**Experiment design:**

We designed the experiment to study the performance of policy gradient approach, particularly our implementation of the REINFORCE method, and the impact of hyperparameters including the learning rate, the number of neurons in each fully connected layer of the neural network (fc\_dim), and the maximum time allowed for each episode (time threshold). The effects of these hyperparameters are studied across 7 runs:

* Run 1: learning rate 0.0003, fc\_dim 256, time threshold 40
* Run 2: learning rate 0.0005, fc\_dim 256, time threshold 40
* Run 3: learning rate 0.0007, fc\_dim 256, time threshold 40
* Run 4: learning rate 0.0005, fc\_dim 128, time threshold 40
* Run 5: learning rate 0.0005, fc\_dim 64, time threshold 40
* Run 6: learning rate 0.0005, fc\_dim 256, time threshold 30
* Run 7: learning rate 0.0005, fc\_dim 256, time threshold 20

The effects of learning rate is studied across runs 1, 2 and 3; fc\_dim across runs 2, 4, 5; and time threshold across runs 2, 6, 7.

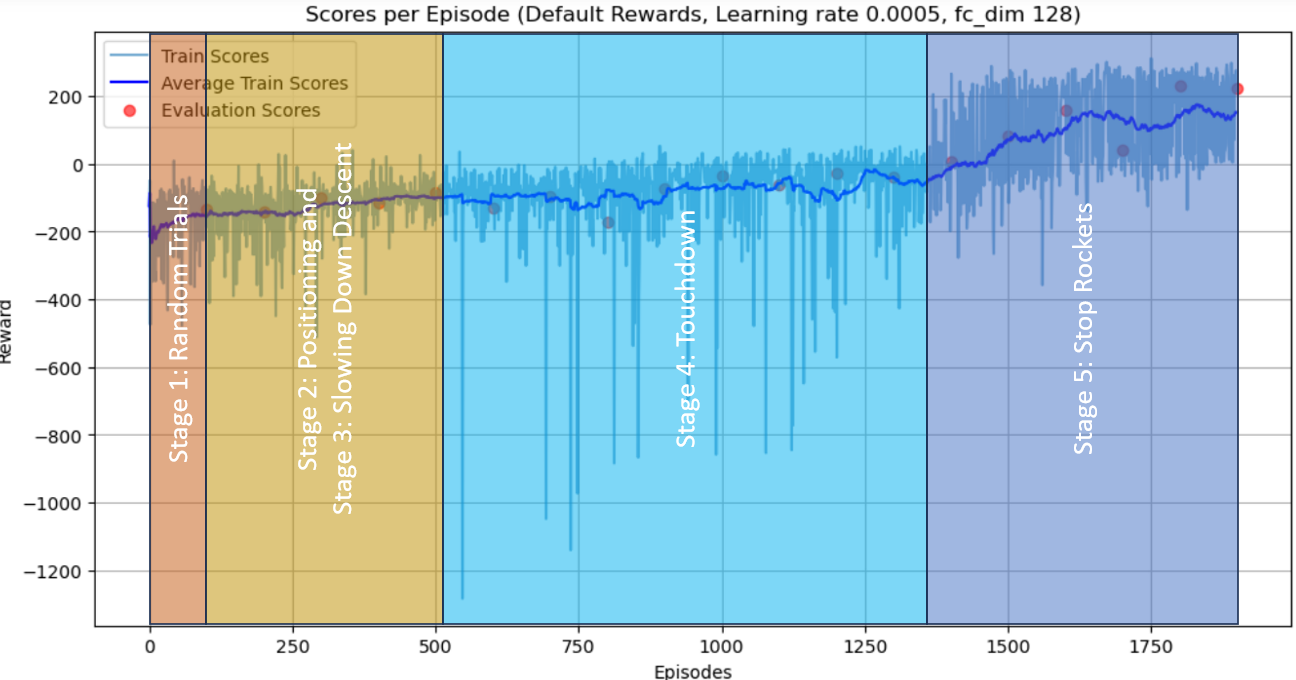
Initially, we used a standardised design of 3000 training episodes in the training process. In addition, to monitor the performance of the agent, we performed an evaluation over 10 episodes after every 100 training episodes, where the function model.learn() is not called.

