PPO exercise report

Marks Osipovs[1]

[1] Johannes Kepler University Linz, Altenberger Str. 69, 4040 Linz, Austria
k01428789@students.jku.at

1 Code

We were provided with a code skeleton where we needed to implement several functions and classes:

1. `ActorNet` – a class for the actor. We had to implement both the model and the forward pass.
2. `CriticNet` – a class for the critic. We had to implement both the model and the forward pass.
3. `ActorCriticNet` – a class that combines the actor and the critic. We had to implement the constructor, the action and the evaluation.
4. `Agent` – a class for the agent. We had to implement the constructor, the action, and the training procedure.
5. `Runner` – a class for the runner. We had to implement the running procedure.
6. Hyperparameters – we were tasked with selecting the best hyperparameters.

The code skeleton was very similar to the example code that was provided to us in the lecture, therefore, we decided to use it as a starting point to our approach. It proved to be very effective: with minimal changes to the `ActorNet`, `ActorCriticNet`, and the `Runner` we were able to get the code to run. The changes involved adapting the code to our environment and fulfilling the task requirements, e.g., using a normal distribution in the `ActorCriticNet` instead of categorical distribution.

2 Hyperparameter search

The problems started appearing when attempting to train the model for prolonged periods of time in order to obtain a good result. In the end, it took us several days to find the hyperparameters that were able to produce a good model.

Table 1. Initial hyperparameters.

Hyperparameter	Value
Episodes	400
Epochs	400
Hidden size	64
Learning rate	0.003
Discount	0.99
Reward scale	1.0
Batch size	1000
Min transitions	2000
Capacity	10000
Use buffer reset	True
Epsilon clip	0.2
Loss scales	0.5, 1.0, 0.01
Betas	0.9, 0.999
Weight decay	0.0001
Checkpoint interval	100

The initial hyperparameters were taken from the example exercise and can be seen in Table 1.

2.1 Model 1 – Model does not learn

After running this model for 400 episodes, it became evident that this model won't be able to learn anything, as the rewards never went above -100 and their mean was at around -150. We've tried several different approaches to fix this.

2.2 Model 2 – A score of 160 in just 400 episodes

After trying many different hyperparameters and their combinations, reducing the number of epochs from 400 to 8 and increasing the maximum number of steps in an episode from 400 to 1600[1], we were able to get a model that quickly was able to achieve good scores, but also had very many bad scores, i.e., a model with very high variance. Its reward progression during training can be seen in Fig. 1. Despite the high variance, this model was able to achieve a score of 160 on the evaluation server. However, we considered that luck and were not happy with the result.

[1] The `max_episode_steps` setting is not a part of hyperparameters of the task but is instead set during the creation of the gym. The default is 400.

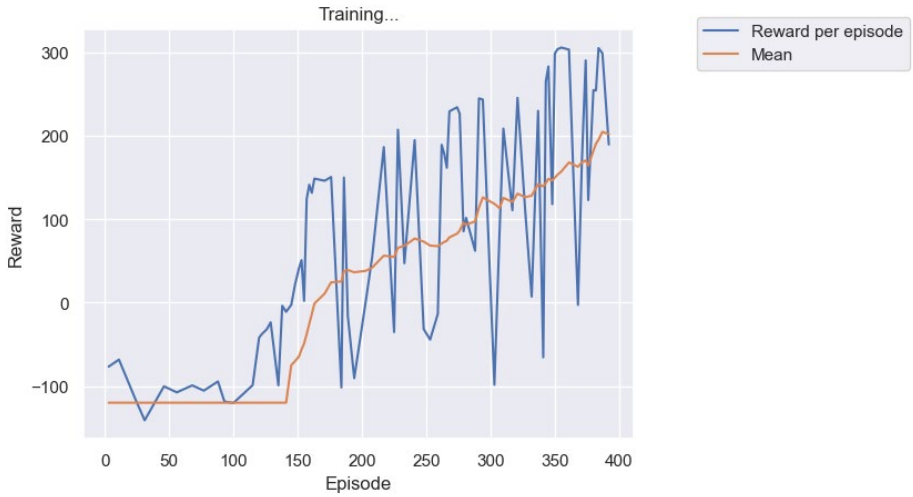

Fig. 1. Rewards during training for a model with 8 epochs and 1600 maximum steps per episode.

