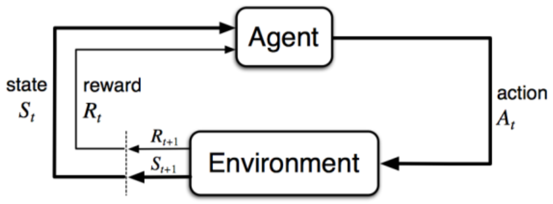

Figure 1 Reinforcement Learning Block Diagram

The reinforcement algorithm is divided into several modules that interact with each other:

Environment

The environment is the scenario (system, machine, game, etc) where the user wants to excel. A reinforcement learning algorithm communicates with the environment in two steps. In the first one, the environment sends information about the current situation (current state, actions). In the second one, the environment receives the decision about the actions the environment should do, and receive a reward for this action as a reply, as well as the next state of the environment.

Between the environment and the algorithm there is usually an environment manager, whose task is to mediate between the format the algorithm needs and the formats of the environment, for every state, action or reward.

In this project, the environment will be the virtual representation (made in Coppelia) of the laboratory in Toulouse.

Agent

The agent is the decision maker who impersonates all the decisions that are made across the training of the algorithm. It is the agent that usually merges all the elements that form a reinforcement learning algorithm.

Replay Memory

At every timestep, the agent is in a state and has to make an action. The action leads to a reward and the next state. This tuple of state, action, reward and next step is called experience. Thus, Replay Memory is the term for the dataset that stores the agent's experiences while training, so they can be accessed for improving the accuracy of the results.

In this project the replay memory will be formed by tuples of all the points where the robot has tried to picked an object, represented by a preprocessed image of 50 px around the point of grip, and the reward of the action once executed.

It is important to outline that, at the end of every episode, if the number of experiences is higher than the batch size, the policy (Neural Network) is trained. To do that a batch of experiences in the Replay Memory is sampled. In a common reinforcement learning algorithm, this sampling is random, whereas in this project, other ways of sampling this batch will be explored.

Policy

The policy is the term that corresponds to the intelligence that recurrently progresses in the decision-making process across every training. In a classical reinforcement problem, the input is the current state of the environment, and the output is the recommendation of the action to be taken, having one node in the output layer for every possible action that the agent can take.

In the cases with some complexity, notably in those cases where the number of possible states is not defined (i.e. a state defined by a photo), a Neural Network is used to reduce the computation needed to achieve acceptable results.

In this project, the input of the Neural Network will be a preprocessed squared snapshot of 50 px centered in the point of grip. The neural network will have one node in the output layer, which represents the probability of the robot picking an object if positioning in this point.

Thus, in this project, for every episode the Neural Network will be run as many times as eligible points in the current scenario. Then, the agent will pick the point whose probability is the highest.

Strategy

The strategy stands for the Epsilon Greedy Strategy. In every reinforcement learning algorithm, there is always a balance between exploration and exploitation. If the agent is in mode exploration, it will pick an action randomly out of the possible options. Nevertheless, in the exploitation mode, the policy will be used to decide which of the actions is the most suitable, looking for maximizing the reward.

In the first episodes of the training, the epsilon greedy strategy must be programmed to explore predominantly, as the scenario is still unknown and an exploration phase could drive results biased. For example, in a scenario with several positive rewards, if only exploiting the environment, the agent could find the less positive reward first, and keep exploiting this reward, without adverting that there is a bigger reward at reach.

However, after the Neural Network is trained, if the scenario is not planned to change, the epsilon greedy strategy should be favoring more and more exploitation episodes, to maximize the output and perfectionate the current results.

State of the Art

The prior version of the solution uses a supervised learning algorithm to train the robot, based on three phases:

1. The first one is the data acquisition, in which the robot tries to pick randomly points inside the box full of objects and saves a cropped snapshot of the camera around the selected point with a tag of success/fail depending if the object was picked or not.
2. The second one is the training phase in which a Residual Convolutional Neural Network (Resnet CNN) and a Neural Network (NN) are used to provide the optimal weights to predict success/fail with the snapshot of the point selected. The Convolutional Neural Network is already trained to provide the main characteristics of a snapshot.
3. The third one is the test phase, which ultimately will be the production phase, in which the algorithm is tested to calculate the accuracy.