For each training episode, a few key metrics are recorded: training episode score, average training score over the last 50 training episodes, and a highscore that is the highest recorded score in the run till that point. For each evaluation, the scores of each of the 10 evaluation episodes, and the average are recorded. At the end of the run, The training scores and average evaluation scores are plotted against the training episodes. The average training scores curve gives a smooth interpretation of the training process. In addition, the videos of the agents performance over five episodes are recorded. Here, success is measured by the craft landing and coming to a stop in between the flags.

By the third run, it is noticed that fixing 3000 iterations is not a good idea as the performance of some agents start to degrade with more episodes. An early stopping mechanism is introduced where the performance of the agent is artificially evaluated after every 100 training episode and evaluation stint. The run will not continue if the average training or evaluation scores are observed to be beyond 200 and have peaked. Not only does this improve the time and space complexity of running the training algorithm, we are also able to evaluate the performance of the agent when it is at its best.

**Main findings of the training process:**

Over the course of the 7 runs, we have observed a few clear stages in the agent’s learning process. We call it the 5 stage learning framework.



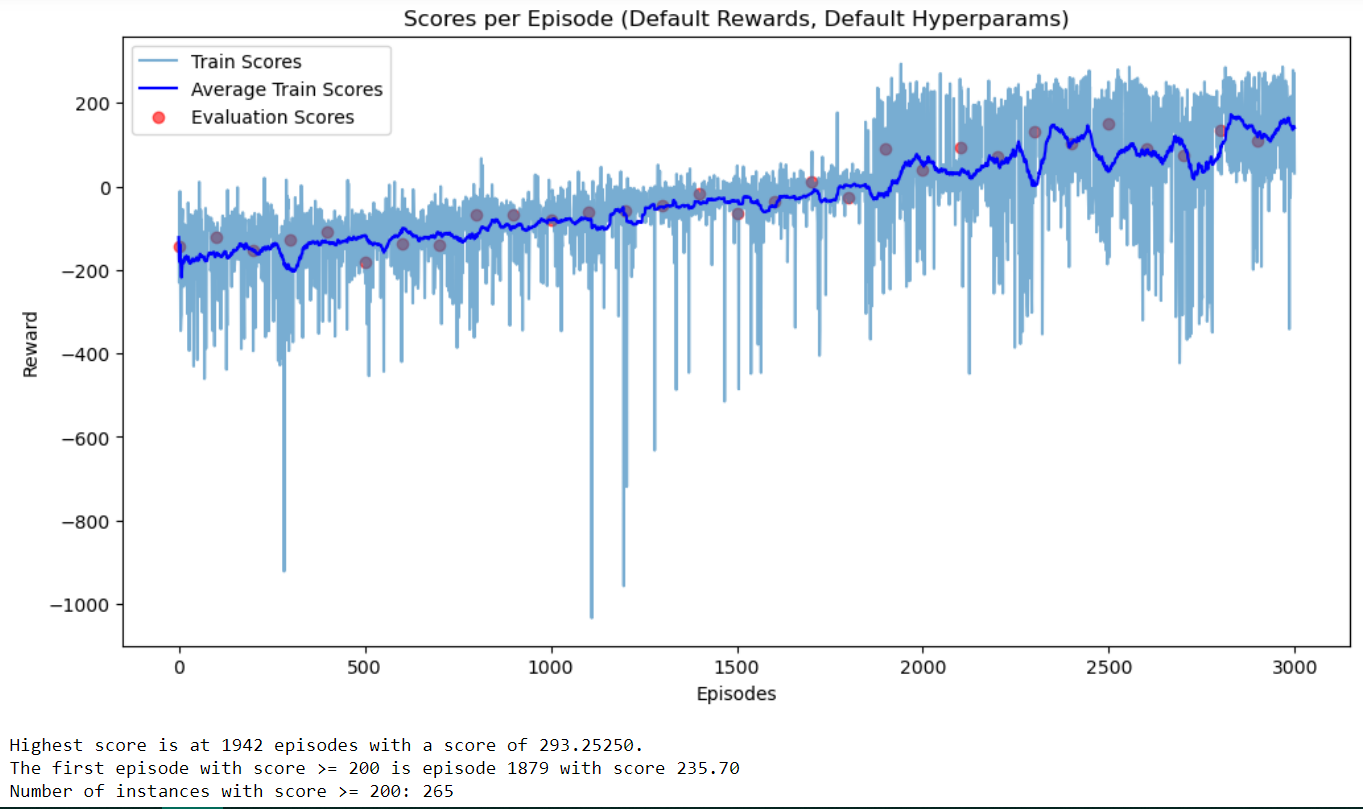
The first stage is random trials, where the agent tries random actions at each step. This stage is marked by a large variance in episode scores. Then, it enters the second stage where it focuses on its positioning to maximise rewards by moving towards the goal; and the third stage, which involves slowing down during the descent. These two stages are represented by a gradual increase in scores, and additionally for the third stage, the time taken per episode is also observed to increase slightly. The fourth stage is the touch down. Here, since the agent does not know when to stop firing its rockets, this is indicated by a huge decrease in score (from the waste of fuel), and an increase in time taken per episode (longer episodes, and to process the larger state, action and rewards arrays). Finally, in the final stage, the model will learn through random action exploration that stopping the rockets when it has landed significantly improves the score. This stage is indicated by a huge increase in individual and average scores, and a huge decrease in time taken per episode. With this framework in mind, we are able to determine the stage each run is in at each episode, and therefore compare the learning process across the different runs.

