UNIVERSITÀ DEGLI STUDI
DI MILANO

Integrating ML in Unity

A.I. for Video Games

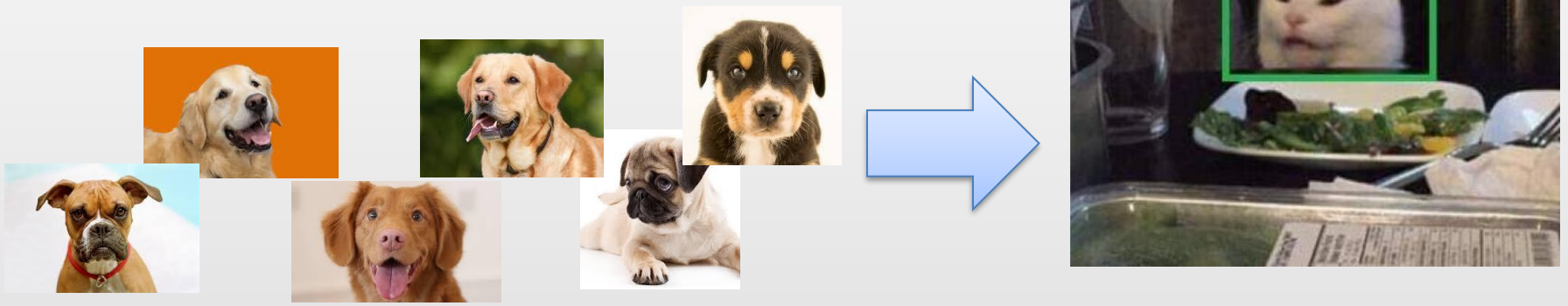
Dario Maggiorini (dario@di.unimi.it)

Machine Learning for (Extreme) Dummies

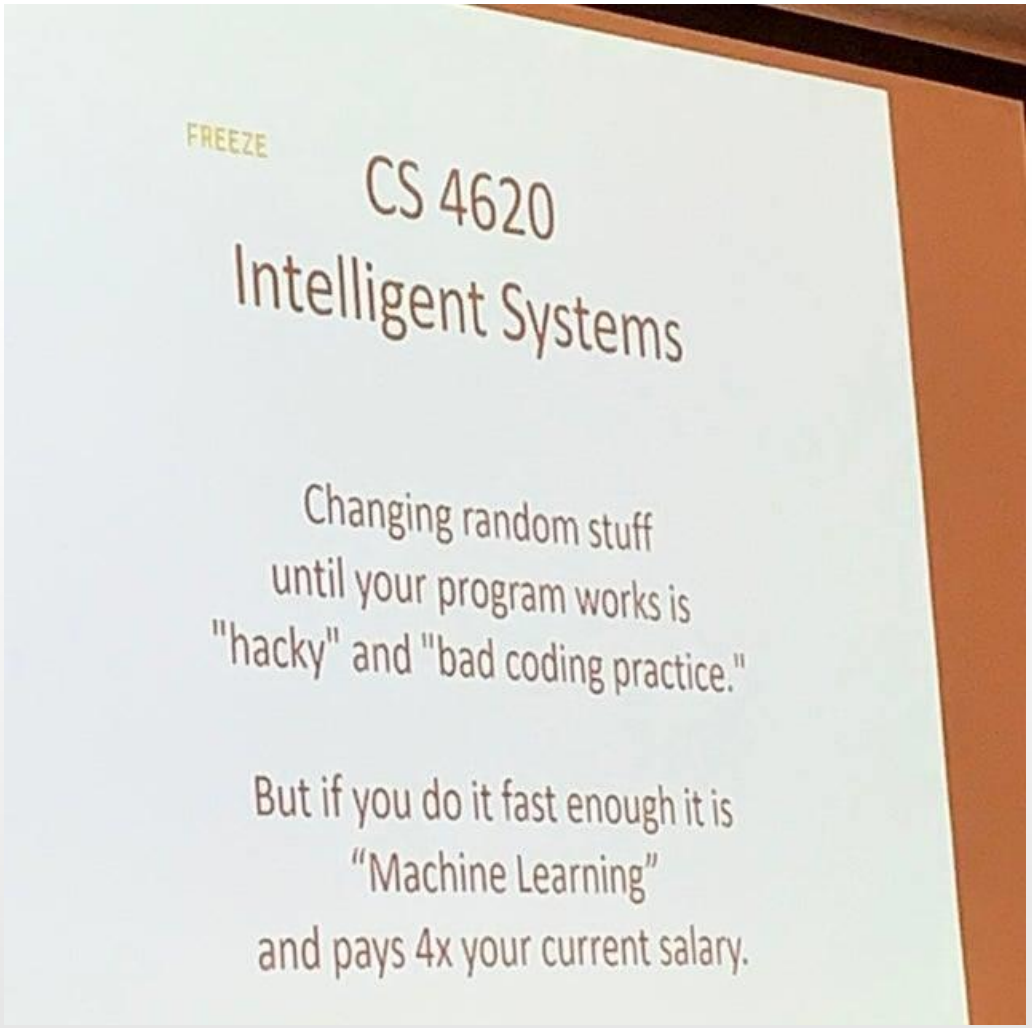
- In (very) sort ...

  Machine learning is the practice of helping software perform a task without explicit programming or rules

- Programmers provide a set of examples and the computer learns patterns from the data



Machine Learning for Practical Scholars



Credit: J. Ben Schafer,
University of Northern Iowa

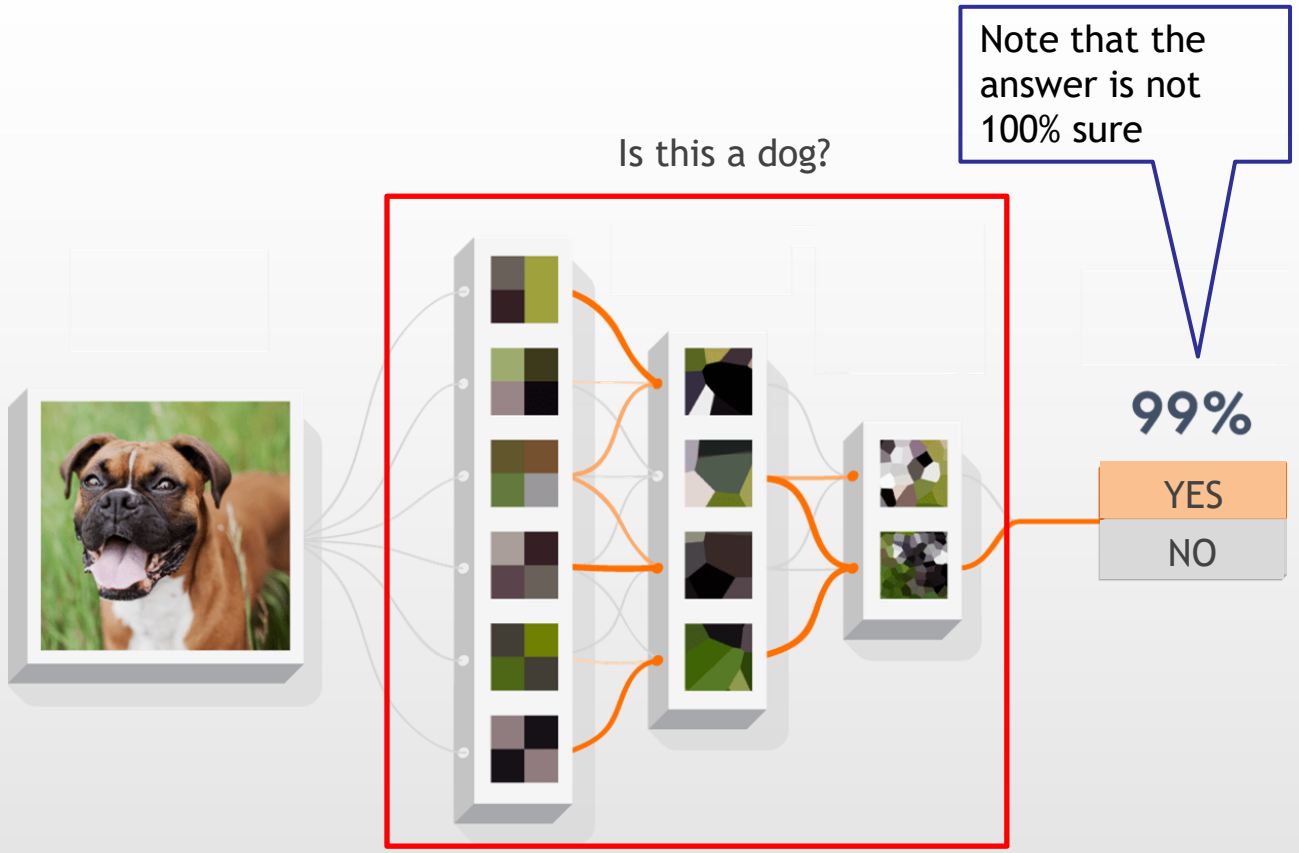
How ML is Providing Results

- An ML system is built over a Neural Network (NN) with many layers
- We put an information in the first layer and a “result” will be coming out from the last one
- The result **is a classification**
  - Given a set of classes, the NN will tell us the probability of the input to belong to each class

ML to Take Decisions

- The input information can be (a subset of) the current knowledge of an agent and the classes could be the action to undertake in each situation

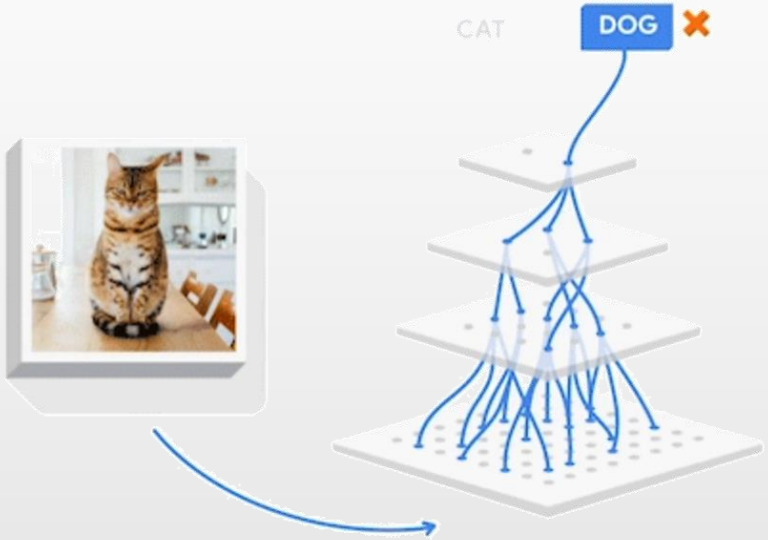
- Each NN is specialized to take one specific decision