2.3 Model 3 – NaNs

The training of model 1 and model 2 was erroneously done with rendering enabled (render = True), with a very large batch size of 1000 and with 1600 maximum steps per episode. The combination of these three facts lead to very long learning times (i.e., 2 hours for 400 episodes), which is why the training of model 2 was limited to just 400 episodes.

Therefore, my next step for model 3 was to increase the training time by running the training for 2000 episodes. However, in the episode 471 (reproduced three times) the procedure crashed as the μ tensor started to consist of NaNs, rendering mathematical operations on it impossible and ending the training procedure prematurely.

We've tried several different hyperparameters in order to get rid of the issue, i.e., reducing the learn rate, the reward scale, or trying a different loss scale. The exact combinations can be viewed in Table 2. In the end, a model with the loss scales of (1.0, 0.5, 0.01), a learning rate of 0.003 and a reward scale of 1.0 proved to be the best. Furthermore, we reduced the maximum steps per episode back to the default 400, and reduced buffer capacity from 10.000 to 2.000.

Table 2. Some results of finding a model that doesn't result in NaNs during training.

Loss scales	Learn rate	Reward scale	Status	Min[2]	Mean[2]	Max[2]
1.0, 0.5, 0.01	0.003	1.0	Pass[3]	-43.38	209.87	317.01
1.0, 0.5, 0.01	0.0003	1.0	Fail (ep. 260)			
1.0, 0.5, 0.01	0.003	0.1	Pass[3]	-99.04	-5.62	278.40
0.5, 1.0, 0.01	0.003	1.0	Fail (ep. 149)			
0.5, 1.0, 0.01	0.003	0.1	Fail (ep. 171)			
0.5, 1.0, 0.01	0.0003	1.0	Fail (ep. 332)			

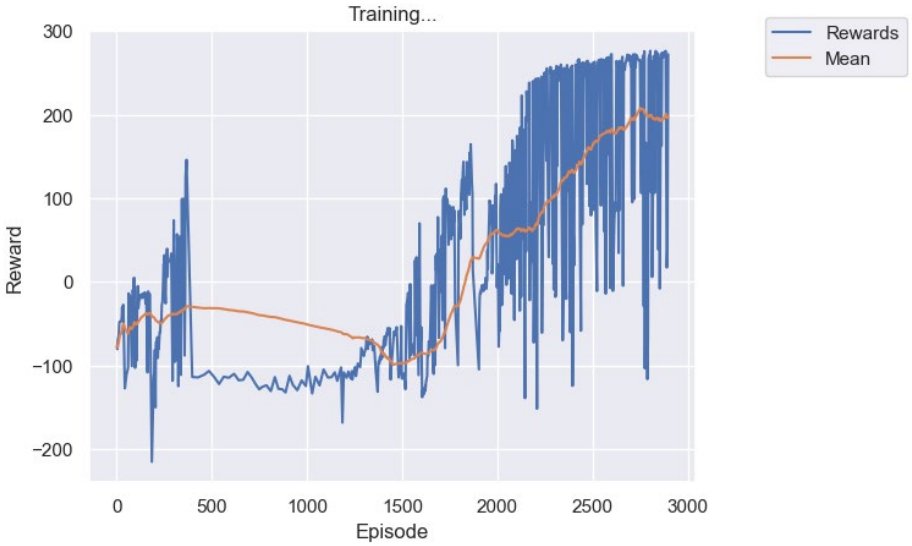

Fig. 2. Training performance of model 3.

After finding the hyperparameters that didn't cause the training to crash, model 3 was tested to run for 10.000 episodes, but it wasn't able to produce consistent results. As can be seen in Fig. 2, it had a high score ceiling with many episodes getting scores of 270 to 280, but equally a low score floor with many episodes getting a score of -100 to 100.

[2] Of the last 100 rewards of the run
[3] Pass = ran for 3000 episodes without a crash