These phases are manually started and finished, and thus is not optimal.

Objectives of the project

Therefore, the project explained in this report will explore a different solution, using reinforcement learning. The CNN will still be a Resnet pretrained, but the NN will follow a reinforcement learning algorithm, which will convert the three phases mentioned above into one dynamic algorithm.

Another pain point of the researches of ICAM these years is that they have had to resolve all the problems in the robot in order to train the CNN/NN algorithm, which sometimes results in slow project advances. Thus, the project will be built under a digital twin of the laboratory in Toulouse, using the simulation software Coppelia.

To do so, the 6 months of the project will be divided in three parts equally distributed in time. The first two months of the project will focus on acquiring the expertise on both the reinforcement learning and Coppelia. The second two months of the project will be reserved for developing the software and the algorithm. Lastly, in the last two months the results will be analyzed, a sensitivity analysis will be conducted, and the report will be written.

Model Deployed

Figure 2 shows the scenario built in the software Coppelia. The robot UR5 is in the middle of the scenario, and has two boxes at either sides of the robot, one full of objects and the other one empty. Each box has two different cameras, one RGB and one that measures depth. The output of these cameras can be seen on the top left and top right of the figure.

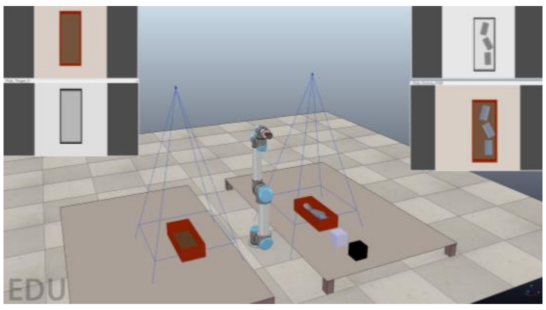

Figure 2 Coppelia Environment Representation

The general procedure of the algorithm will start by making RGB and depth a photo of the box full of objects. The depth camera will preprocess the points, being eligible points only those with a height higher than a threshold, to prevent the robot from picking a point in the base of the box. Then, the reinforcement learning algorithm will select, following exploration/exploitation, the point that will be picked.

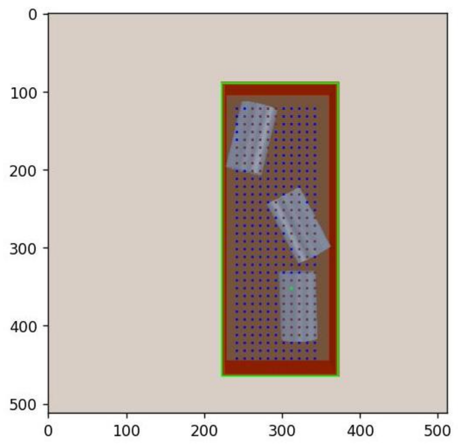

Figure 3 Preprocessed photo of the virtual box with contour, grid, and the point where the robot will pick

Figure 3 shows a green rectangle, corresponding to the computed contour of the box, some blue points, corresponding to non-eligible points in the grid (height lower than threshold), some red points, corresponding to the eligible points, and a green point, corresponding to the point out of the eligible points that the algorithm has decided to place the vacuum gripper. (image of exploration episode)

Figure 4 shows a schema of the CNN and the Neural Network, that relates the snapshot of the camera with the success/fail variable. Note that the final layer of the NN is only one node (success/fail).

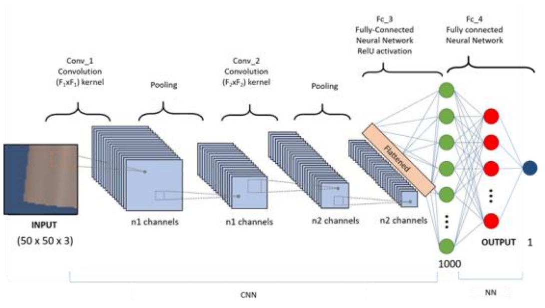
Conv_1 Convolution (F_1xF_1) kernel
Pooling
Conv_2 Convolution (F_2xF_2) kernel
Pooling
Fc_3 Fully-Connected Neural Network RelU activation
Fc_4 Fully connected Neural Network
INPUT (50 x 50 x 3)
n1 channels
n1 channels
n2 channels
n2 channels
Flattened
1000
OUTPUT 1
CNN
NN

Figure 4 Illustrative Block Diagram of a CNN followed by a Neural Network

Once the point has been selected, the robot targets the point, places the vacuum gripper there, and activates it. If the pick was successful, the robot places the point in the other box. If not, it returns to the position of safety (Figure 2). As the algorithm improves, the reinforcement learning algorithm will predominantly use the Neural Networks to select the points, and the accuracy of the algorithm will be higher and higher.