Training a Neural Network

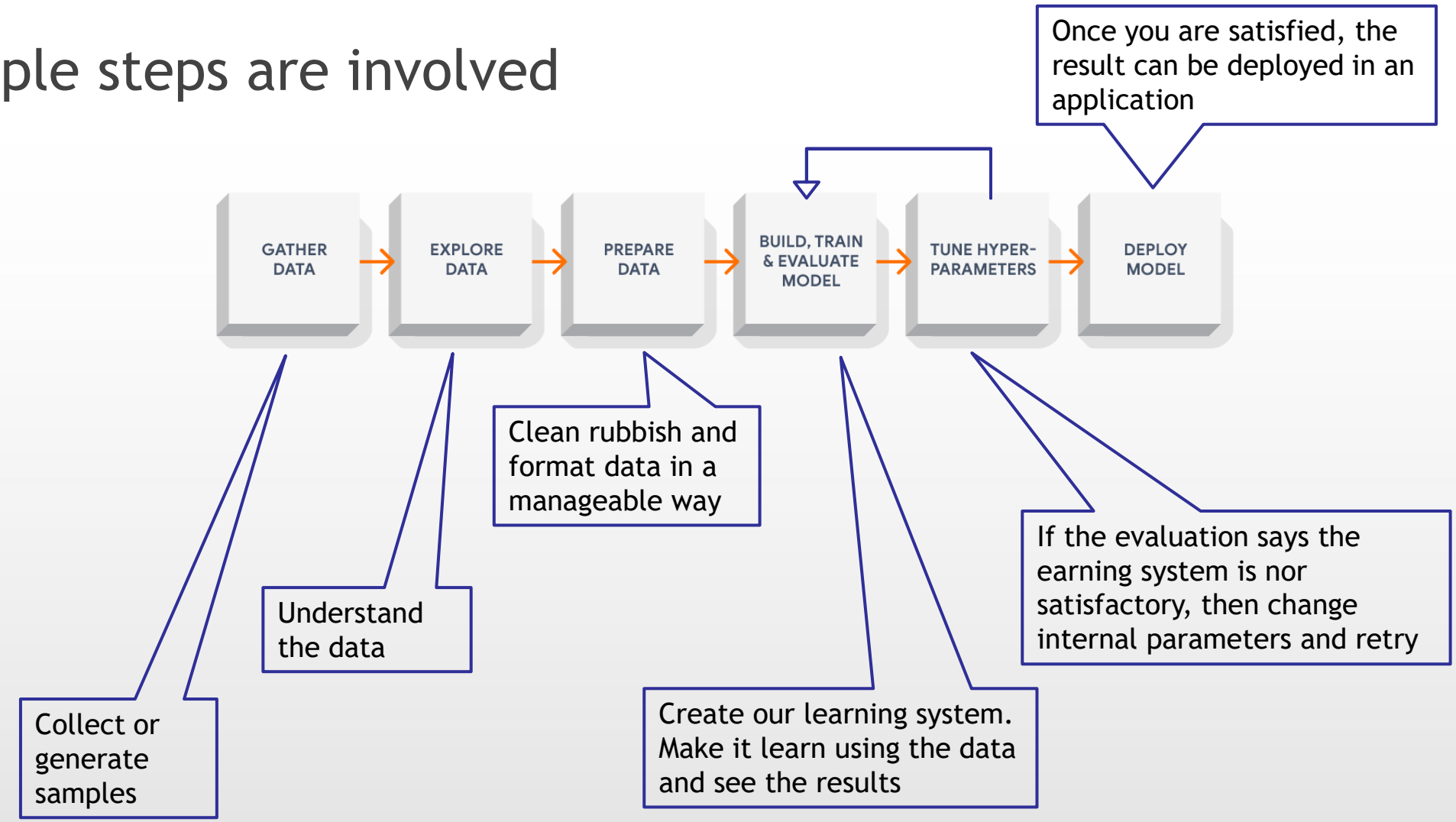
- We submit the examples to the NN and see the results
- By providing some sort of feedback to the system, the NN “adjust” its synapses and gradually converge toward making always the “right” classification

- We have a lot of possibilities here
  - We can let the system understand by itself if the classification is right (unsupervised learning) or we can tell it (supervised learning)
  - We can give rewards and/or penalties based on the quality of the classification (reinforced learning)
  - ... and many other variants



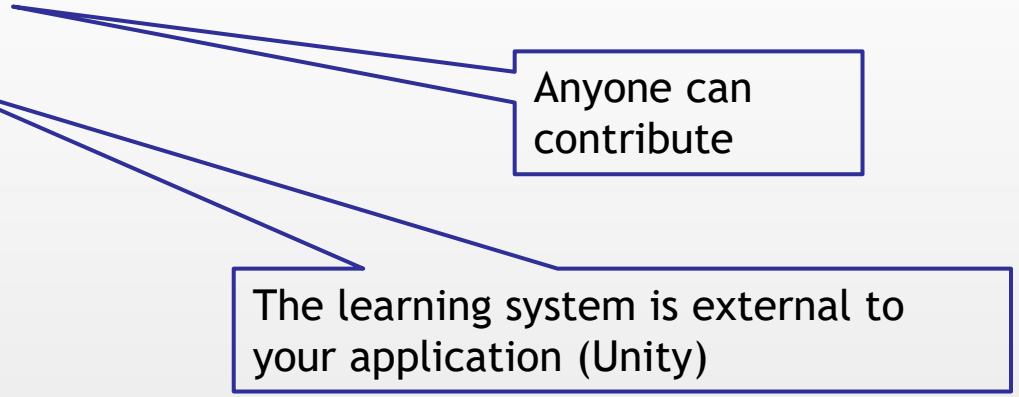
How to Solve a Problem With ML

Multiple steps are involved



What is TensorFlow?

- TensorFlow is a platform for machine learning
  - It is a collection of tools and libraries to let users develop and deploy ML applications
    - Basically, all the software to help you in the steps of the previous slide
  - Open sourced
  - End-to-end



Connecting Unity to TensorFlow

- The connection between Unity and TensorFlow is performed using the ML-Agents package

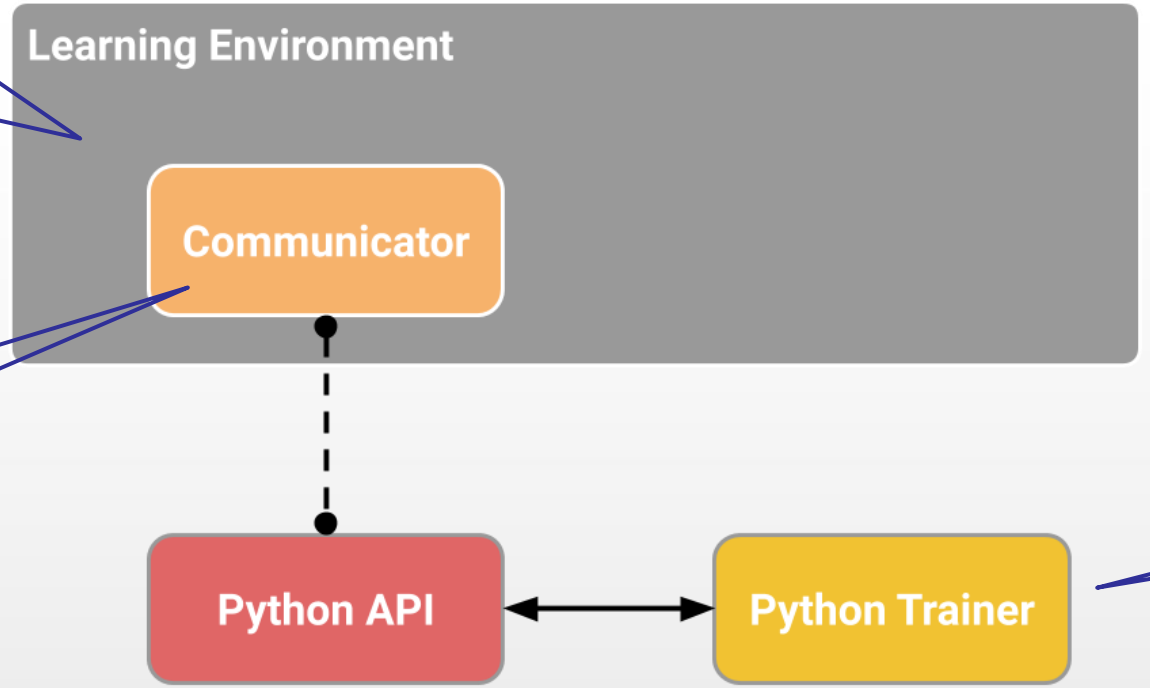
BEWARE of the version!
Once you have one working stick with it!

- ML-Agents will make Unity communicate with TensorFlow through a python API
  - No worries, you do not need to learn python :)

ML-Agents Architecture

This is your scene!
Agents are Gameobjects sensing the environment and performing actions.

They interface to the rest of the architecture via a subclass of the Agent component

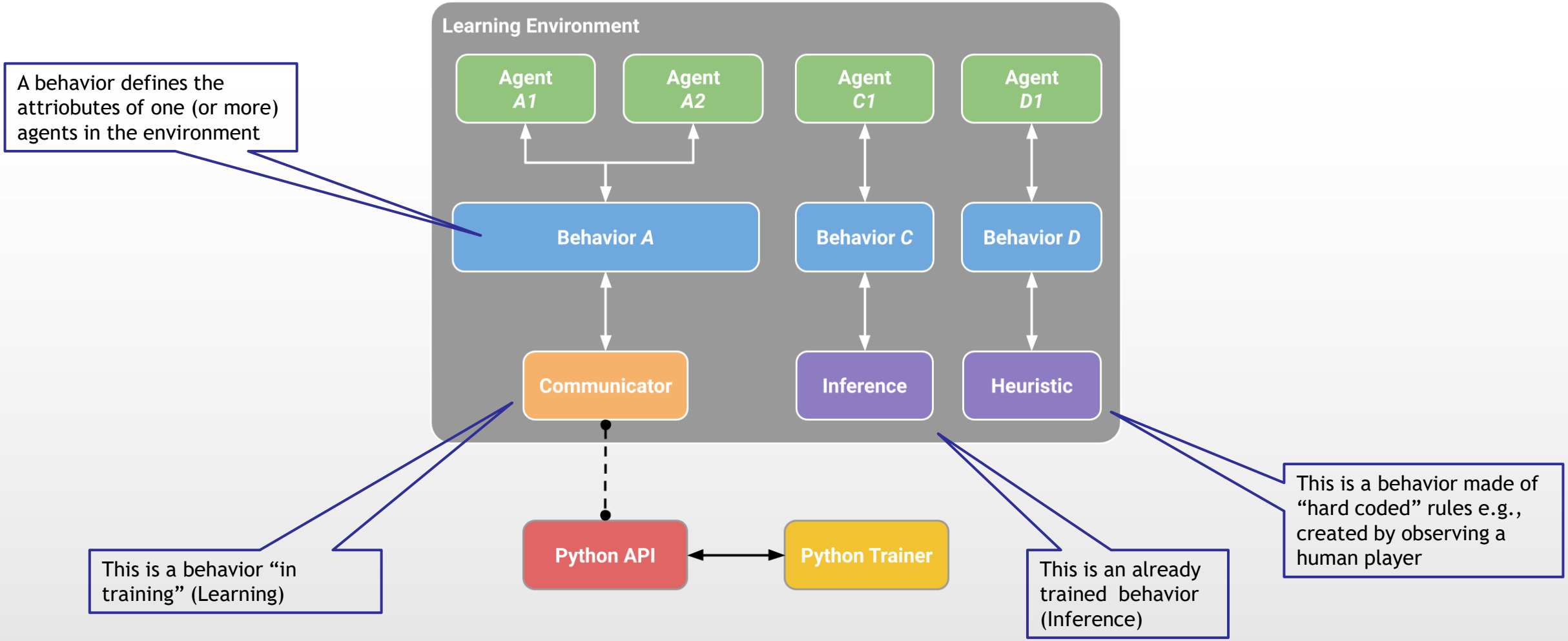


The communicator is living inside the learning environment and is a middleware taking care to communicate with the python library performing the training

Here is where the magic happens. ML algorithms are implemented to train your agent

This is an API library to the ML library

ML-Agents Architecture in Action



Setting Up the Environment

1. Install the python language interpreter
   - Got to http://python.org/ and follow the instructions
   - Select version 3.6.1 or later (3.9.x works fine)

If you have more than one version of python installed, use pip3

2. Install python torch and mlagents
   - We can do this using the python package manager (pip)

Windows users DO THIS FIRST!