However, it is wise to remain cognisant that the boundaries of these stages are not definitive (may be blurred). Furthermore, the learning process can be highly stochastic even with the same hyperparameter configuration. The advance to the next stages in the framework, while possibly affected by hyperparameter configurations, is nevertheless highly unexpected and dependent on chance. Thus, the scores and success rate of each individual run only provide a rough gauge of the performance we can expect.

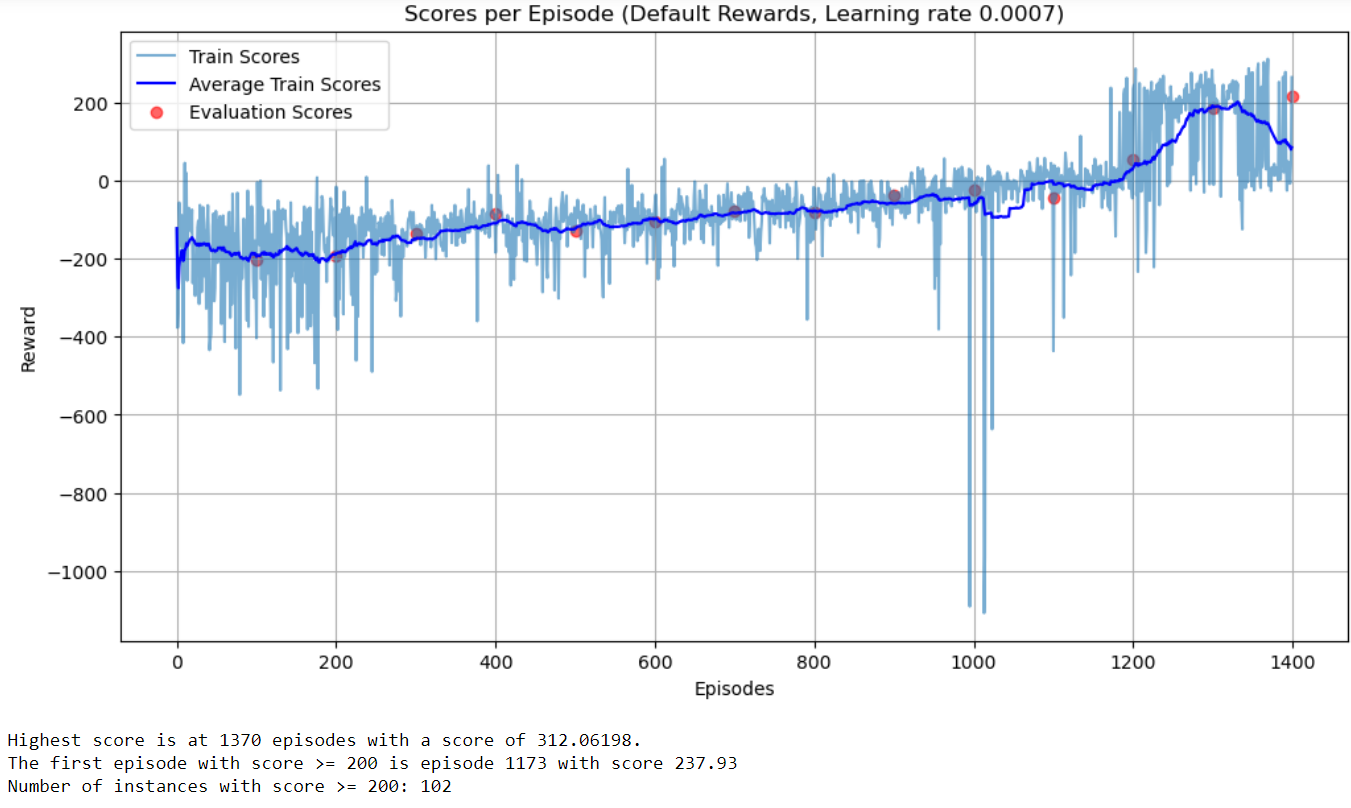
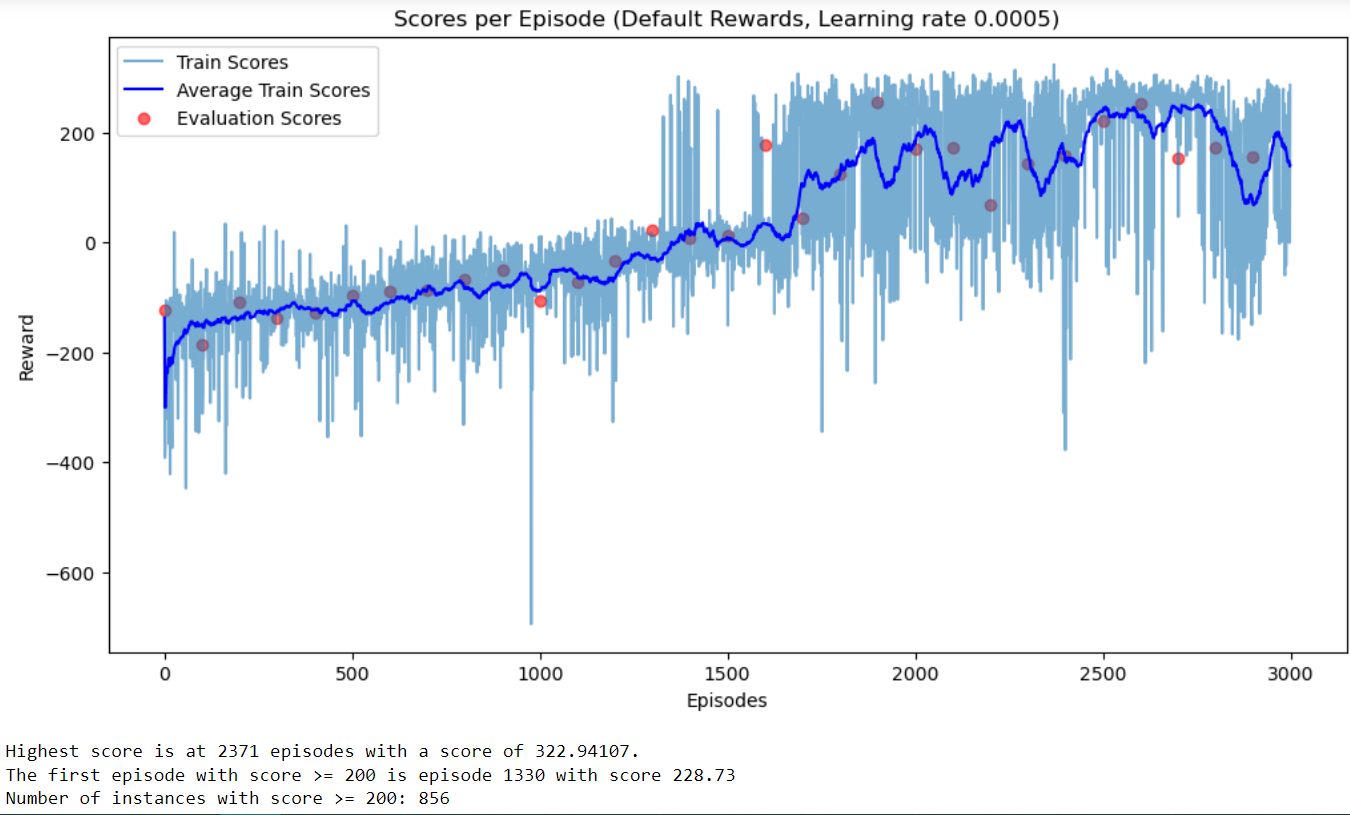
**Analysis of the effect of tested hyperparameters:**

