# Introduction

This report covers the process that how i re-implemented existing project about Deep Deterministic Policy Gradient (DDPG) algorithm. This project is sample learning project created by udacity. It covers training and running phases. It is using Unity Environment. I run this project on windows 11, so I used Reacher with 20 agents. Each agent has two joints of the Robotic arm and each agent has circle to reach.

# Installing requirements and setting up environment

First of all, I installed [NVIDIA CUDA 11.8](https://developer.nvidia.com/cuda-11-8-0-download-archive) and [miniconda3](https://docs.anaconda.com/miniconda/install/#quick-command-line-install) to my laptop. My laptop’s specs are:

* Intel i9-13980HX CPU
* 32 GB RAM
* RTX 4070 Laptop GPU with 8 GB DRAM
* Windows 11

I’m skipping mentioning about some other necessary softwares like VS Code, Git bash etc. It’s important to have Nvidia GPU to train model with cuda cores. I installed miniconda3 with cmd prompts:

curl https://repo.anaconda.com/miniconda/Miniconda3-latest-Windows-x86\_64.exe -o .\miniconda.exe

start /wait "" .\miniconda.exe /S

del .\miniconda.exe

Then I cloned github repository, created new virtual environment with miniconda:

cd C:\Users\MyBirer\Desktop

git clone https://github.com/SHIVOH/DeepReinforcementLearning-DDPG-for-RoboticsControl.git

ren DeepReinforcementLearning-DDPG-for-RoboticsControl drl

cd drl

conda create --name drlnd python=3.6

conda activate drlnd

After a few hours to solve version problems, finally I found stable working version for this project is python 3.6. If you want to learn more details, [check link please.](https://github.com/udacity/deep-reinforcement-learning?tab=readme-ov-file#dependencies)

I installed pytorch with cuda with conda. If you want to install via another installer, you can check [the link.](https://pytorch.org/get-started/locally/)

conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia

install other necessary packages:

pip install numpy unityagents

At this point I encountered some package errors. I checked error logs and finally run the project.

# Testing current models

I arranged project folder structure like this:

A screenshot of a computer program

Description automatically generated

Then I wrote the test\_reacher.py to test current actor\_solved.pth and critic\_solved.pth models:

from unityagents import UnityEnvironment

import numpy as np

import torch

from ddpg\_agent import Agent

import time

env = None

try:

    print("Starting Unity Environment for testing...")

    env = UnityEnvironment(

        file\_name='./env/Reacher\_Windows\_x86\_64/Reacher.exe',

        no\_graphics=False,  # Görsel modda çalıştıralım

        worker\_id=0

    )

    # get the default brain

    brain\_name = env.brain\_names[0]

    brain = env.brains[brain\_name]

    # create DDPG agent

    agent = Agent(state\_size=33, action\_size=4, random\_seed=0)

    # load the trained weights

    agent.actor\_local.load\_state\_dict(torch.load('./models/actor\_solved.pth'))

    agent.critic\_local.load\_state\_dict(torch.load('./models/critic\_solved.pth'))

    print("Starting test episodes...")

    num\_episodes = 5  # Test için episode sayısı

    for i\_episode in range(num\_episodes):

        env\_info = env.reset(train\_mode=False)[brain\_name]  # train\_mode=False

        states = env\_info.vector\_observations

        scores = np.zeros(len(env\_info.agents))

        while True:

            actions = agent.act(states, add\_noise=False)  # Test için noise kapalı

            env\_info = env.step(actions)[brain\_name]

            next\_states = env\_info.vector\_observations

            rewards = env\_info.rewards

            dones = env\_info.local\_done

            scores += rewards

            states = next\_states

            if np.any(dones):

                break

        print(f'Episode {i\_episode+1} Score: {np.mean(scores):.2f}')

        time.sleep(0.5)  # Her episode arasında biraz bekleyelim

    env.close()

except Exception as e:

    print(f"\nError occurred:")

    print(f"Type: {type(e).\_\_name\_\_}")

    print(f"Message: {str(e)}")

    raise

finally:

    print("\nTest completed.")

Here are the results:

A screenshot of a computer game

Description automatically generatedA screen shot of a computer

Description automatically generated

The terminal output shows the testing of a trained DDPG (Deep Deterministic Policy Gradient) agent on the Reacher environment. As a result of this test, I can say that the environment has continuous observation and action spaces. Each agent receives 33 state variables as input and each agent can perform 4 continuous actions. There are no visual observations required.

Agent was tested for 5 episodes. Here are the ranks of episodes:

* Episode 1: 37.25
* Episode 2: 38.63
* Episode 3: 39.25
* Episode 4: 39.09
* Episode 5: 39.19

The scores indicate that the agent performs consistently well, maintaining an average score above 38 across all episodes. This demonstrates that: The trained agent has learned an effective policy for controlling the robotic arm. Performance is stable with little variance between episodes. Agent contacts with the target locations successfully.

Those models were trained by repository owner, that’s why it works properly. Then I tried to train by myself.

# Training

A screenshot of a computer generated image

Description automatically generated

qwe

Coding

1. Results
2. References