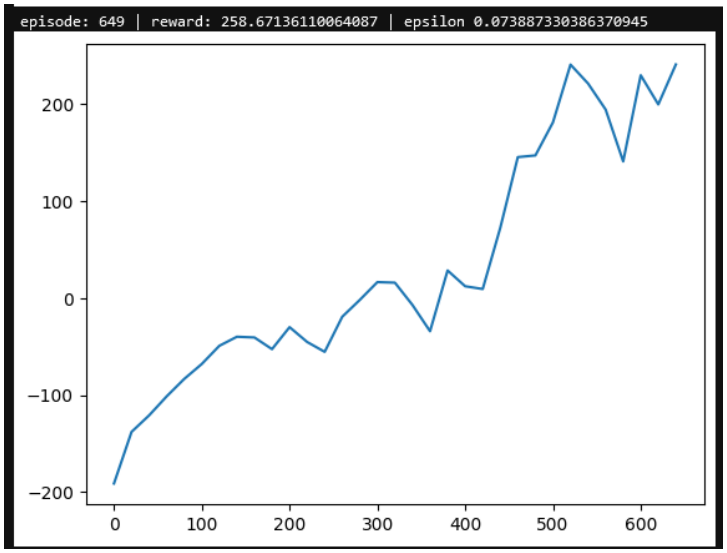
Voor het initialiseren van de parameters van mijn netwerk heb ik gekeken naar de waardes die stable baselines als default gebruikt (https://stable-baselines.readthedocs.io/en/master/modules/dqn.html)

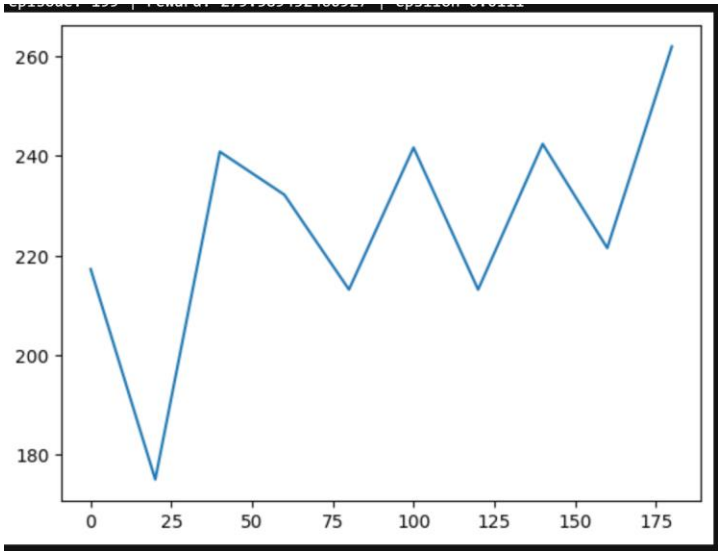
```
DDQN = Agent(policy = EpsilonGreedyPolicy(alpha = 0.0005, epsilon = 1, epsilon_decay_rate = 0.996, min_epsilon = 0.01, state_size = 8, action_size = 4),
             tau = 0.01, gamma = 0.99,memory = Memory(capacity = 100000), n_episodes = 700, batch_size = 64, update_frequency = 4, env = env)
```

- Een hoge gamma is goed voor dit environment omdat die vooruit moet kijken om een mogelijke crash landing te vermijden.
- Een epsilon_decay_rate van 0.996 zorgt ervoor dat de epsilon ~0.01 is na 1000 episodes.
- Ik heb Huber loss gebruikt omdat het minder gevoellig is voor outliers

één van de problemen waar ik eerst tegen aan liep was dat mijn netwerk ongelofelijk lang duurde om te trainen (denk dagen) er was wel vooruitgang in reward gain maar het was erg langzaam. Ik ben toen gaan zoeken in documentatie en papers van DQNs en DDQNs waar dat aan zou kunnen liggen. Al snel vond ik dat ik er een parameter in mijn netwerk ontbrak: training_frequency. Na training frequency te hebben toegevoegd (elke 4 steps) convergeerde mijn netwerk veel sneller. Hij zat eerst vast op ~-40 avg reward na 10k episodes (hij zat vast in het gedrag om nooit te landen maar te blijven zweven), en nu convergeert die naar een policy van 200+ avg reward na ~700 episodes.

Het duurt in totaal een paar uur om het netwerk volledig te trainen.





Het getrainde model heeft een gemiddelde reward van 221 per episode over 200 episodes. Ik kan het model eventueel nog wat verder proberen te trainen om hem meer stabiel te krijgen maar mogelijk overfit die dan.