Once the model is deployed, a sensitivity analysis will be conducted to find the optimal hyperparameters that provide the best accuracies. Additionally, an improvement on the batch of images used to train the NN will be presented. In a general reinforcement learning algorithm, the batch of images is randomly selected. In this project two new ways of selecting the batch of images will be presented. The first one will be selecting those points whose prediction is closer to 0,5 (indecisive sampling, algorithm not deciding between success/fail). The second one will select those points whose prediction is further from the real outcome of the point (worst-case sampling).

Results

A sensitivity analysis has been conducted to find the best hyperparameters for the whole model. Specifically, the following parameters have been optimized:

- Percentage Decay: Term acquainted in the study, directly related with the actualization rate in the Epsilon Greedy Strategy, which is responsible for deciding if between exploration and exploitation. A percentage decay of 30% means that when the training has arrived to the 30% of all its training episodes, epsilon is 0,5, and thus the probability of exploration vs exploitation is 50%.
- Learning Rate: Learning rate of the Neural Network. A high learning rate will trust more the new trainings than the old weights. Therefore, a high learning rate can drive a faster convergence, but it can lead to higher variance in the final results.
- Memory Capacity: Number of experiences that can be stored in the Replay Memory, which will be used for training the NN. A high Memory Capacity drives an unbiased random distribution, but can have difficulties in a changing environment.
  Batch Size: Number of experiences used from Replay Memory to train the NN in every episode. A higher batch size drives faster convergence but needs higher computation resources.

For every sensitivity analysis three different plots have been analyzed.

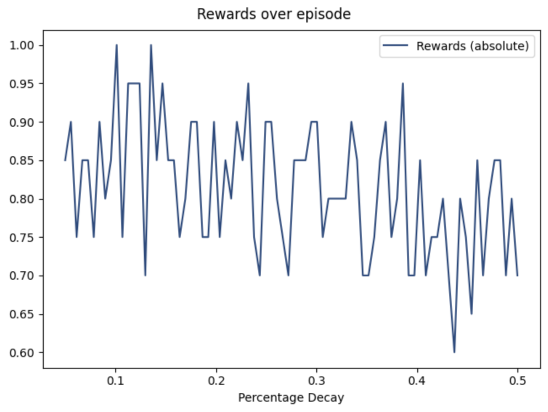

Figure 5 Sensitivity plots: Average of last rewards

The first plot in Figure 5 corresponds to the average of the last 10 rewards for every training. Therefore, this plot represents how the algorithm is performing in the last episodes.

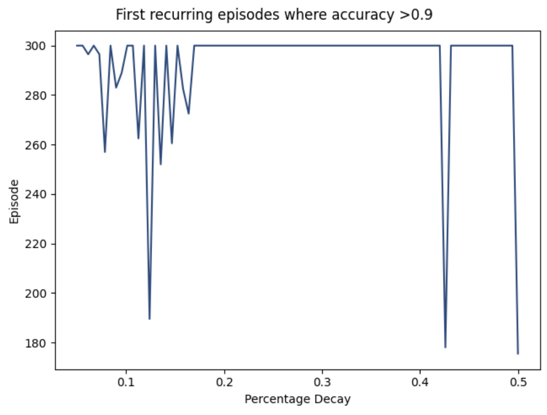

Figure 6 Sensitivity plots: First episode with recurring accuracies of above 0.9

The second plot in Figure 6 shows the first episode for every training in which the accuracy in the training batch is higher than 0,9. Hence, this plot shows how fast the algorithm is converging.