Type this on a command line

```
pip install torch
pip install mlagents==0.XX.X
```

VERY IMPORTANT
This depends on the version of your unity package!

This way, we can be sure the version of mlagents and torch will be compatible with each other and with the installed version of python. Moreover, all required libraries are automagically installed

3. Install the ml-agents unity package
   - This can be done via the package manager window
   - Unity version 2018.4 or later is required

And YES! We are done!

Compatibility Matrix

Source: https://github.com/Unity-Technologies/ml-agents

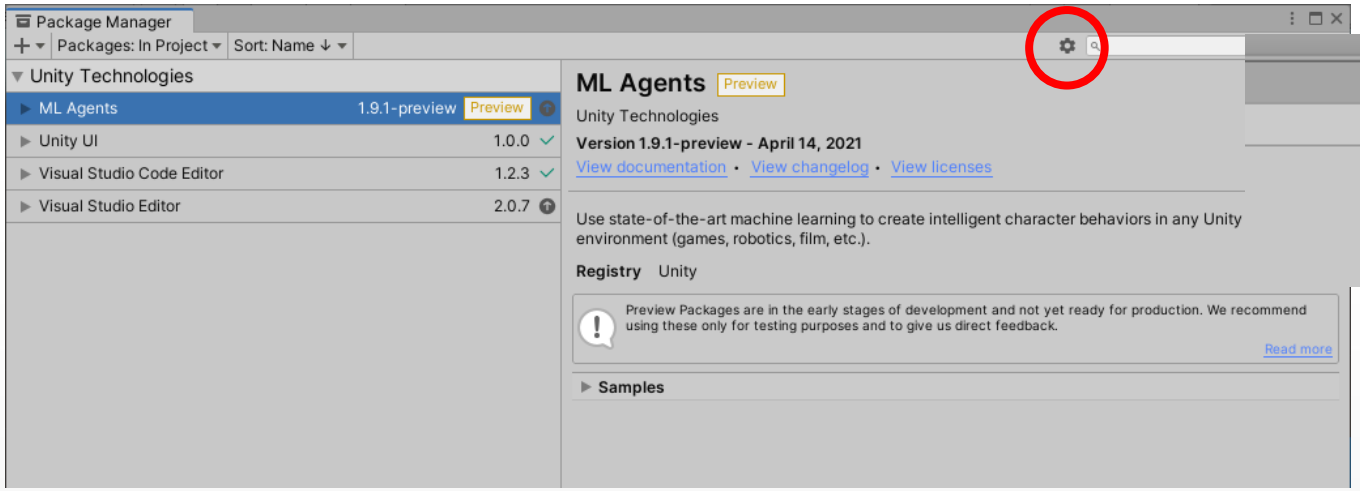
Version	Release Date	Source	Documentation	Download	Python Package	Unity Package
main (unstable)	--	source	docs	download	--	--
Release 18	June 9, 2021	source	docs	download	0.27.0	2.1.0
Verified Package 1.0.8	May 26, 2021	source	docs	download	0.16.1	1.0.8
Release 17	April 22, 2021	source	docs	download	0.26.0	2.0.0
Release 16	April 13, 2021	source	docs	download	0.25.1	1.9.1
Release 15	March 17, 2021	source	docs	download	0.25.0	1.9.0
Verified Package 1.0.7	March 8, 2021	source	docs	download	0.16.1	1.0.7
Release 14	March 5, 2021	source	docs	download	0.24.1	1.8.1
Release 13	February 17, 2021	source	docs	download	0.24.0	1.8.0

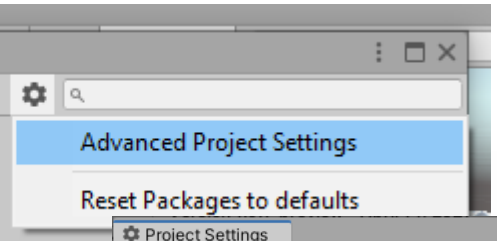
In the GIT you will find this one

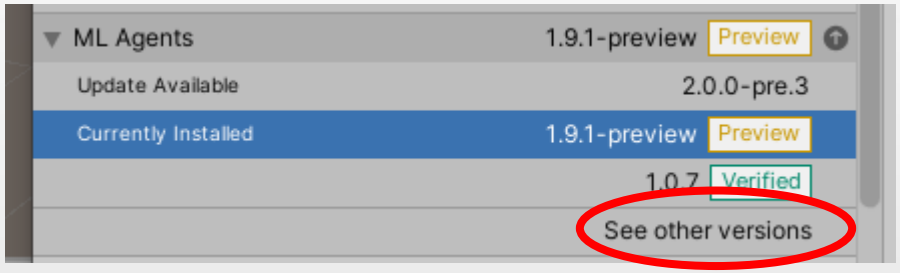
1.0.8 is THE LAST verified version

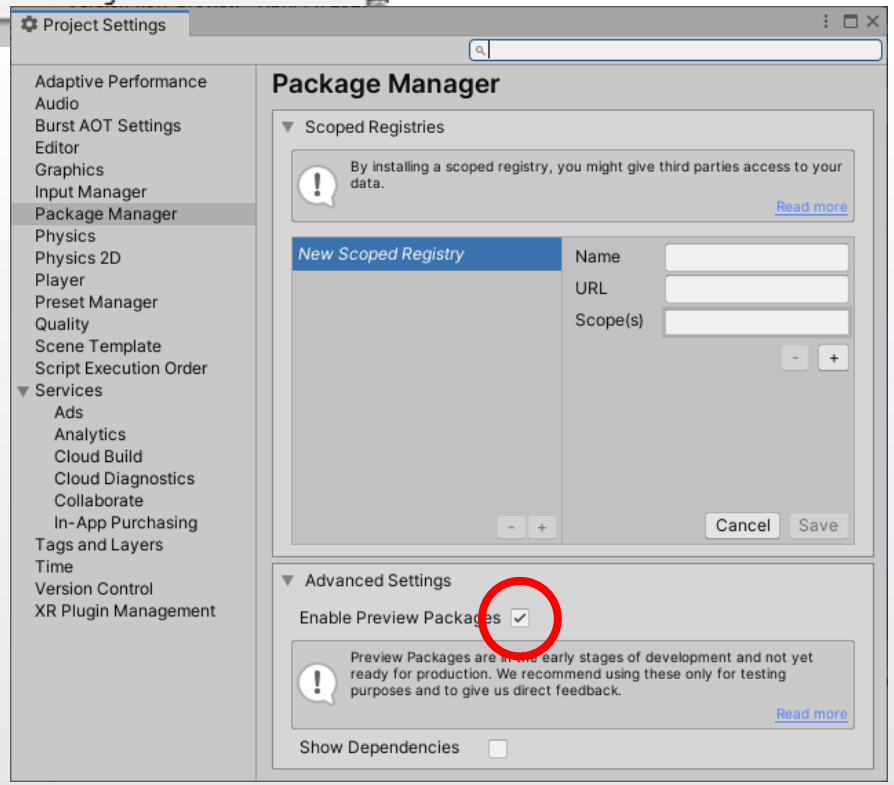
What you should be looking for: the most recent non-beta preview supporting advanced python APIs

