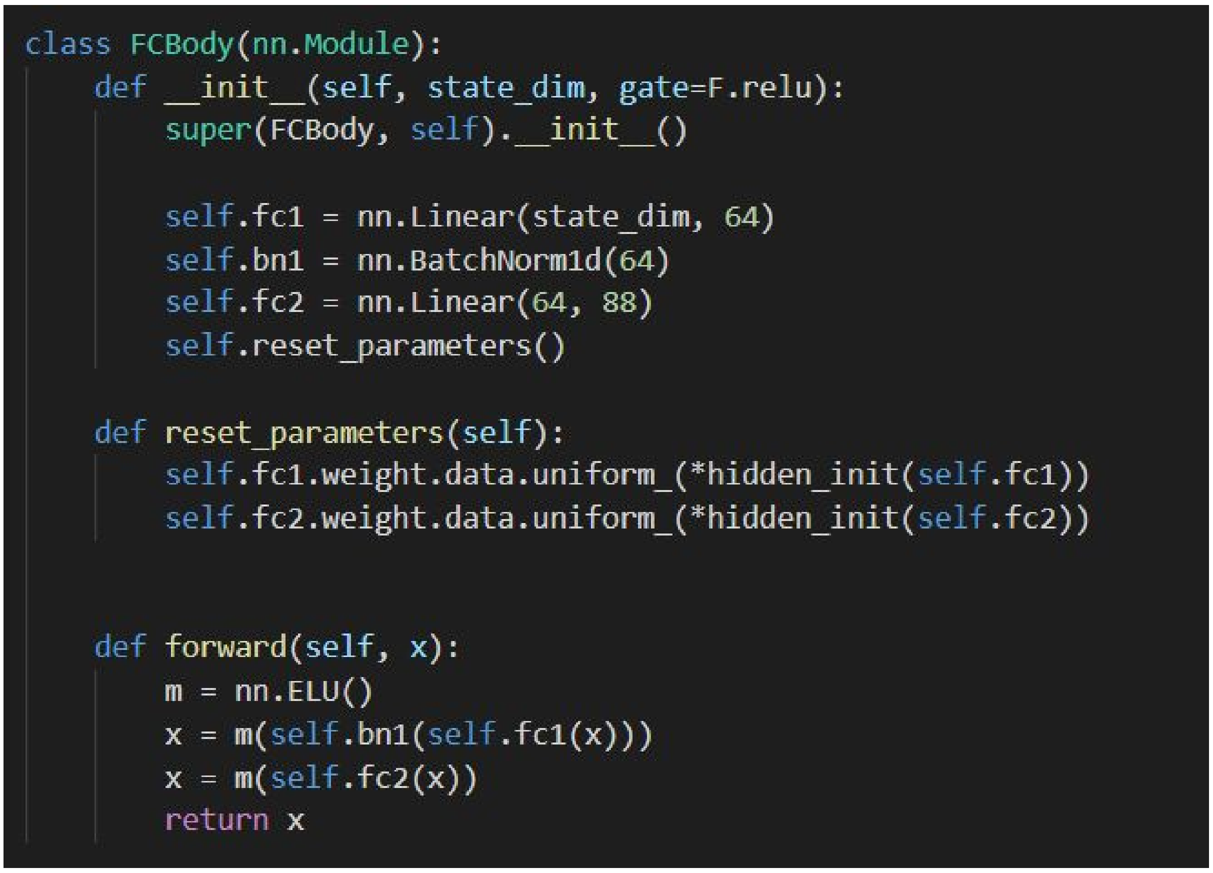
Report

Project 2 Control

In order to create this project we based in the code
https://github.com/ShangtongZhang/DeepRL

we did some adjustems because the default implementations does not work using the current Unity environment and also changed the default neural network.

in this case we use different activation function like elastic linear units and also batch normalization.