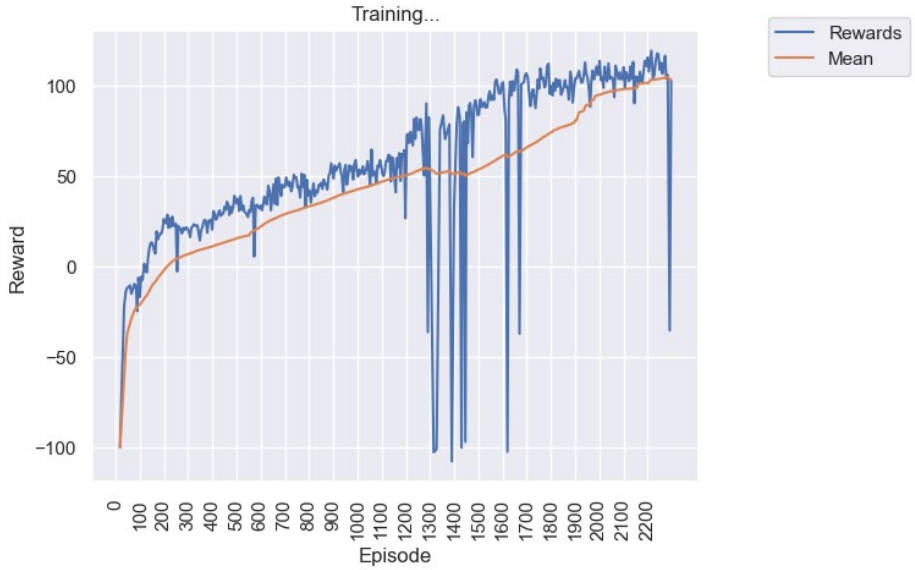

Fig. 3. Learning rate = 0.0003, hidden size = batch size = 128.

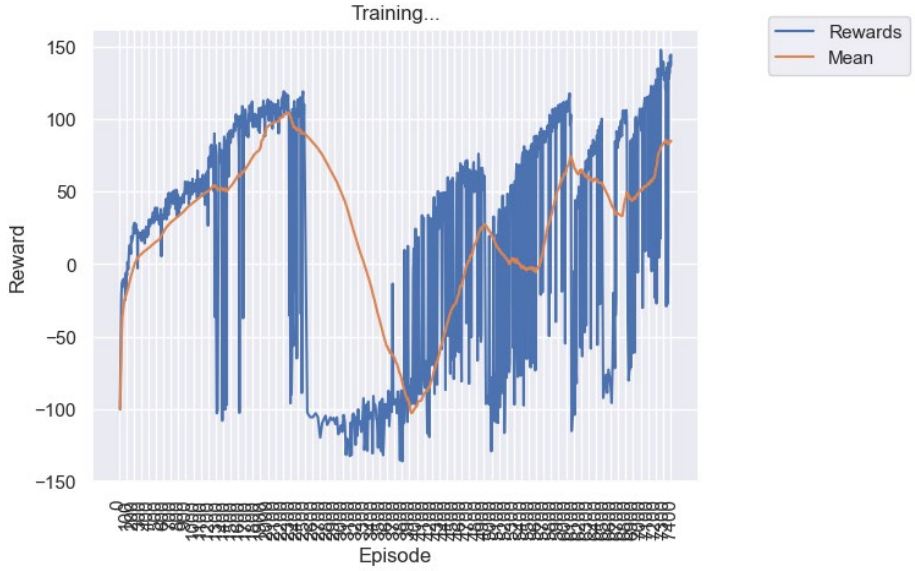

Fig. 4. Same as Figure 3, but the training continued for 8000 episodes.

Table 3. Final hyperparameters.

Hyperparameter	Value
Episodes	2300
Epochs	8
Hidden size	128
Learning rate	0.0003
Discount	0.99
Reward scale	1.0
Batch size	128
Min transitions	2000
Capacity	2000
Use buffer reset	True
Epsilon clip	0.2
Loss scales	1.0, 0.5, 0.01
Betas	0.9, 0.999
Weight decay	0.0001
Checkpoint interval	100

2.4 Model 4 - Breakthrough

At this point, we were hitting a dead end, so we tried a completely new approach with hyperparameters that were based on the ones from the provided solution as well as hints from other students to use the same low value for both the hidden size and the batch size. We reduced the learning rate to 0.0003 (to prevent NaNs) and tried training with batch size = hidden size = 64 and 128. The model with 128 proved to be very successful, as can be seen in Fig. 3. For the first time, a model was training, and the results of bad episodes were increasing steadily as well, keeping the variance low. We hit a peak at around episode 2300, after which the performance dropped, and it took the model a long time to get good scores again. However, the variance increased after the drop and never improved again (see Fig. 4.).

3 Final result

The model used for the submission is model 4 with the hyperparameters that can be seen in Table 3. Training stopped at episode 2300, before the performance drops. The model achieved a score of 312 when evaluated locally and a score of 280 on the submission server.