Installing Preview Packages in Unity







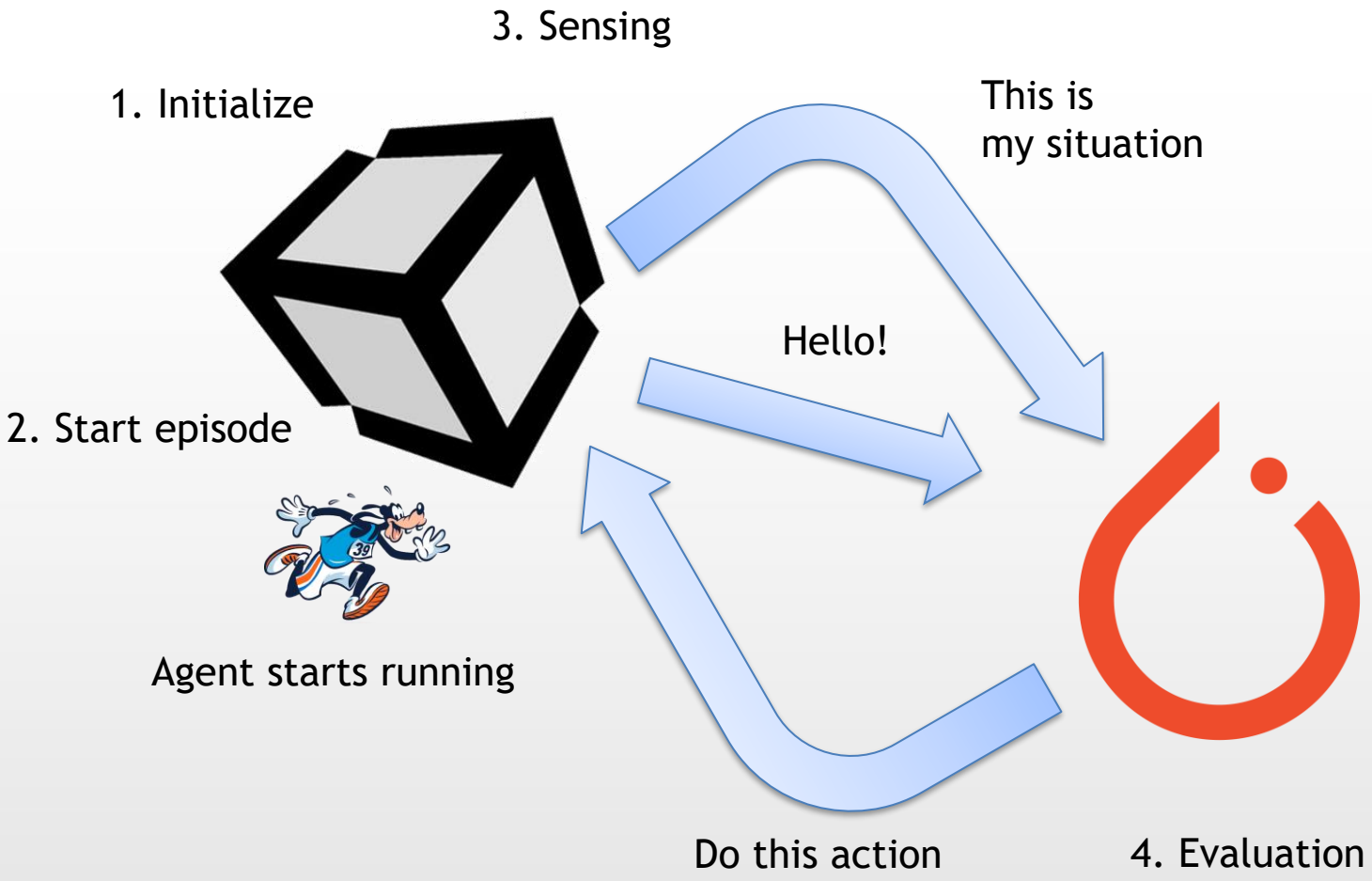


Creating an Agent

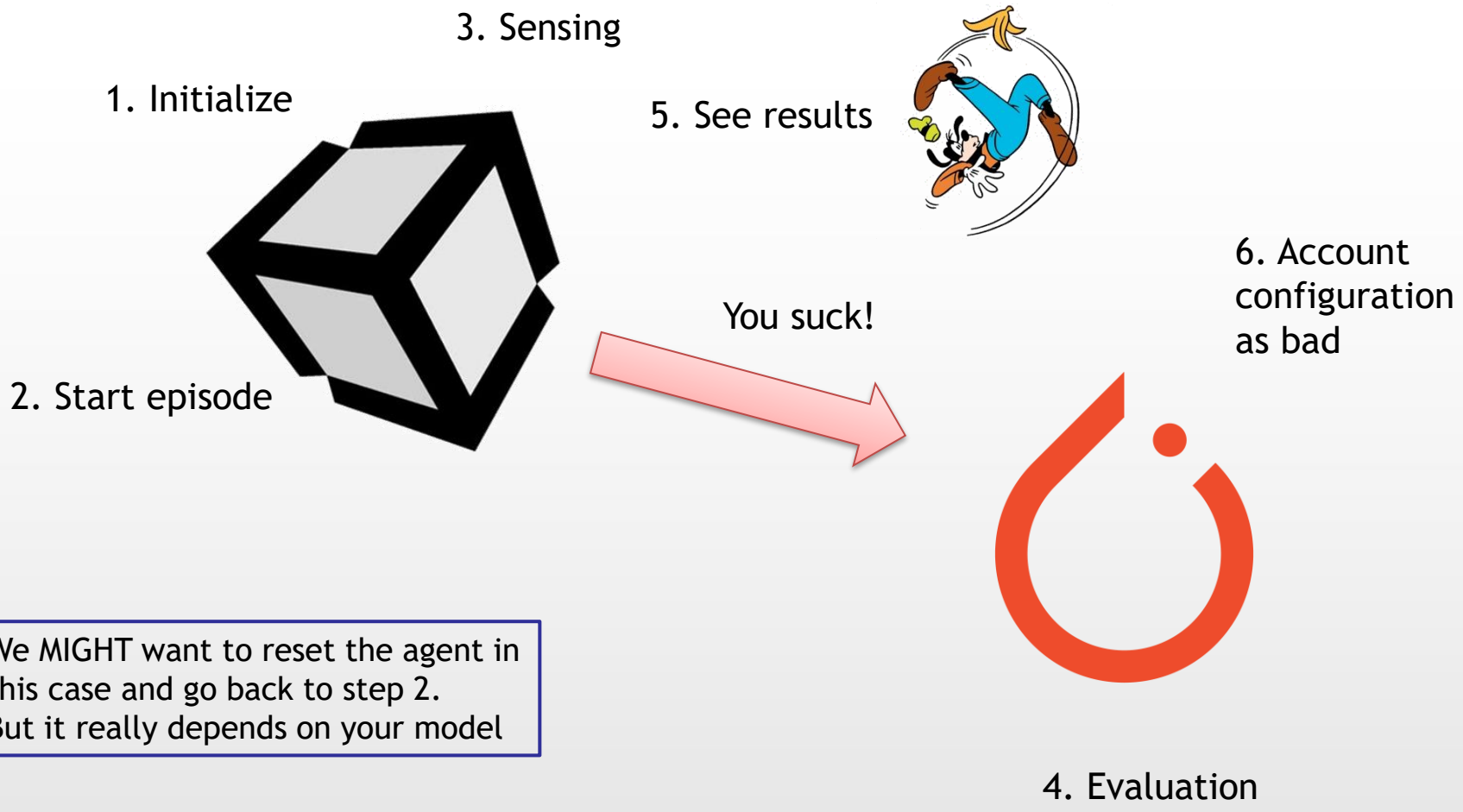
- To create an agent, we must extend the Agent class
  - Doh!

- Inside our class, we must provide methods to:
  - Initialize the agent
  - Start each single run (an episode, whenever the agent is reset)
  - Sense the environment (provide input to the NN)
  - Perform the action requested by the NN and evaluate its outcome

What is Going to Happen?

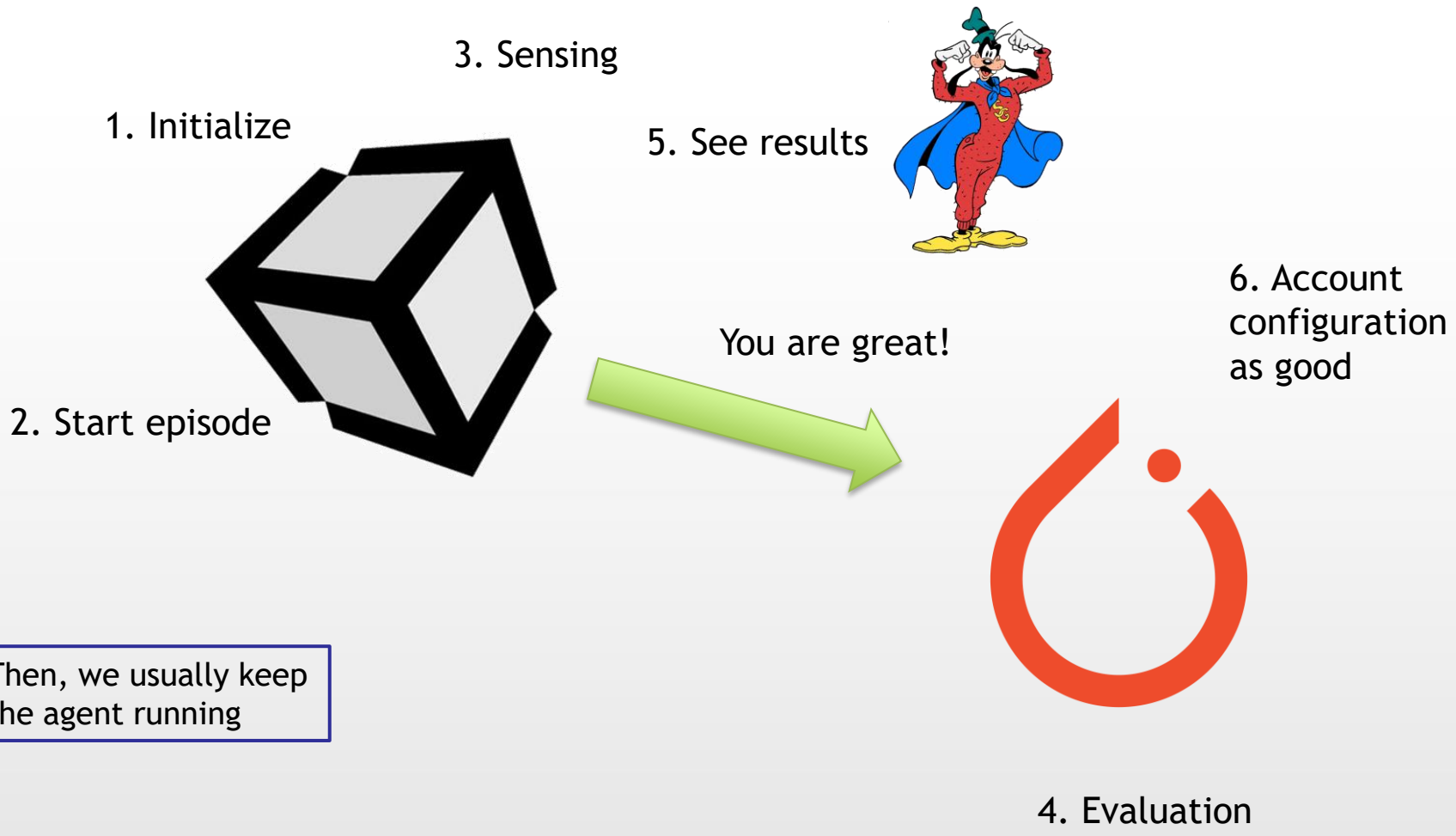


What is Going to Happen? (Case A)



We MIGHT want to reset the agent in this case and go back to step 2.
But it really depends on your model

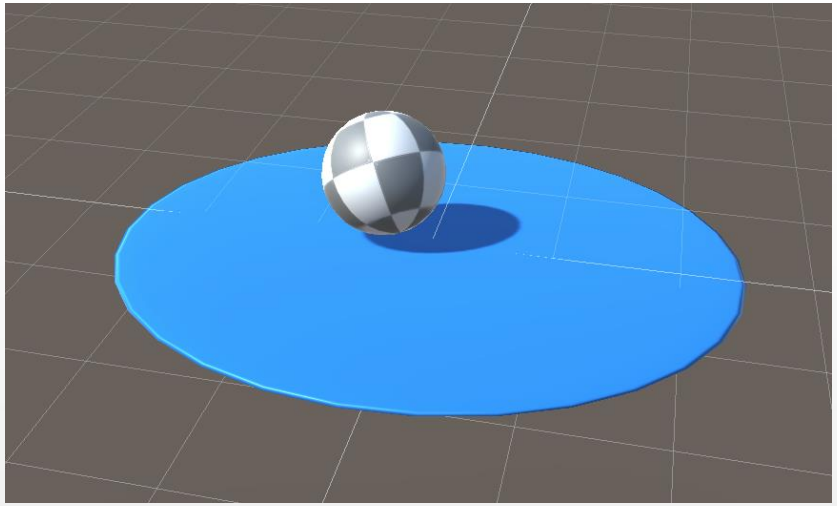
What is Going to Happen? (Case B)



Then, we usually keep the agent running

Let’s Try With a Simple Agent

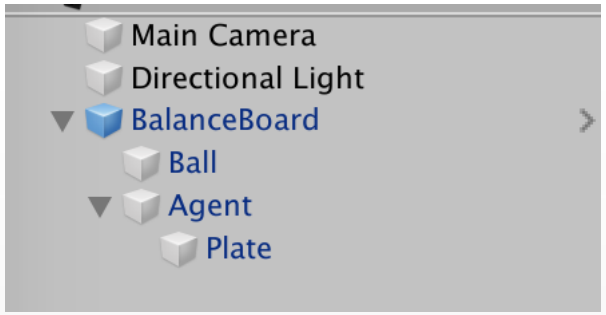
- We want to create an agent to keep a ball balanced on a round platform
- At start, the ball will be placed randomly off-center above the platform and the platform will be rotated randomly around the x and z axis