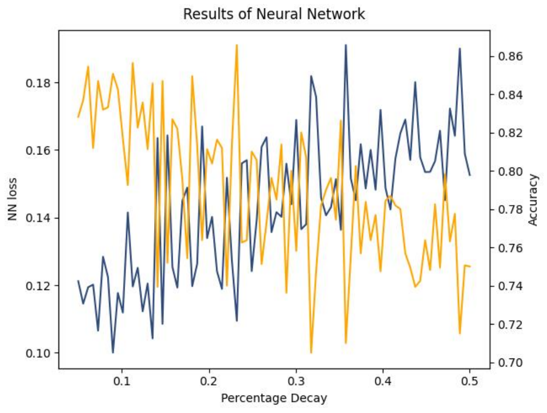

Figure 7 Sensitivity plots: Accuracy on training batch and loss function

The third plot in Figure 7 shows the last values of the accuracy and loss of the Neural Network, representing how well the Neural Network has been optimized through the episodes.

Once the hyperparameters were optimal, the different memory samplings were compared to find the most suitable one. Figure 8 shows the accuracy growth over episodes in the reinforcement learning algorithm, for the optimized hyperparameters, for the three cases of batch sampling mentioned above (Randomly Sampling, Indecisive Sampling and Worst-case Sampling, respectively).

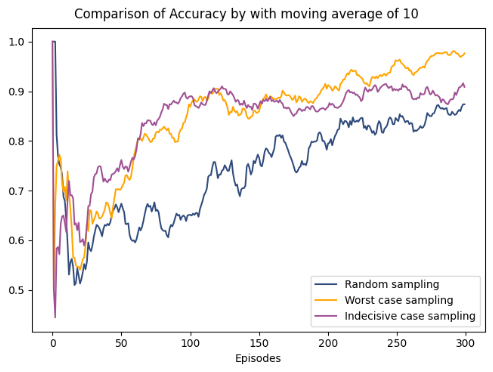

Figure 8 Comparison on results based by batch sampling

The model which seems to be converging faster is the indecisive sampling, with a convergence of around 90% in the episode 100, but the worst-case sampling has the best accuracy of everyone. Therefore, if this model was to be implanted in a warehouse, the worst-case sampling would be selected.

As indecisive sampling and worst-case sampling offer different advantages throughout the episodes, a dynamic sampling has been explored, which means that the way of sampling changes across the training. As the indecisive sampling has a fast convergence, and the worst-case sampling is giving a good final result, the training will start with indecisive sampling, then random sampling (to soften the changes) and lastly worst-case sampling.

Two different models have been created, with split 30/40/30 % and 15/35/50% of indecisive sampling, random sampling and worst-case sampling, respectively. The result is in Figure 9:

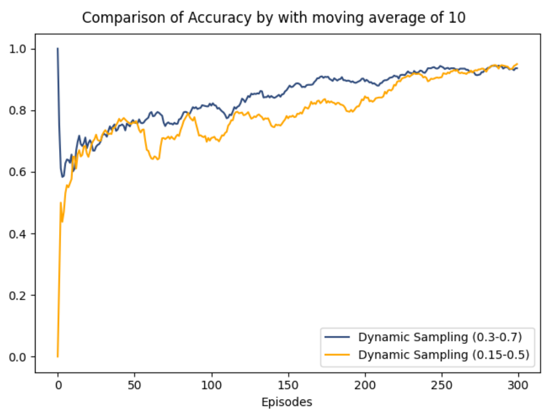

Figure 9 Comparison of the dynamic sampling strategies

It seems that the model in blue (30/40/30 % split) is converging faster, which makes sense as the indecisive sampling is longer. Nevertheless, the two models finish with similar end results, which do not seem higher than for the worst-case scenario. In the Figure 10 all the models have been analyzed:

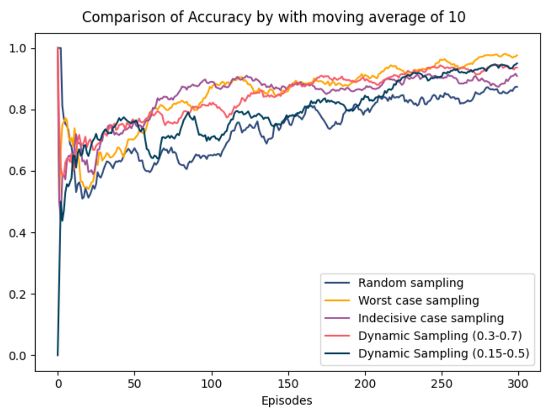

Figure 10 Comparison of all the final results

As mentioned, the dynamic sampling, though promising, seems to have worse results that the worst-case sampling, and therefore the worst-case sampling is the selected one.

