**Project 2 Report – DDPG**

**1 Learning Algorithm**

In this project we are using a model-free, off-policy actor-critic algorithm using neural networks as deep function approximators for high dimensional policies with continuous action spaces.

The way actor-critic works is that we set the actor as the approximator for the policy which selects action and the critics that evaluate policy according to the action taken. The algorithm we are using in this project is known as Deep Deterministic Gradient Policy (DDPG), which is like DQN but for continuous spaces, as opposed to discrete ones.

It is important to highlight there are three specific advances made to DQN which will be applied here to make Deep networks work and perform in a stable manner, namely the following:

1. We minimise correlation between samples - by incorporating training from samples taken from batches of a replay buffer trained off-policy
2. TD backups synchronisation – we perform updates to target values by using a trained Q-network under the temporal difference backups to achieve consistency through training.
3. Batch normalisation – tackles the problem of a phenomena known as internal covariate shift which brings up training time due to the reduction in the learning rate because of the increasing number of nonlinearities. To counter act this problem normalisation is implemented to each training mini-batch.
   1. **Model Architectures**

We used two different neural networks each for the Actor and critic but each one has two different network – the regular network and the target network. This is where we use soft updates to train the most up-to-date network – the regular network and use the target network – for predicting to stabilise the training in the regular network. The soft updates I used in the network was chosen to be Tau = 0.001.

**1.1.1 Actor Network**

For the Actor neural network, we used 2 hidden layers with 128 and 256 units respectively, with each hidden layer we used a rectified non-linearity but for the output layer we used tanh activation function to bound the actions – control pendulum problem. In addition, we added a dropout layer with p=0.20 to help elevate us from the problem with overfitting.

**1.1.2 Critic Network**

The critic network was like the Actor network with 2 hidden layers, 128 and 256 units respectively but with the difference of the output layer returns a value without the constraint of the tanh function imposed on the Actor network. In addition, we added a dropout layer with p=0.20 to help elevate us from the problem with overfitting.

**1.2 Hyperparameters**

To deal with continuous action space we have chosen Adam optimiser algorithm for each network – Actor and Critic with a learning rate of 10⁻⁴ and respectively. This tries to optimise a given action for any given state, which is how we can deal with the continuous space as opposed to a discretise state. For the target Q network, we used a L2 weight decay of 0 and a discount of gamma = 0.95. I trained minibatch sizes of 128 with a replay buffer size of .

**Summary - Hyperparameters**

BUFFER\_SIZE = int(1e6) # replay buffer size

BATCH\_SIZE = 128 # minibatch size

GAMMA = 0.95 # discount factor

TAU = 1e-3 # for soft update of target parameters

LR\_ACTOR = 1e-4 # learning rate of the actor

LR\_CRITIC = 1e-3 # learning rate of the critic

WEIGHT\_DECAY = 0 # L2 weight decay

TRAIN\_EVERY = 20 # How many iterations to wait before updating target networks

NUM\_AGENTS = 20 # How many agents are there in the environment

**2 Plot of Rewards**

Line chart

Description automatically generated

**3 Ideas for Future Work**

**3.1 Guided policy search (GPS)**

To overcome many of the challenges that comes to bear when using Actor-critic framework, I would explore the Guided Policy Search (GPS) algorithm, which Levine, Finn, Darrell and Abbeel have used in their paper End-to-end training of deep visuomotor policies (1).

GPS is an algorithm which transforms policy search into a supervised learning problem by breaking it down into three phases:

1. Uses trajectories to find locally linear approximations of its dynamics by utilising the full-state observation.
2. Then finds the optimal policy along these trajectories
3. Uses supervised learning to train a non-linear policy to reproduce the state-to-action mapping of the optimised trajectories.

**References**

1. Levine, Sergey, Finn, Chelsea, Darrell, Trevor, and Abbeel, Pieter. End-to-end training of deep visuomotor policies. arXiv preprint arXiv:1504.00702, 2015.