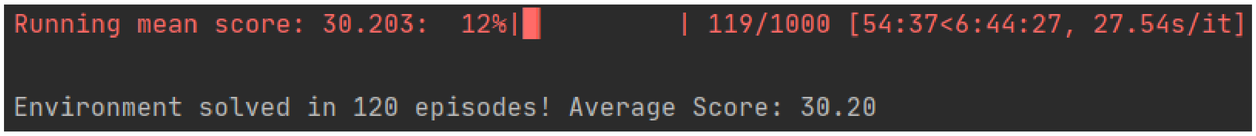
Deep RL Project 2 - Continuous Control

Environment Type: 20 agents
Algorithm: Deep Deterministic Policy Gradients (DDPG)
Episodes required to solve environment: 120

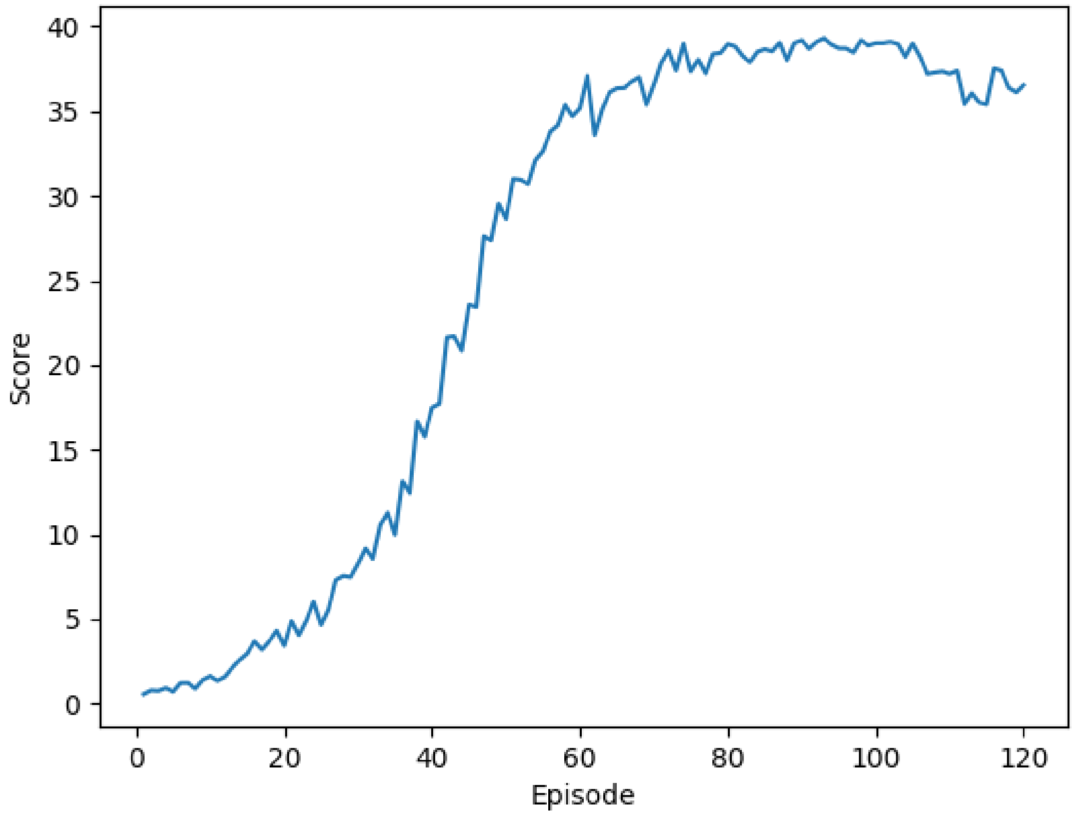


```
Running mean score: 30.203:  12%|█         | 119/1000 [54:37<6:44:27, 27.54s/it]

Environment solved in 120 episodes! Average Score: 30.20
```

Note: The running mean score is calculated by averaging the scores of all the agents over the last 100 episodes.

