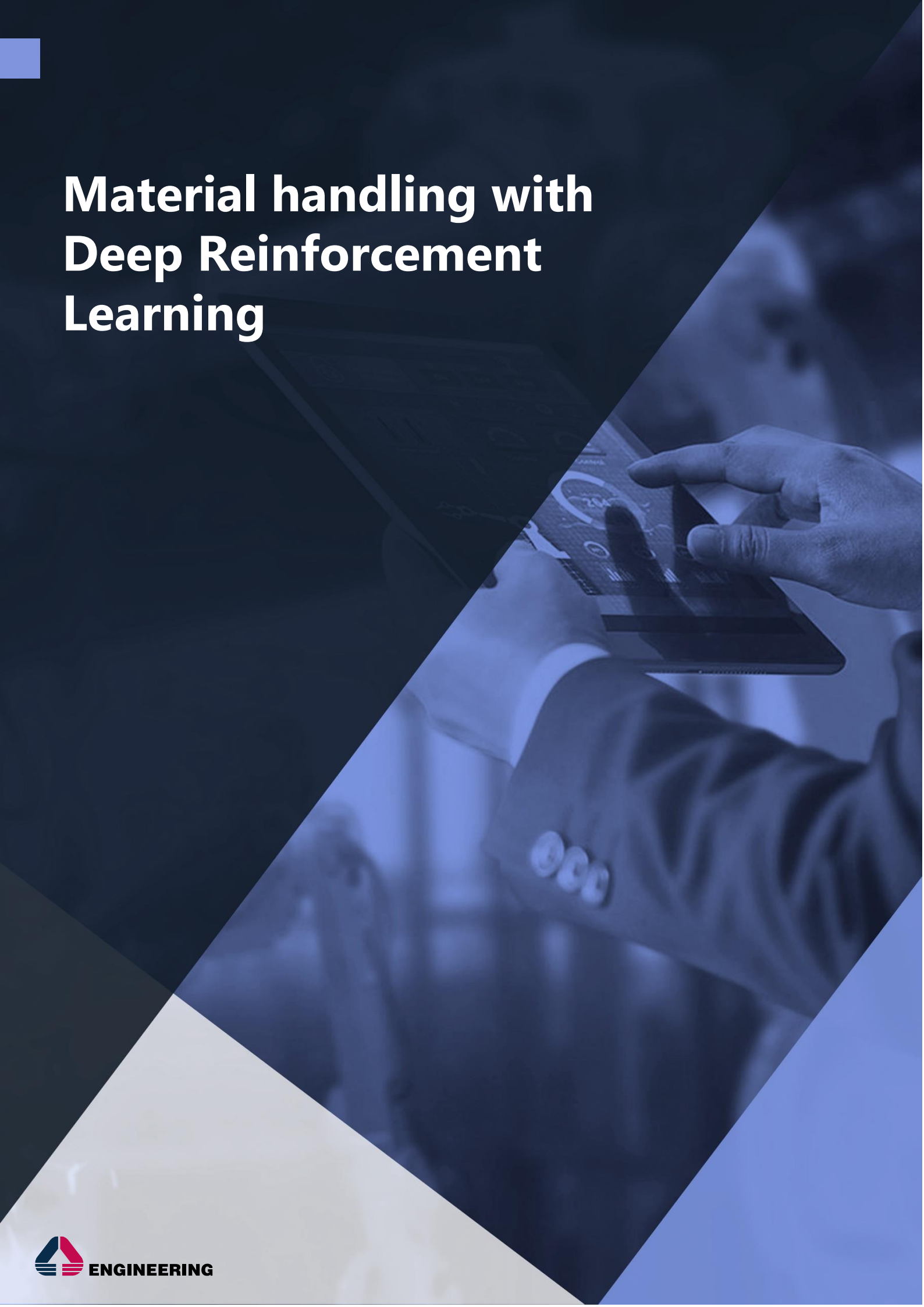
Material handling with Deep Reinforcement Learning