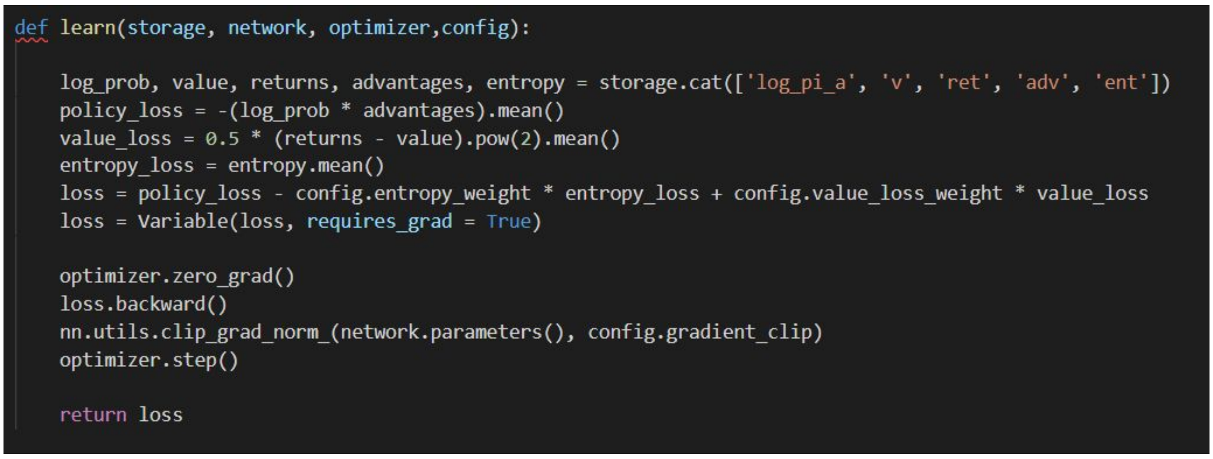
```python
class FCBody(nn.Module):
    def __init__(self, state_dim, gate=F.relu):
        super(FCBody, self).__init__()

        self.fc1 = nn.Linear(state_dim, 64)
        self.bn1 = nn.BatchNorm1d(64)
        self.fc2 = nn.Linear(64, 88)
        self.reset_parameters()

    def reset_parameters(self):
        self.fc1.weight.data.uniform_(*hidden_init(self.fc1))
        self.fc2.weight.data.uniform_(*hidden_init(self.fc2))


    def forward(self, x):
        m = nn.ELU()
        x = m(self.bn1(self.fc1(x)))
        x = m(self.fc2(x))
        return x
```

In the case of training the agent, i did some changes because there are some problems creating two unity agents at the same time so we cannot wrapper the environment as in the base code so we created two methods inside RFutils.py which allows to collect the episodes and training.



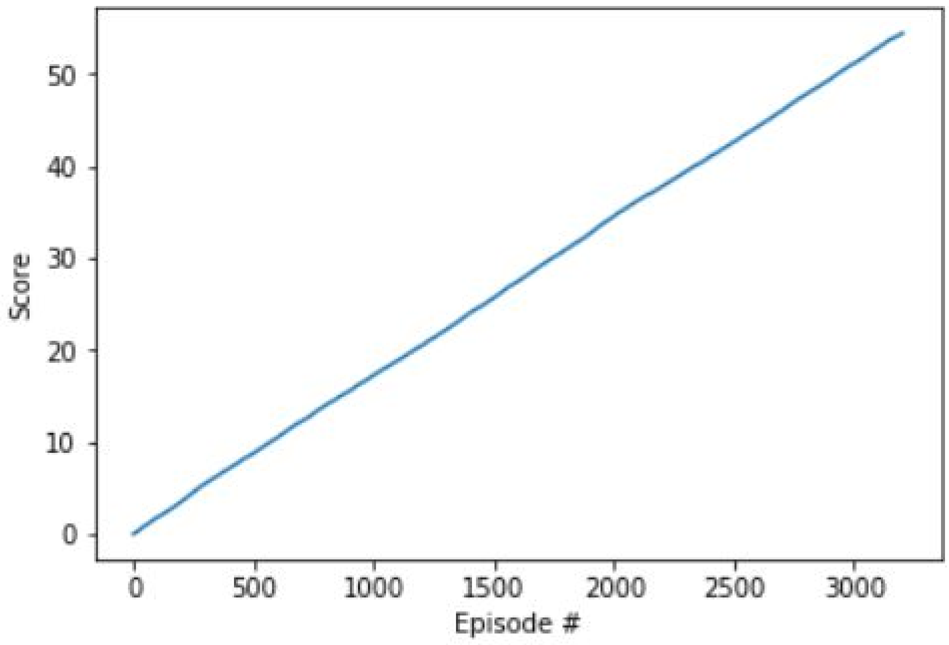
```
def learn(storage, network, optimizer,config):

    log_prob, value, returns, advantages, entropy = storage.cat(['log_pi_a', 'v', 'ret', 'adv', 'ent'])
    policy_loss = -(log_prob * advantages).mean()
    value_loss = 0.5 * (returns - value).pow(2).mean()
    entropy_loss = entropy.mean()
    loss = policy_loss - config.entropy_weight * entropy_loss + config.value_loss_weight * value_loss
    loss = Variable(loss, requires_grad = True)

    optimizer.zero_grad()
    loss.backward()
    nn.utils.clip_grad_norm_(network.parameters(), config.gradient_clip)
    optimizer.step()

    return loss
```

The results are shown in the image below


Score
Episode #