Training Score Plot

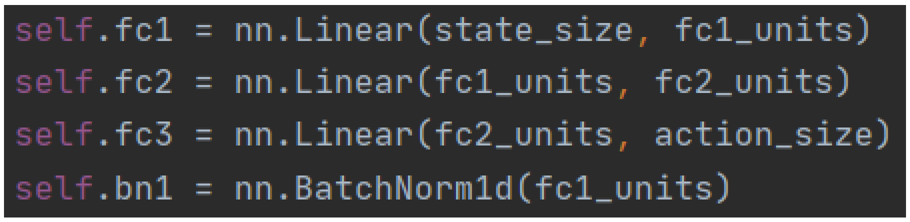


DDPG Hyperparameters

Hyperparameter	Description	Value
BUFFER_SIZE	Determines the maximum size of the replay buffer to store past experiences	1e6
BATCH_SIZE	Specifies the mini-batch sized used for learning updates	256
GAMMA	The discount rate on historical rewards	0.99
TAU	The soft update parameter, controlling the degree to which the actor and critic target networks are updated during a learning step	1e-3
LR_ACTOR	The learning rate for the actor model optimiser	2e-4
LR_CRITIC	The learning rate for the critic model optimiser	2e-4
WEIGHT_DECAY	L2 regularisation on model weight parameters to reduce overfitting	0

Neural Network Architectures

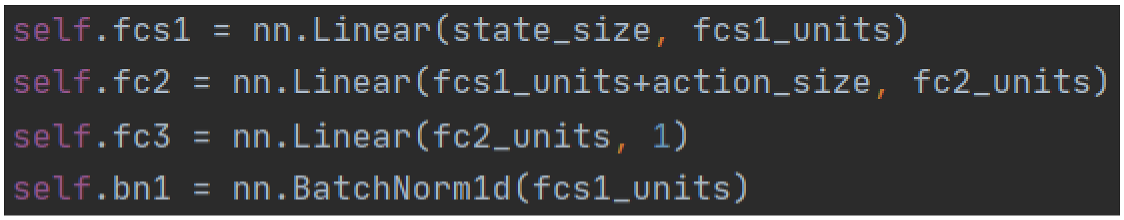
Actor



```
self.fc1 = nn.Linear(state_size, fc1_units)
self.fc2 = nn.Linear(fc1_units, fc2_units)
self.fc3 = nn.Linear(fc2_units, action_size)
self.bn1 = nn.BatchNorm1d(fc1_units)
```

- State_size = 33
- Action_size =4
- Fc1_units = 128
- Fc2_units = 128
- Activations = ReLU (fcs1), ReLU (fc2), Tanh

Critic



```
self.fcs1 = nn.Linear(state_size, fcs1_units)
self.fc2 = nn.Linear(fcs1_units+action_size, fc2_units)
self.fc3 = nn.Linear(fc2_units, 1)
self.bn1 = nn.BatchNorm1d(fcs1_units)
```

- state_size = 33
- fc1_units = 128
- fc2_units = 128 + 4
- Activations = ReLU (fcs1), ReLU (fc2), None

Note: The state and action vectors are input at layers fcs1 and fc2 respectively. This allows the fcs1 to specialise in generating state representations prior to being combined with the action vector.

Further Work/Improvements

SINGLE AGENT ENVIRONMENT

The training script and learning algorithm could be run “as is” on the single agent variant of the Reacher environment. This would provide useful insight into the differences in learning performance without the advantage of parallelised workers.

PPO IMPLEMENTATION

An implementation of the PPO algorithm was implemented in ppo_agent.py but has not been integrated with a training script as an option to use, nor have its hyperparameters been tuned. Given the algorithm’s success in parallelised learning environments, it might prove a viable approach for the Reacher environment.

SINGLE ACTOR-CRITIC NETWORK

Combining the actor and critic into a single network would ensure that the actions and state-values generated relied on representations that were applicable to both objectives. This may help improve the model’s ability to generalise, reducing convergence time during training.