ENGINEERING

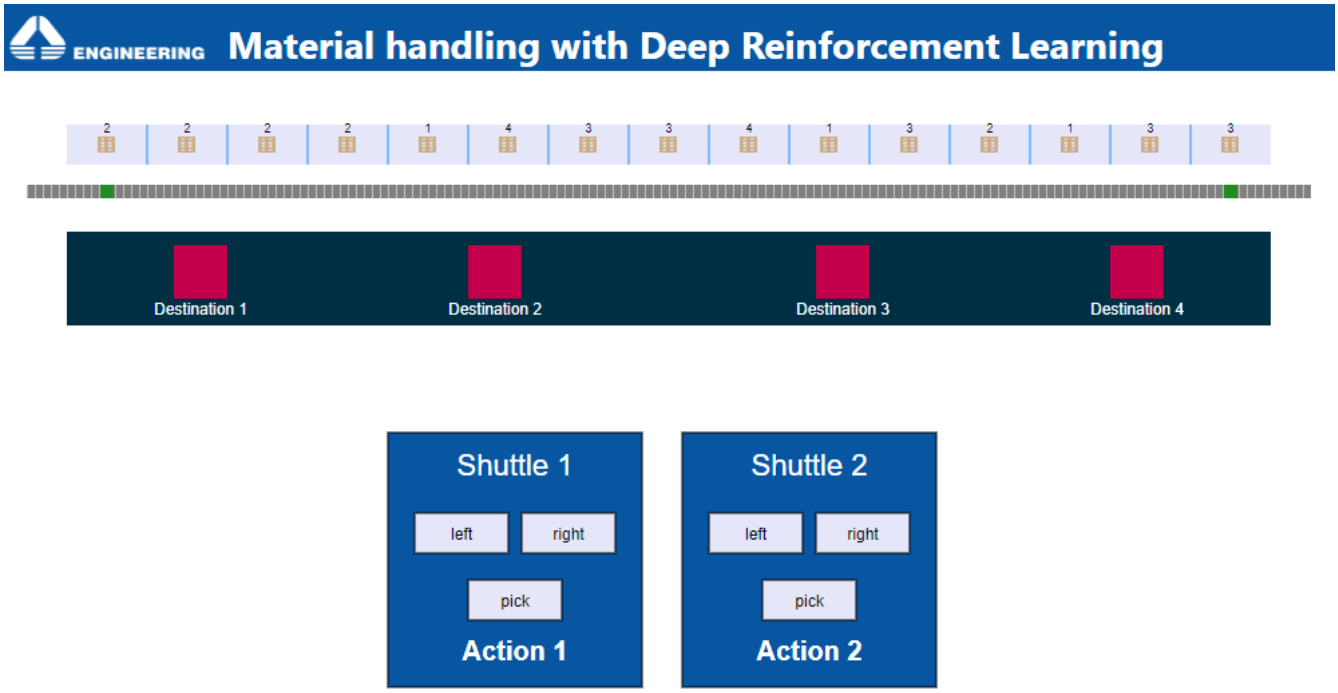
TABLE OF CONTENTS

1 INTRODUCTION 2
2 OBSERVATIONS AND ACTIONS 3
2.1 GLOBAL STATE AND SUB-STATE 3
2.2 ACTIONS 4
3 BRAIN CONCEPT 5
4 BRAIN TRAINING AND ASSESSMENT 6

1 INTRODUCTION

In this simulation model is proposed an application of Deep Reinforcement Learning (DRL) for material handling related to the manufacturing industries.
Material handling is the process of movimentating materials from place to another within a manufacturing plant. The costs for this activity are not negligible (such as labor, equipment, time and distance) and therefore material handling heuristics are continuously changed to meet production constraints and customer demand by adjusting resources accordingly. Thus, it is increasingly important to have a very flexible materials management system, which makes optimal decisions quickly in order to best handle uncertain situations.

In the example realised there are two shuttles responsible for transporting materials (one material at a time) sharing the same track. The materials are placed in the pallet rack and each is assigned a destination chosen from 4 destinations. Once the pallet has arrived at its destination, it remains at the destination for a fixed time, after which the destination will be free again. The objective is to transport the materials to their destinations in the shortest possible time, avoiding collisions between the two shuttles and considering that each destination can hold a maximum of 1 pallet at the same time. The creation of the policy to control the two shuttles was done with Microsoft Project Bonsai.



2 OBSERVATIONS AND ACTIONS

2.1 GLOBAL STATE AND SUB-STATE

In the proposed Brain, a global state (simulation state) and a sub-state have been defined. The global state contains all the information necessary for the brain, both for training and variables to be evaluated within functions such as the reward or termination function. The sub-state instead is a subset of the global state and contains only the input variables for the Reinforcement Learning model. Each variable is associated with its own discrete range of values and may be a single value or an array. The following table shows the variables used in the sub-state for training the model.