Learning rate:

The learning rate hyperparameter controls the rate of update of parameters in the neural network, using the Adam optimizer.



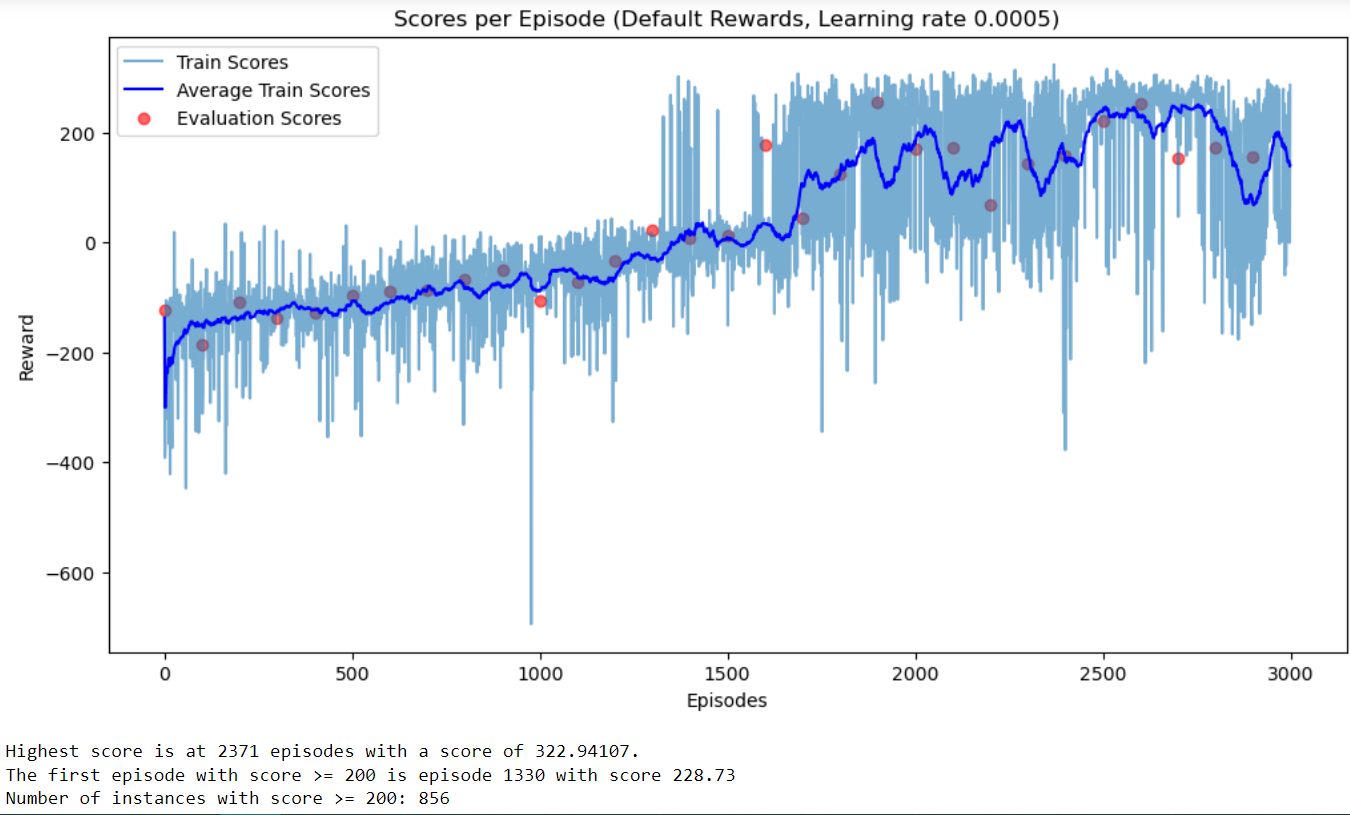
Comparing the results across runs 1, 2 and 3, we observe that for run 1 above, a learning rate of 0.0001 is too low and as a result the model failed to converge within the upper bound of 3000 episodes. Here, convergence can be defined as having a definitive peak in scores, or an average training or evaluation score of at least 200. The slow jagged increase in average training scores later in the run indicates that the craft only managed to stop in a portion of the episodes, and that the agent is transitioning between stages 4 and 5 of the learning framework.