- The agent must learn how to keep the ball on the platform tilting it

Setting Up the Scene

- In the scene, it is wise to define the agent and all the objects related to it in a hierarchy
  - This will come handy later



- The ball is a sphere with a RigidBody
- The plate is an object with a cylindrical collider

You are NOT required to call this Gameobject “Agent”.
Use whatever name you like

- The Agent Gameobject will host the ML Agent
  - By tilting Agent, we will tilt the platform

Balance Board Agent Header

Source: BBAgent
Folder: ML

Unity ML Library

Unity ML datatypes

This is helpful to distinguish our fields from the superclass fields

```
using UnityEngine;
using Unity.MLAgents;
using Unity.MLAgents.Sensors;

public class BBAgent : Agent {
    [Header("Specific to Balance Board")]
    public GameObject ball;                    // a reference to the ball
    public float maxTilt = 0.25f;              // maximum tilt (0.25 π rad = 45°)

    private Rigidbody ballRbProxy;             // shorthand to the ball Rigidbody
```

NOTE: The version in the repository will be slightly different to support a couple of extensions

Initialize

Source: BBAgent
Folder: ML

This is the same as Start() and Awake(), but it is driven by the learning environment

```
// This method is initializing the system
// will be called once at start
public override void Initialize() {
    ballRbProxy = ball.GetComponent<Rigidbody>();
}
```

Here, all we need to do is to set the proxy for the RigidBody of the ball

Start Episode

Source: BBAgent
Folder: ML

The activity between agent resets is called "an episode"

```
// This method is run when a sequence begins. i.e., we start trying to balance the ball
public override void OnEpisodeBegin () {
    // fist, we set the board to a sligtly random inclination
    gameObject.transform.rotation = new Quaternion (0f, 0f, 0f, 0f);
    gameObject.transform.Rotate (new Vector3 (1, 0, 0), Random.Range (-10f, 10f));
    gameObject.transform.Rotate (new Vector3 (0, 0, 1), Random.Range (-10f, 10f));
    // second, we make sure the ball is still
    ballRbProxy.velocity = new Vector3 (0f, 0f, 0f);
    // third, we place the ball above the plane and off-center by a random value
    ball.transform.position = new Vector3 (Random.Range (-1.5f, 1.5f), 2f, Random.Range (-1.5f, 1.5f))
        + gameObject.transform.position; // the ball is detached from the agent, so we must consider
                                         // the agent position
}
```

In our case, when the agent resets, we must set a random orientation form the plate and a random position above the plate for the ball

Sensing

Source: BBAgent
Folder: ML

Sensing the environment means building a description of the surrounding into a vector.
This vector will be fed to the NN in order to take a decision about “what to do next”

We will see this later

```
// This method is inspecting the world and fill the observation vector
// the number of inserted element MUST be the same as the Space Size parameter
public override void CollectObservations(VectorSensor sensor) {
    sensor.AddObservation(gameObject.transform.rotation.z); // one float
    sensor.AddObservation(gameObject.transform.rotation.x); // one float
    sensor.AddObservation(ball.transform.position - gameObject.transform.position); // three floats
    sensor.AddObservation(ballRbProxy.velocity); // three floats
}
```

ALWAYS REMEMBER!
Semantic is NOT included!

These are just NUMBERS for the learning system. The Neural Network does not know and does not care where they are coming from and what you use them for!

Here we are collecting an environment description based on eight numerical values.

1. The platform rotation around the z axis
2. The platform rotation around the x axis
3. The ball relative position on the x axis
4. The ball relative position on the y axis
5. The ball relative position on the z axis
6. The ball velocity on the x axis
7. The ball velocity on the y axis
8. The ball velocity on the z axis

See Results

Source: BBAgent
Folder: ML

Once again, semantic is NOT included!

The Neural Network is giving you NUMBERS.
It does not care what you are going to do with them, as long as you like the results

```
// This method is called when the Neural Network has some directions about what to do
// The vector used as parameter will have the same size as the Space Size parameter
public override void OnActionReceived(float[] vectorAction) {
    float xAction = 2f * Mathf.Clamp(vectorAction[1], -1f, 1f);
    float zAction = 2f * Mathf.Clamp (vectorAction [0], -1f, 1f);

    float xTilt = gameObject.transform.rotation.x;
    float zTilt = gameObject.transform.rotation.z;

    if ((zTilt < maxTilt && zAction > 0f) || (zTilt > -maxTilt && zAction < 0f)) {
        gameObject.transform.Rotate (Vector3.forward, zAction);
    }

    if ((xTilt < maxTilt && xAction > 0f) || (xTilt > -maxTilt && xAction < 0f)) {
        gameObject.transform.Rotate (Vector3.right, xAction);
    }

    if ((ball.transform.position - gameObject.transform.position).magnitude < 4f) {
        // it the ball is still on the platform, we are happy, give a reward, and go o
        SetReward (0.1f);
    } else {
        // otherwise, we penalize the current configuration
        SetReward (-1f);
        // and we end the run (a new one will be restarted)
        EndEpisode ();
    }
}
```

Output of the NN

Requests to rotate around each axis

Current platform rotation

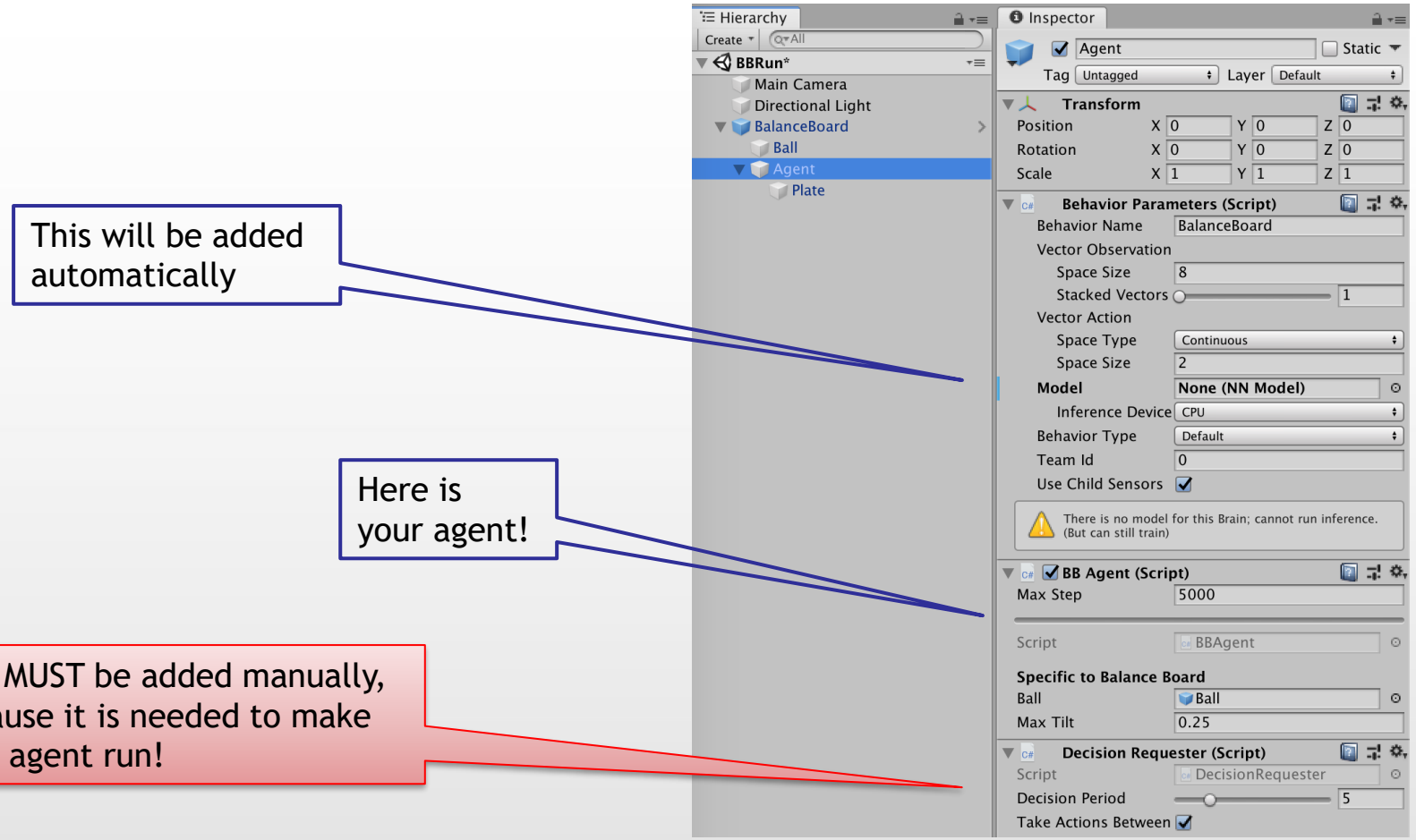
For each axis, if there is a request to rotate and we did not reach the maximum value, then apply a rotation

There is an assumption here about the radius of the platform and the size of the ball

If the ball is still on the platform, we get a reward (the NN was good), otherwise, we get a penalty and start a new episode (we lost the ball)

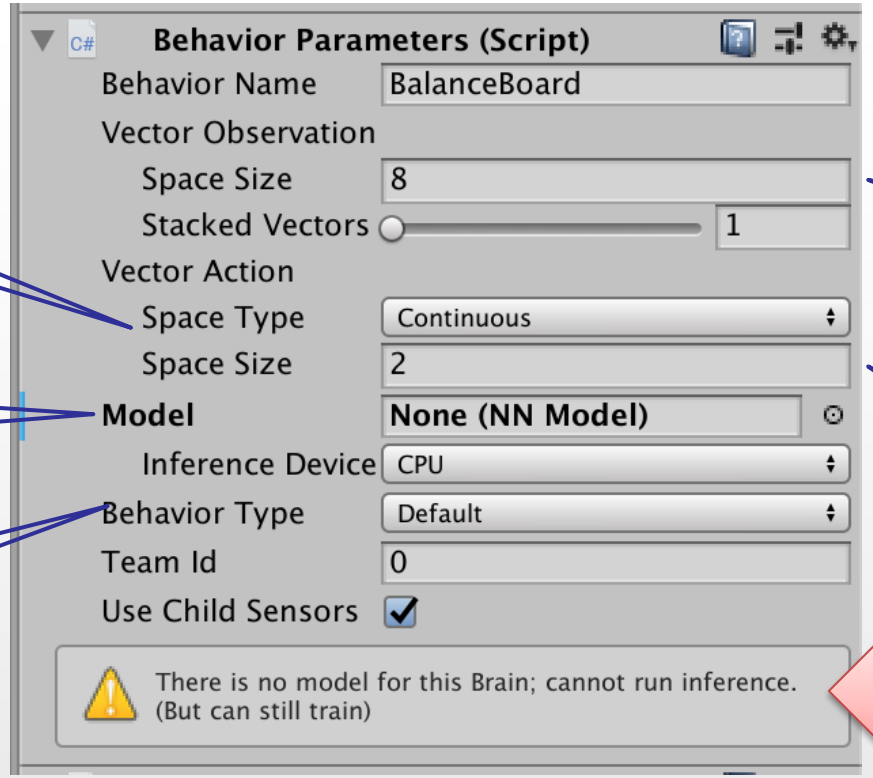
Adding the Agent to the Scene

Add the agent by drag'n'drop as usual (it is a MonoBehaviour!)