Conclusions

There are several conclusions to drive from the results. The first conclusion is that the indecisive sampling seems to have a faster improvement in accuracy, but a lower final accuracy than the worst-case sampling. Thus, the best solution is the worst-case Sampling, because the final accuracy is far higher than the other two solutions. The final accuracy over 300 episodes of training is around 97%.

In second place, a dynamic sampling, even though it has not given the correct results, could be furtherly explored as it could provide the perfect balance of fast convergence and final accuracy.

The third conclusion concerns the digital environment. It seems to be working well as it mimics perfectly the normal operation of an anthropomorphic robot, such as the one in the laboratory. In terms of the objects used, the shape and the orientation of the object is similar to the objects used in the laboratory. The dynamic behavior of the objects is also an acceptable representation of the physical world. Therefore, it can be used for testing the algorithms built in the laboratory when the real environment is not working.

Nevertheless, there are two features of the robot that could be improved to better represent the reality: the vacuum gripper and the sensor that detects if the object is attached. The vacuum gripper only grips an object if the point if the middle of the tool is touched by the object. In that case, the base of the gripper may not be entirely covered by an object, and still pick the object (something impossible in a real vacuum gripper). Additionally, the sensor that detects if an object is gripped corresponds to the same point, in the middle of the vacuum gripper. The object is detected only if the object is touching the middle of the base of the vacuum gripper, which could not always be the case in the real world (Due to pression in the vacuum gripper the height of the base could change).

The fourth conclusion of the project is regarding the robustness of the results. Even though these results are promising, the scenario is a software. The main next step should be to deploy the solution in the real laboratory, to confirm the results in a physical scenario.

Bibliography

Antoniadis, P. (2022, 7 2). *Activation Functions: Sigmoid vs Tanh*. Retrieved from Baeldung.com: https://www.baeldung.com/cs/sigmoid-vs-tanh-functions#:~:text=and%20presents%20a%20similar%20behavior,instead%20of%201%20and%200.

Armesto, L. (n.d.). *Introduction to CoppeliaSim Course | CoppeliaSim (V-REP)*. Retrieved from youtube.com: https://www.youtube.com/watch?v=PwGY8PxQOXY

Arubai, N. (2022, 5 20). *wiki.ros.org*. Retrieved from Documentation: https://wiki.ros.org/

deeplizard. (2019). *Convolutional Neural Networks (CNNs) explained*. Retrieved from Youtube: https://www.youtube.com/watch?v=YRhxdVk_sIs

Education, I. C. (2020, 07 15). *Machine Learning*. Retrieved from ibm.com: https://www.ibm.com/cloud/learn/machine-learning

Gharat, S. (2019, 4 14). *medium.com*. Retrieved from What, Why and Which?? Activation Functions: https://medium.com/@snaily16/what-why-and-which-activation-functions-b2bf748c0441

Lizard, D. (n.d.). *deeplizard.com*. Retrieved from Reinforcement Learning - Developing Intelligent Agents: https://deeplizard.com/learn/playlist/PLZbbT5o_s2xoWNVdDudn51XM8lOuZ_Njv

Nations, U. (n.d.). *Do you know all 17 SDGs?* Retrieved from sdgs.un.org: https://sdgs.un.org/goals

Rastogi, A. (2022, 3 14). *ResNet50*. Retrieved from blog.devgenius.io: https://blog.devgenius.io/resnet50-6b42934db431

Regular API reference. (n.d.). Retrieved from https://www.coppeliarobotics.com/helpFiles/en/apiFunctions.htm: coppeliarobotics.com

reshalfahsi. (2019, 10 23). *arm-suction-sim*. Retrieved from github.com: https://github.com/reshalfahsi/arm-suction-sim

Technology, I. (2021). *What are Convolutional Neural Networks (CNNs)?* Retrieved from youtube.com: https://www.youtube.com/watch?v=QzY57FaENXg