Both runs 2 and 3 above managed to converge beyond the score of 200. For run 2, this occurred at the 2000th episode, evidenced by the high average evaluation score of 254.71. Run 3 converged earlier at the 1300th to 1400th episode mark, with evaluation scores of 187.41 and 216.36 respectively. This shows that the model with a lower learning rate of 0.0005 converges later and has slightly better performance than a higher learning rate of 0.0007, which is in line with our expectations. Comparing the performance of the ship from the video clips, we can see that the ships from run 1 exhibit more risky behaviour of landing quicker, and they also have a tendency to fire the rocket after touch down. The ships from runs 2 and 3 on the other hand all landed steadily, and stop firing their rockets after touchdown. With success defined as stopping the ship in between the flags, the success rates of runs 1, 2 and 3 over the five episodes in the videos are 2, 1 and 3 respectively. However, the success rate of run 2 is underreported due to the performance of the model having diminished after convergence. This can be due to the learning of wrong behaviour or unlearning, which reduces performance.

Fc\_dim:

The fc\_dim hyperparameter is a measure of the complexity of the neural network, in terms of the number of neurons in each of the two hidden fully connected layers. A more complex network is expected to be able to learn better compared to a simpler model.



A graph with blue lines

Description automatically generatedA graph showing a line of blue and white

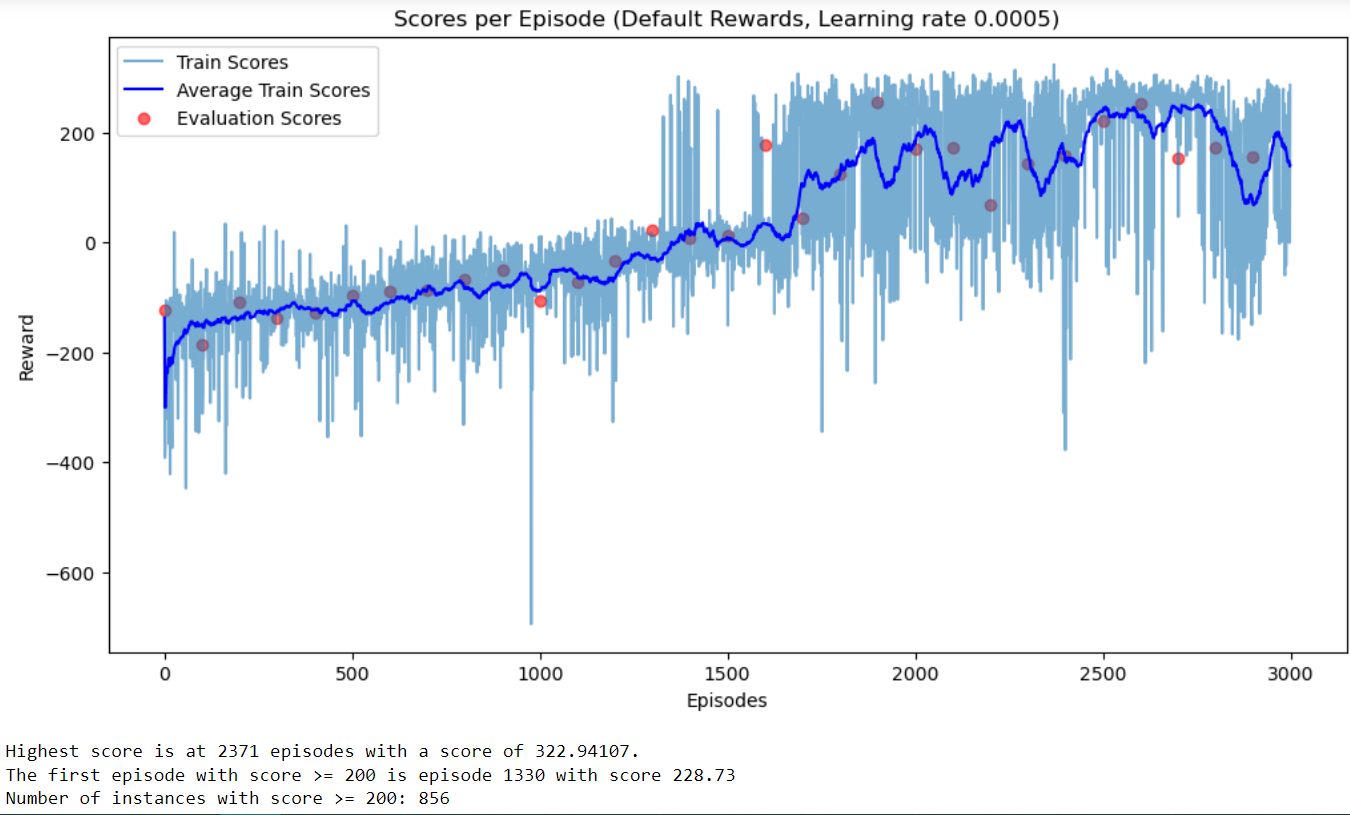
Description automatically generated with medium confidence