Behavior Parameters Component

The Behavior (sic.) Parameter component is managing the NN configuration



Identificator of the behavior for this NN

Type of output: discrete or continue values

Size of the input: the length of the parameter for the CollectObservation() method

At the beginning, we do not have a neural network configured to do inference. We must create one!

How to behave. Train, infer, or learn from a player

Size of the output: the length of the parameter for the OnActionReceived() method

Of course, this panel is going to change with the version of your unity package

Creating Your First Neural Network

- To create a working neural network, we must train the agent

- To do this:
  1. Start TensorFlow using the ml-agents toolkit
  2. Start the scene from inside the editor, and let it go
  3. Go, get a nap

Seriously!
Depending on complexity, training can take hours or even days

Start Training

- We must use the command prompt again

  ```
  mlagents-learn BalanceBoard.yaml --run-id=BBTrain
  ```

  MUST be unique between runs

  Go get my basic yaml file form inside Assets/ML

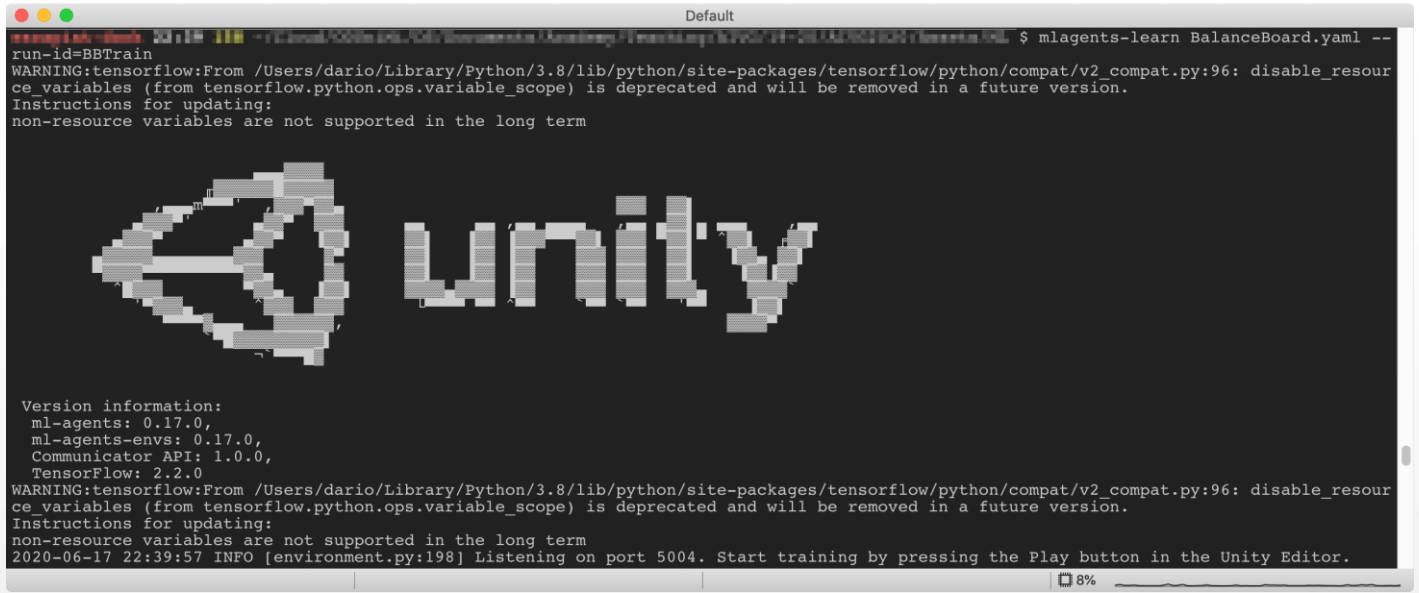
- The yaml file is a textual description of the learning environment.
  It must describe how the agent will be trained
  - The length of the training
  - The hyperparameters of the NN
  - How ml-agents will communicate with torch
  - The policy to use to adjust the weight of the synapses

  A correct configuration of the environment is essential for an optimal training of the agent!
  See https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Training-Configuration-File.md to understand what is inside a yaml file

- The run-id must be a unique identifier for the training
  - A “result” directory will be created, and every training will have its own sub-directory with logs and statistics

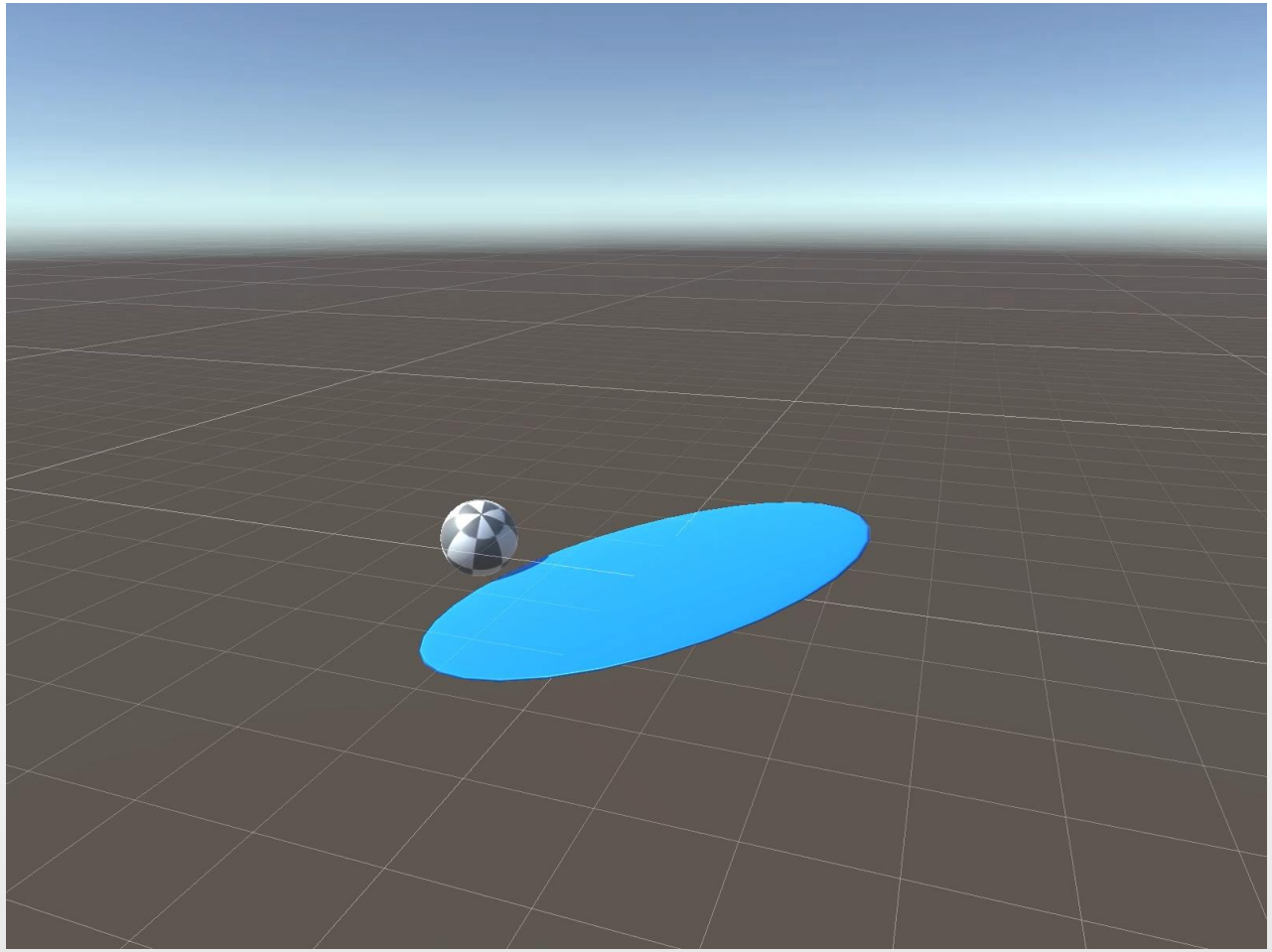
Start Training

- The result is not going to be very exciting



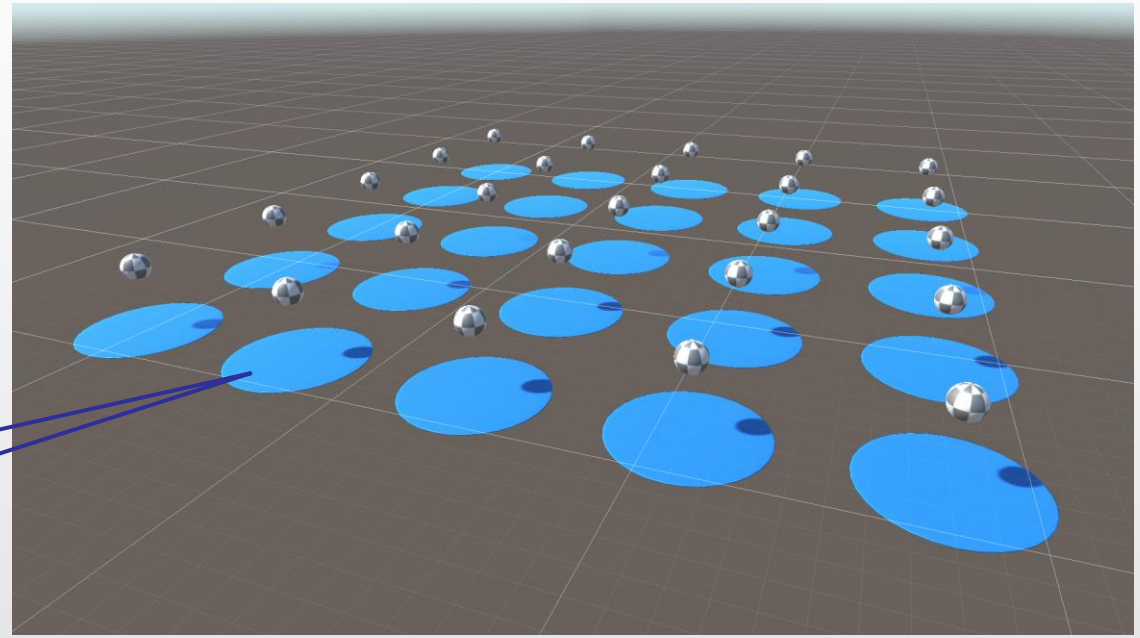
- What you are looking for is the sentence “Start training by pressing the Play button in the Unity Editor”
  - That means that the environment description is correct, and you can start the scene inside the unity editor

Let’s Go Training



Wait a Sec!