Variable	Domain	Type	Description
pallet_state	[0,4]	array[15] of discrete integer values	4 if pallet rack cell is empty, otherwise the number of the pallet's destination (0 -> destination 1, 1 -> destination 2, 2 -> destination 3, 3 -> destination 4)
shuttle1_start_position	[0,14]	discrete integer value	current position index of shuttle 1
shuttle2_start_position	[0,14]	discrete integer value	current position index of shuttle 2
destination	[0,1]	array[4] of discrete integer values	0 if the i-th destination is occupied, 1 if it is free
shuttle_is_delivering	[-1,13]	array[2] of discrete integer values	-1 if the shuttle is not transporting pallets, otherwise a discrete index of the final position of the shuttle to arrive at
distance	[-14,14]	discrete interger values	distance between the two shuttles (shuttle1_start_position - shuttle2_start_position)

The *pallet_state* array is used by the model to figure out which cells of the pallet rack contain pallets and their destinations through spatial mapping (the i-th cell of the array represents the i-th cell of the pallet rack). Note that the first and last two pallets in the pallet rack can only be transported to destination 1 or 2 and destination 3 or 4 respectively.
The variables *shuttle1_start_position* and *shuttle2_start_position* contain the discrete position of the two shuttles at each instant so that the model can keep track of them.
The *destination* array indicates with a spatial mapping whether the i-th destination is available or not, as each destination can contain a maximum of 1 pallet at a time.
The *shuttle_is_delivering* array allows the model to understand whether each shuttle is transporting a pallet and, if so, contains the shuttle's final destination. This is because during the transport of a pallet, the shuttle cannot interrupt the transport and the second shuttle must adjust its movements so as not to get in the way.
Finally, the *distance* variable indicates the discrete distance between the two shuttles calculated as the difference between the position of the first shuttle and the position of the second shuttle. This variable is useful to understand clearly when a collision occurs.

The remaining variables, which do not appear in the sub-state and are therefore global variables, are listed below:

- *collisions*: this variable is set equal to 1 if the *distance* between the two shuttles is less than or equal to 1 and therefore there is a collision. In this way, we want to guarantee a safety distance between the two shuttles that is at least 2 discrete positions (except when the two shuttles are in position 0 and 1 or 13 and 14, because the outer shuttle could not move further away than that. In this case the distance can be equal to 1). When the first (second) shuttle has to transport the pallet to the last (first) destination, then the minimum distance can also be 1, as the second (first) shuttle cannot go beyond the side limit of the track.

- *invalid_action*: the value of this variable becomes equal to 1 when one of the two shuttles performs an invalid action e.g. attempts to pick up a pallet from an empty cell or attempts to move to the left even though the position of the shuttle is already to the far left of the track.

- *mission_complete*: this variable is set equal to 1 when one of the two shuttles decides to take a pallet from the pallet rack and will not encounter the other shuttle during its transport. Once a shuttle is transporting a pallet, it will be the responsibility of the other shuttle to ensure that a collision does not occur.

- *finish_simulation*: variable that allows the brain to understand when all the pallets have been transported to their destination and thus the simulation can end.

2.2 ACTIONS

Actions define how the Brain interacts with the simulated environment based on the actions returned. Since the agents to be controlled are the two shuttles, 2 actions are defined, one for each shuttle. The actions of each shuttle are shown in the table below.

Action	Discrete value
no_action	0
left	1
right	2
pick_pallet	3

The movements of the shuttles are discrete, i.e. they move towards the left or right cell of the pallet rack. When the 'pick_pallet' action is performed, the shuttle picks up the material in front of it and automatically takes it to its destination.

3 BRAIN CONCEPT

A concept indicates what the Brain has to learn and within it the curriculum is defined, which indicates how the training engine should train the AI. The curriculum contains the following elements:

- source, i.e. the simulator that interacts with the AI and provides the observable state information. Within the Brain is defined with the *simulator* statement and describes the interface to the simulator.

- the reward function, which allows each iteration to assign a value to the AI so that the cumulative reward at each episode can be maximised more and more until the optimum is reached.
  The reward function used is shown below.
  reward = -1.5 - state.invalid_action + state.mission_complete * 10 - state.collisions * 10

    - The value -1.5 is used to tell the AI to reach its goal within a simulation as soon as possible so as to be penalised as little as possible.

    - The value of the global state variable *invalid_action* is subtracted from the value of the reward, in order to train the AI not to commit invalid actions.

    - In the event that in an iteration within an episode the state variable *mission_complete* takes a value of 1, then we add a value of 10 as a reward. This value is greater than the previous values in order to give more importance to this sub-objective, to make the AI realise that it has to pick up the pallets from the pallet rack without collisions.

    - When the two shuttles collide, the value of the global state variable *collisions* is set to 1 in order to penalise the AI by a value of 10.

- The terminal function is used to indicate when an episode can end, i.e. when the value of the variable *finish_simulation* is equal to 1.