Comparing runs 2, 4 and 5 (fc\_dim of 256, 128 and 64 respectively), we can see that the model in run 4 exhibited convergence first at 1800 to 1900 episodes (with average evaluation scores of 228.89 and 222.95 respectively), followed by the model in run 2 at 2000 episodes (with an average evaluation score of 254.71), while lastly run 5 at 2900 episodes (with an average evaluation score of 189.20). This can be explained by the lower capacity of the neural network with fc\_dim 64 in capturing key trends in the state, action and rewards data, leading to poorer parameter estimation. However, as policy gradient approach is highly stochastic and unexpected, how early or late a model converges (and the best scores reached at convergence) could also be due to pure chance.

With the exception of the number of episodes required to converge, the general shape of the scores curves across the three runs are similar. As for the video playback, the ships in all three runs managed to stop firing the rocket after landing, exhibiting advanced stage 5 behaviour and showing that learning is thorough and complete. The ships also descended steadily, with very few crashes on impact. The success rates of runs 2, 4 and 5 over five episodes are 1 (past convergence), 4 and 3.

Time threshold:

The time threshold is a special hyperparameter specially implemented for this lunar lander problem. It stands for the maximum allowable time for each episode in seconds, and effectively discourages the ship from prolonging an episode by firing its rocket in an idle position. When the time threshold is met, the done variable is forcefully set to True and a reward of -50 is given as a punishment.



A graph showing a line of blue and white lines

Description automatically generated with medium confidenceA graph showing a number of scores

Description automatically generated with medium confidence

Comparing runs 2, 6 and 7 (time threshold 40, 30 and 20 respectively), we observe convergence first for run 2 (at 2000 episodes, with an average evaluation score of 254.71), followed by run 6 and 7 at 2500 episodes, with average evaluation scores of 194.40 and 223.32 respectively. This implies that a larger time threshold can accelerate model convergence, which is possible considering that a longer episode provides more data for the neural network model to train on, and there is also a higher likelihood of more random exploratory actions from a larger number of steps. In addition, the video evidence highlights that the ships from run 2 learned the best, as most of them stopped firing their rockets after touchdown, while the ships from runs 6 and 7 all tend to keep firing their side rockets and move themselves to the left. The success rate is the highest for run 6 with three out of five ships successfully stopping within the flags (despite firing their side rockets), followed by runs 2 and 7 with one out of five each.