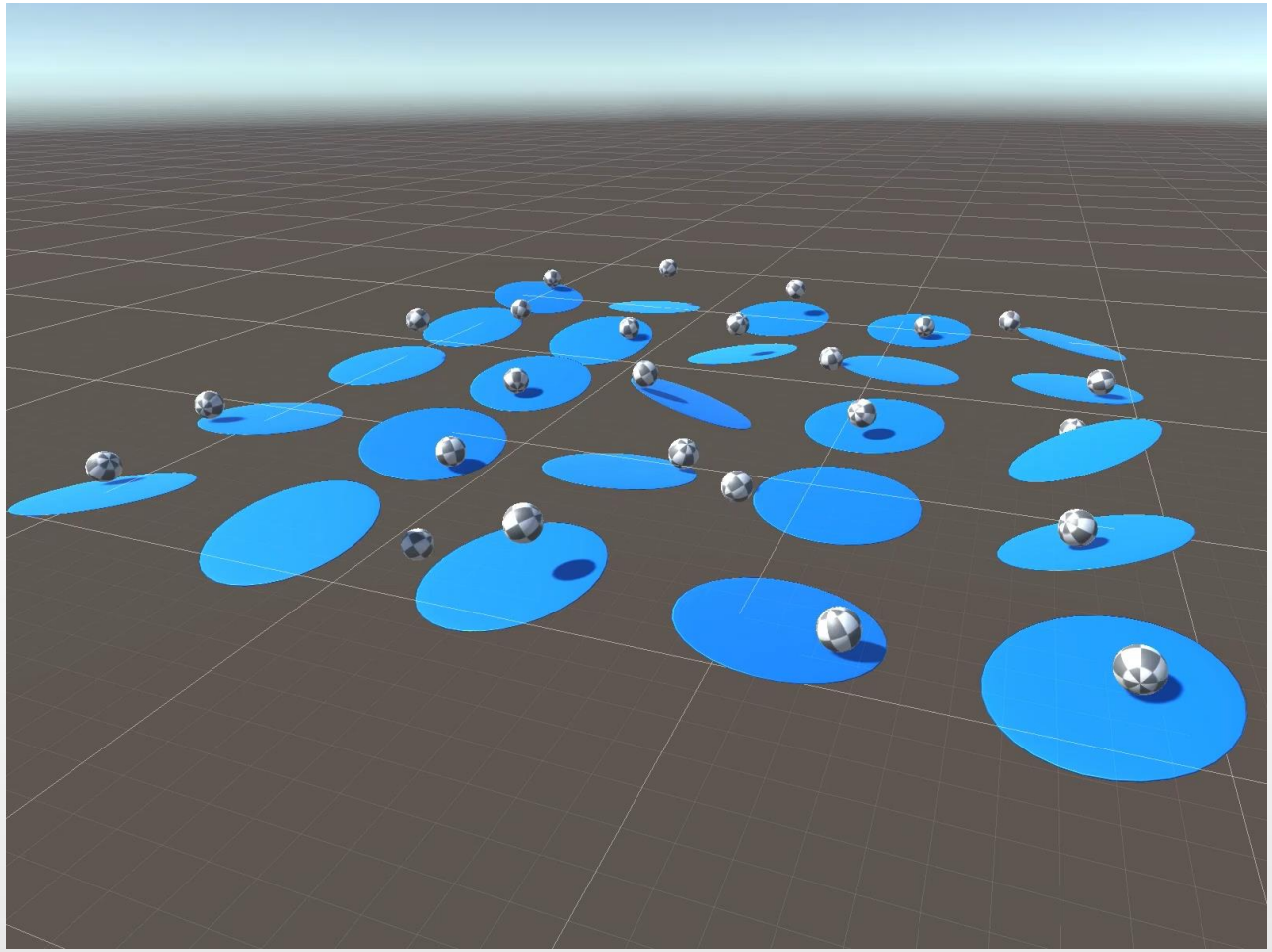
- This is going to take literally FOREVER
- But, on the other hand, we said that many agents can share the same behavior
  - Read ... they can TRAIN the same behavior
- So ... why don't we use many of them?
  - Like in a team training
  - The behavior will learn from all of them in parallel



Having the agent defined as a self-contained prefab is making your life much easier here!

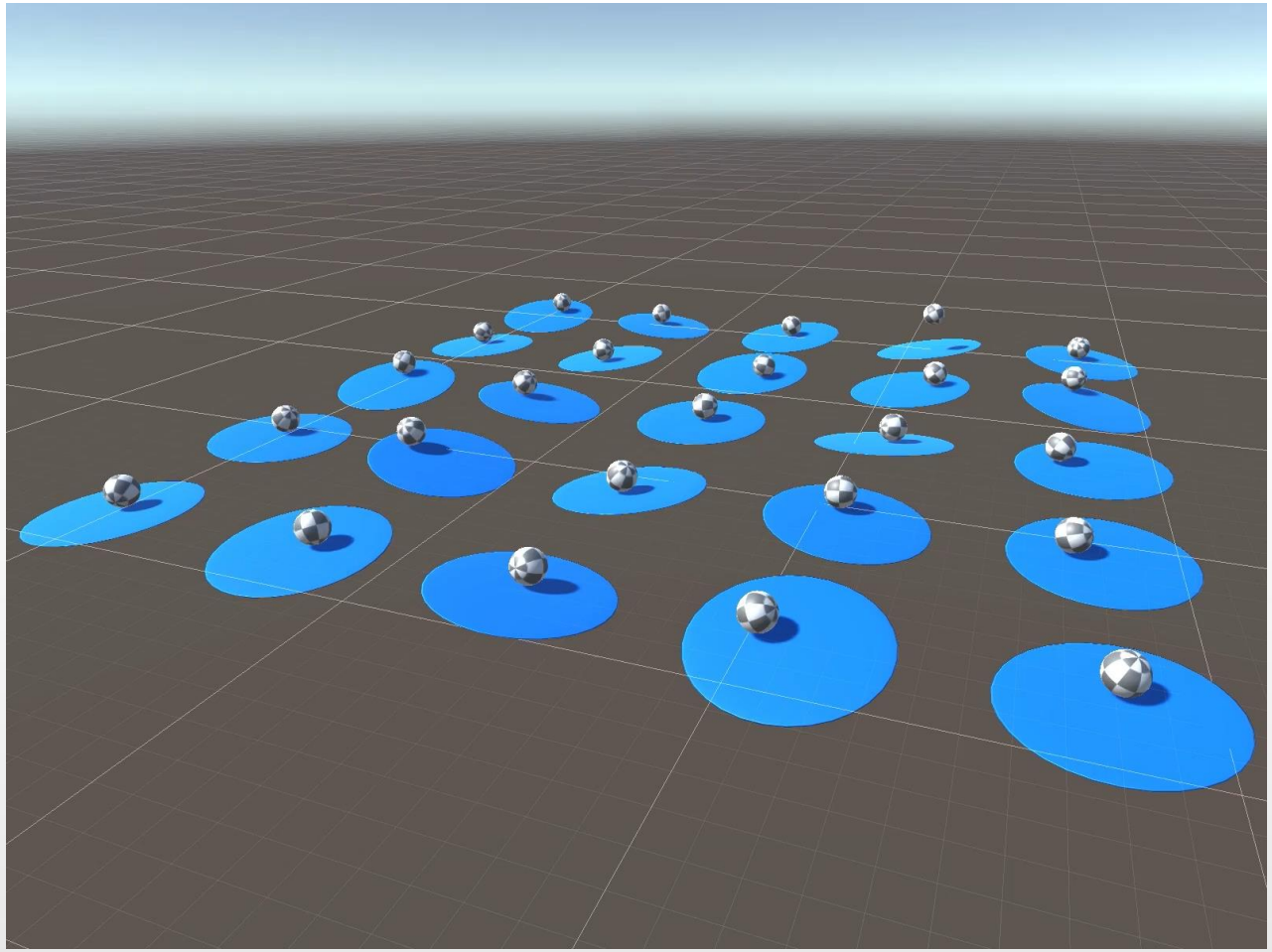
Let’s Go Training ... more

Scene: BBTrain
Folder: ML



And After Just FIVE Minutes ...

Scene: BBTrain
Folder: ML

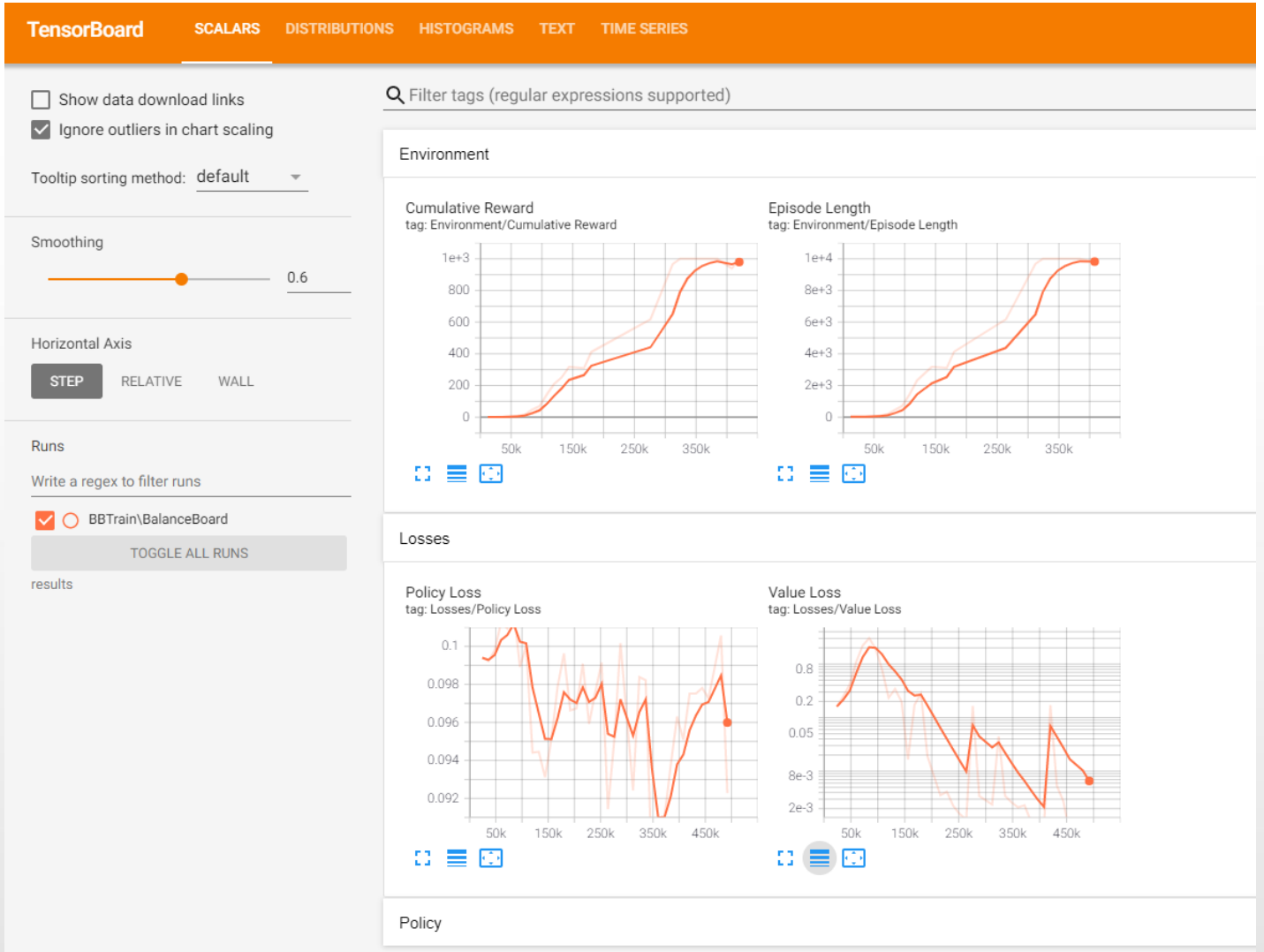


How is my Training So Far?