Finally, some of the training parameters were set as follows:

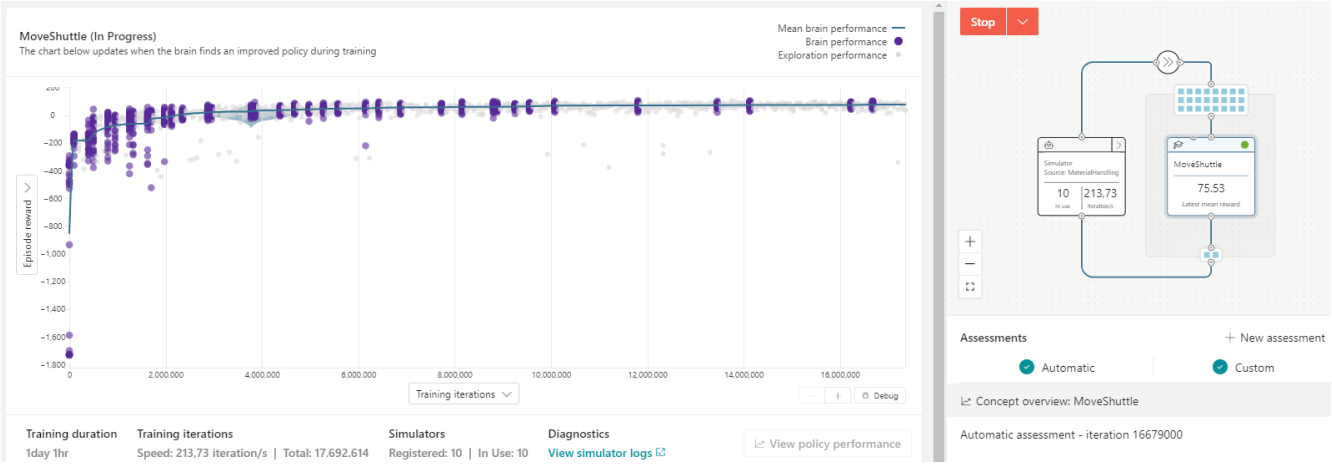
- *EpisodeIterationLimit* = *150*. The Brain must be able to get all the pallets to their destination within 150 iterations. This condition was set in order to avoid too negative a reward if the shuttle does not bring all the pallets to their destination.

- *NoProgressIterationLimit* = 12000000, so as to continue training even in periods when there is no improvement. This is especially useful when the reward comulative is close to the maximum plateau.

- *TotalIterationLimit* = 30000000, so as to stop the training engine when there is likely to be no improvement.

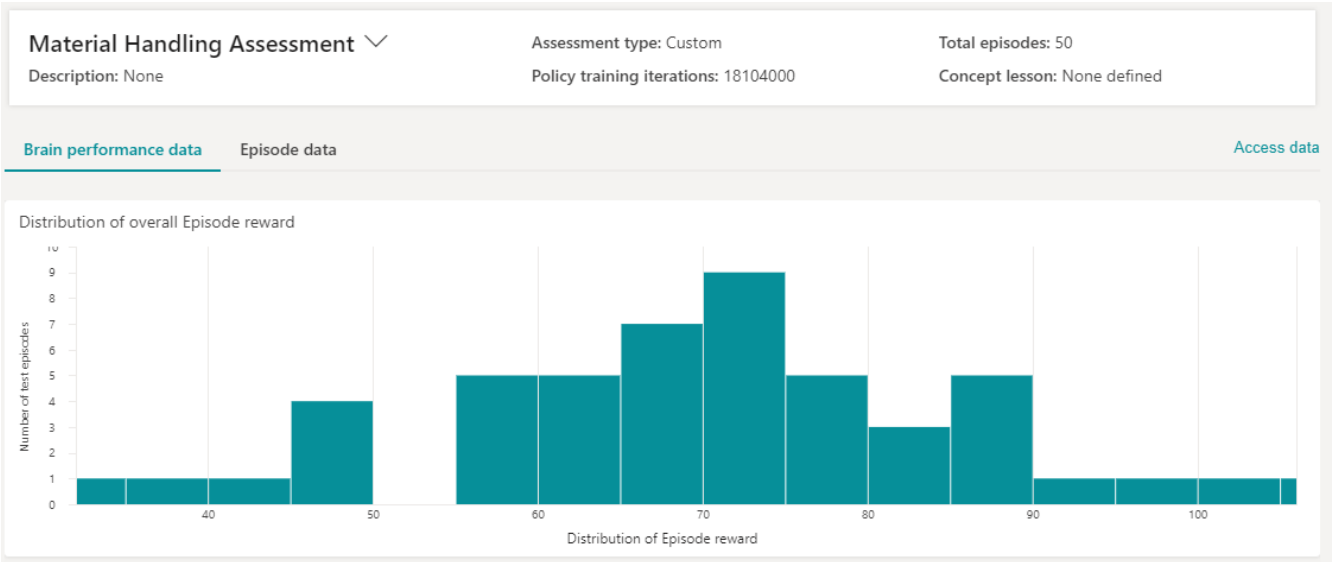
4 BRAIN TRAINING AND ASSESSMENT

After starting the training, you can view the performance plot in the Bonsai data interface as it evolves over time. The performance plot is shown below.



