### Shared experience

Parallelised training is well-known is be beneficial in DL and DRL (Mnih et al., 2013; Silver et al., 2016; Lecun et al., 2015; Teng et al., 2019), to cope with the amount of data needed, and manipulating the training time, to obtain significant results. The main objective of parallelisation is to reduce training time, and where training of deep nets with hundreds of millions of weights and billions of connections between units took weeks two years ago, advances in parallelisation have reduced training time to a few hours (Lecun et al., 2015). Part of these advances in parallelisation comes from novel algorithms, which benefit from parallelisation other than reduced training time. These novel algorithms incorporate experience of the agent to update the function approximation in question, thereby reducing the amount of data needed, and this is exactly what PPO does.   
Parallelisation, at least in Unity, works by running concurrent training environments, either internally or externally. *External* refers to running multiple applications of the same environment, often used when training on GPU’s via cloud computing. *Internal* refers to having multiple environments, with the agents being linked to the same brain, within one application, see figure 19. Internal concurrent training is a way to utilise parallelisation when training on a CPU, and training using GPU’s isn’t a possibility (Teng, 2019).

*Figure 19 – Internal concurrent training*  
 *Four similar environments prior to initialisation, each with an agent, a target and static obstacles. All the agents are linked to the same brain, thereby gathering experience to the same buffer.*

(Butcher, 2018) outlines one trade-off when considering using parallelised training;

*“However, training your algo can be complicated. - If you try to 'parallize' an algo's training by executing the algorithm on multiple processing devices at once, you can get the wrong result because of the feedback loop between the algorithm and the environment. But if you don't do this and try "gradient-based training" you will end up with a huge amount of irrelevant experiences and good behaviours can be forgotten.”* (Butcher, 2018)

The previous section addressed how to raise the quality of the experience gathered, thereby reducing the amount of *irrelevant* experience collected.  
This section addresses the challenge surrounding the *feedback loop between the algorithm and the environment.*

Agents of concurrent environments benefit from the shared experience every time they finish an episode, and if the average length of an episode is long, potentially unbounded, sharing experience can become a burden. It is this burden which is referred to as the challenge surrounding the *feedback loop between the algorithm and the environment.*

A concrete example on this is presented shortly, but this study proposes a solution to this problem. The solution is to add the number of concurrent environments as part the curriculum, and thereby introduce the benefit of shared experience after a certain amount of learning has taken place.

**Cost of parallelisation**

**Added after the figure in another version:**

Running concurrent training environments, externally as well as internally, fills the same experience buffer with prior observations. The usual benefit is that training accelerate, because the share of favourable prior observations increases faster with more agents to harvest. However, it can of cause go the other way, if the environment contains too much complexity at the time being. Furthermore, if nothing prevents the agents from wandering around to infinity, a bad spiral unfolds. This spiral is powered by two events; firstly, the agents never benefit from sharing experience because they rarely finish an episode. Secondly, the shared experience is low quality of the wandering. In this way, sharing becomes a burden. This is referred to as the feedback-loop-effect by (Butcher, 2018) in the quotation below.

*“However, training your algo can be complicated. - If you try to 'parallize' an algo's training by executing the algorithm on multiple processing devices at once, you can get the wrong result because of the feedback loop between the algorithm and the environment. But if you don't do this and try "gradient-based training" you will end up with a huge amount of irrelevant experiences and good behaviours can be forgotten.”* (Butcher, 2018)

It is given space here, because it is indeed a real problem. However, clever design of the training process, as introducing gradual complexity through CL, limiting the number of steps etc., can often prevent the bad spiral to occur.