- This is important to know if your training takes a lot of time
  - Run the following from acommand prompt

    ```
    tensorboard --logdir results
    ```

    Then, point a browser to http://localhost:6006/



After the Training

- The whole training (500000 steps) took me around ten minutes
  - And I did it on a laptop training with CPU. So, you might run much faster

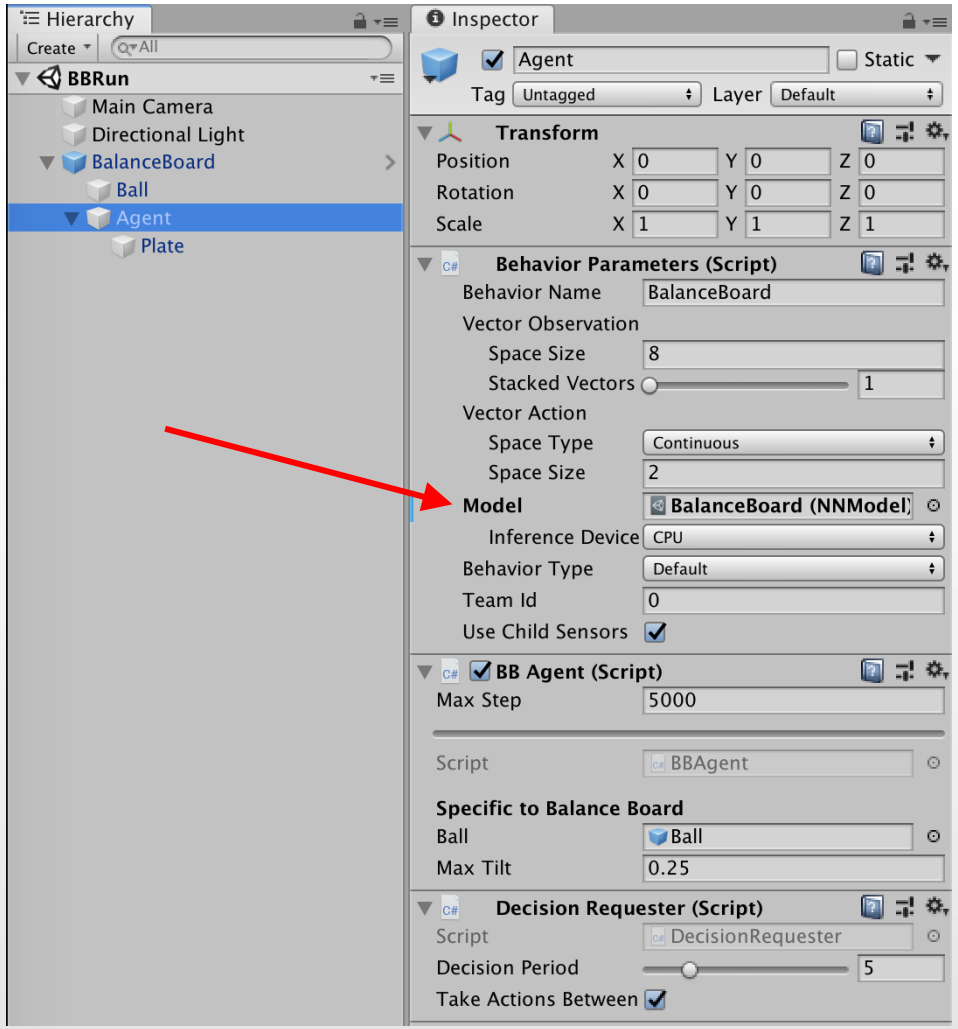
- When the training is over, a dump of the trained NN (your ready-to-use behavior) will be available under the path "results/*id of the training*". The name of the file is the id of the behavior with extension .onnx (or .nn, depending on the mlagents version)

- You can find both my BalanceBoard.yalm and BalanceBoard.onnx in the assets database

Using a Trained Neural Network

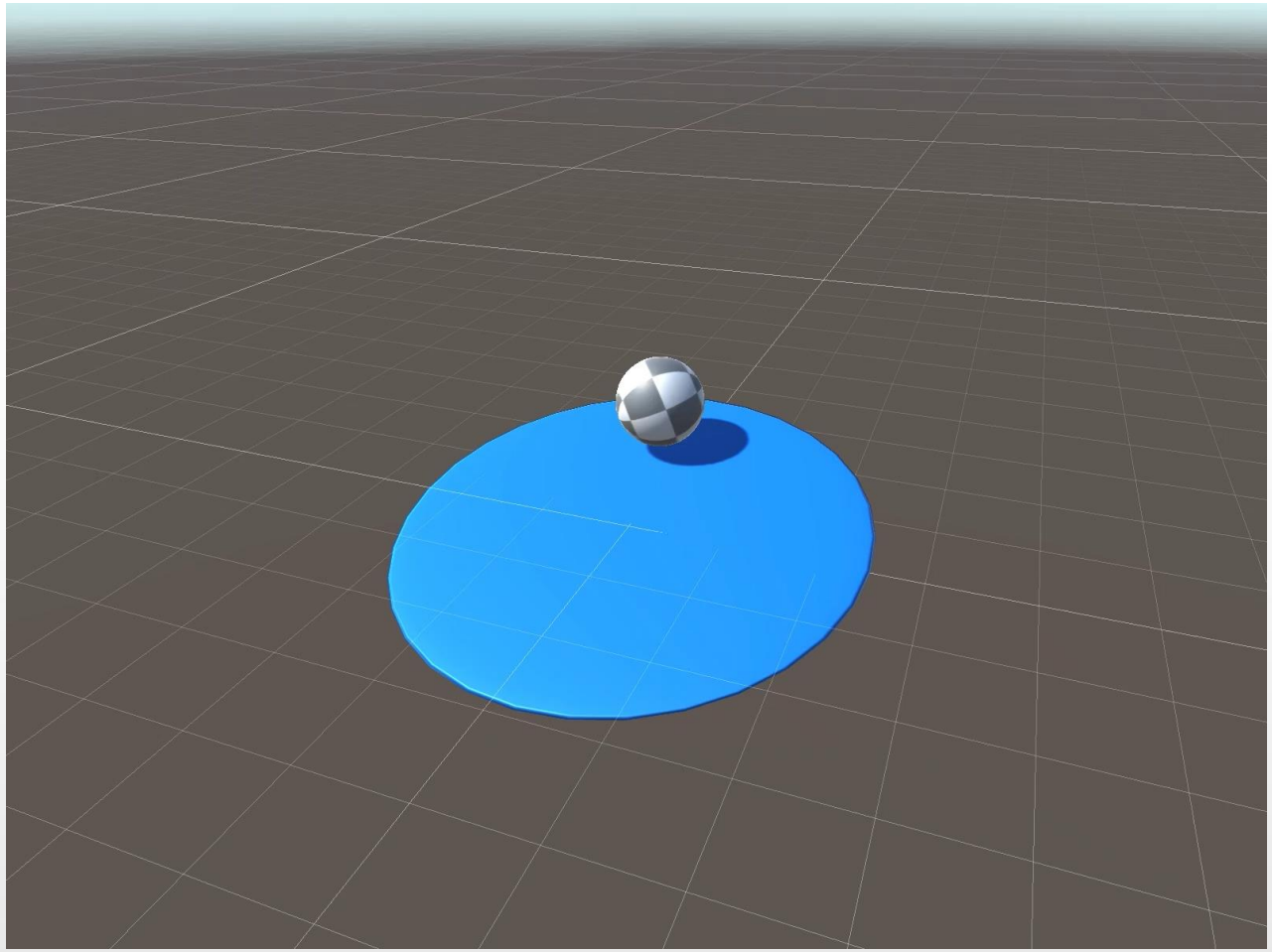
- To add it to the scene, just put the .onnx file in the Model field of the Behavior Parameters component

- Then, you can run the scene again
  - No need for python now!
    The inference will be performed inside Unity



A Trained Neural Network

Scene: BBRun
Folder: ML



Challenging Your Agent

- If you need to stress your agent while training, you can provide it with randomize input from the environment
  - Putting randomness in the environment will ensure all agents will train in the same way

- Using the text on the right, the environment can provide balls of different sizes and mass

- Random values can be extracted using EnvironmentParameters.GetWithDefault

```
parameter_randomization:
  mass:
    sampler_type: uniform
    sampler_parameters:
      min_value: 0.5
      max_value: 10
  scale:
    sampler_type: uniform
    sampler_parameters:
      min_value: 0.75
      max_value: 3
```

Challenging Your Agent

Source: BBAgent
Folder: ML

```
// This method (external to the interface) can be useful when you want the training
// system to generate random parameters.
public void GetTrainerParameters() {
    ballRbProxy.mass = Academy.Instance.EnvironmentParameters.GetWithDefault ("mass", 1.0f);
    float scale = Academy.Instance.EnvironmentParameters.GetWithDefault ("scale", 1.0f);
    ball.transform.localScale = new Vector3 (scale, scale, scale);
}
```

This method can be easily called on initialization and when starting a new episode

Learning From a Human

Source: BBAgent
Folder: ML

- If it is too difficult for the NN to converge, you can use a human to play and Tensorflow/Torchic will learn from that
  - Set Behavior Type to Heuristic, and a Heuristic method will be used inside the agent

```
// This method will be called when Behavior Type is set to Heuristic.
// In this case, the agent will learn by observing the user playing
public override void Heuristic (float [] actionsOut) {
    actionsOut [0] = Input.GetAxis ("Vertical");
    actionsOut [1] = -Input.GetAxis ("Horizontal");
}
```

- In this case, the input from a controller will be used to keep the ball on the platform

Are We Done?

- Of course not!

- What if our trained NN sucks?
  - E.g., giving us the right answer only 50% of the times
- Our only option is to tune the hyperparameters in the environment, retrain ... and hope for the best
  - Especially if training lasts for days

Needing More Stuff?

- Unity ml-agents toolkit documentation
  https://github.com/Unity-Technologies/ml-agents
  In particular, read the “getting started guide”
  - Anyway, I do NOT suggest using the git repository for the ML library instead of pip and the package manager

- TensorFlow resources repository
  https://www.tensorflow.org/resources/
- PyTorch documentation
  https://pytorch.org/docs/stable/

- BEWARE! Online documentation may be inconsistent (or plainly wrong) due to radical changes between ml-agents versions.
  ALWAYS make sure you are reading the right document for your version
  - And NO, do not trust something just because Unity Technologies wrote it