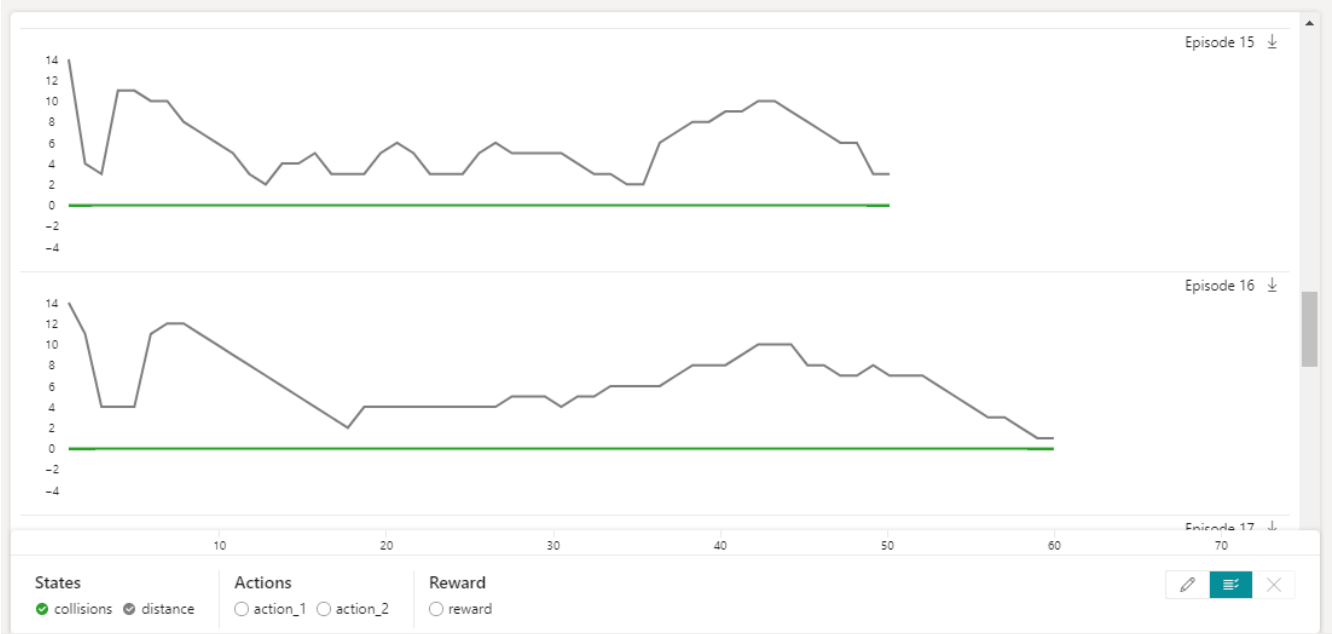
This plot shows the average performance (cumulative reward per episode) of the Brain by launching test episodes (30 by default) to evaluate the Brain's learning.
It can be observed that during the initial phases of the training, the standard deviation of the test episodes cumulative reward is high, while as the training continues it becomes lower and lower, converging towards the optimal average cumulative reward.
After training the Brain for about a day, the cumulative reward will stand at about an average of around 75. At this point we have reached a good level of training and can stop training as there is no meaningful progress.

Once the brain training is finished, you can make a custom assessment by indicating the number of episodes you want to launch. Through the assessment interface, it is possible to view the distribution of the reward, shown in the image below.



In this distribution, we can see that it is similar to a normal distribution and the bin with the most episodes is the one containing the average reward obtained during the training phase on automatic assessments, i.e. about 75. Therefore this corroborates what we saw during the Brain training.

It is also possible to display the value assumed by each state variable during episodes. For example, in the following plot you can see that there are no collisions during the simulation and the distance between the two shuttles respects the safety distance.



In the following plot, on the other hand, we can see that in these two episodes there are no invalid actions and the reward always takes a value of -1.5 adding up to 10 when a shuttle picks up a pallet and takes it to its destination since there are no collisions.