École Centrale Lyon

S9 - Option project
Enonce du BE

Learning behaviors by latent imagination

Students :
Matheus Machado
Samuel Muniz
Tulio Navarro
Filipe Penna Ceravolo Soares

Tutors :
Alexandre Chapin
Liming Chen

27 mars 2023

Table des matières

1 Introduction

Dreamer to control is an artificial intelligence (AI) model is an example of a type of AI model called a "latent imagination model" that is designed to control a simulated agent's behavior based on imagined scenarios.

In the Dreamer model, the agent is trained to learn a representation of the environment that includes both observed data and imagined scenarios. This representation is then used to plan and execute actions that achieve the agent's goals. The imagined scenarios are generated by a separate module called the "imagination module", which generates sequences of latent states that the agent can use to plan its actions.

The Dreamer model is designed to learn and adapt to new environments and tasks, and has been shown to achieve state-of-the-art performance on a variety of challenging control tasks in simulation.

2 Experimental procedure

Some important resources which are going to be exploited in this BE are build considering the x64 architecture, which unfortunately does not contemplate Mac's newer versions, which follow the ARM architecture.

3 Server

Due to the large computational effort required to run the model, it is necessary to use the server hardware of the École Centrale de Lyon, which has a GPU more powerful than that of an ordinary computer. To access the server, open a command prompt and paste the command :

```
ssh projet22@156.18.90.98
```

Then you must enter the password :

```
thaish
```

Then you must confirm the access writing :

```
yes
```

From then on, the server will give you access to a Linux environment in which you will run all the following configuration setups.

4 Windows Subsystem for Linux

In order to be able to install the applications and libraries needed to start the model dreamer, it will be necessary to install the operating subsystem WSL (Windows Subsystems for Linux), or to have a Linux OS in a x64 machine.

If you have a Linux OS, please move to the following section.

The WSL is a feature of Windows 10 and Windows 11 that allows users to run a Linux environment directly on their Windows machine, without the need for a virtual machine

or dual-boot setup. WSL is essentially a compatibility layer that enables Linux binaries to run natively on Windows.

It will be used to run Docker. So if you already have docker in your machine, please move to the following section.

Instruction : Install $WSL2$ and a Linux OS in your machine (to run Docker)

It is possible to check if a version of WSL2 is already installed in your machine using the command

```
wsl --list --verbose
```

If you have a Linux OS already installed in your machine, try to move to the following topic. There are three options :

1. wsl2 is not installed
2. wsl2 is installed, but no Linux OS version is installed
3. wsl2 is installed and already contains a Linux OS version

In newer windows machines, wsl2 is already installed from factory. If it is not installed, you must install it through :

```
wsl --install --web-download
```

After that is necessary to install a Linux version. We recommend the Ubuntu 18.04 (since was the version used to do this tutorial). To do that, run the following command :

```
wsl --install -d Ubuntu-18.04 --web-download
```

On one of the machines we tested, the first command was stuck in the Ubuntu installation, stopping it and running the second command

5 Docker install and setup

Instruction : Install Docker

Please find all necessary indications in the following link. `https://docs.docker.com/get-docker/`. Please note that Docker Desktop will not work properly if WSL2 in not properly configured in windows machines.

6 Docker image and container

To run the codes in this BE, it is necessary to create a local docker container, from an image.

Docker is a platform that enables developers to build, package, and deploy applications in a standardized way, using containers. Containers are lightweight and portable units of software that include everything needed to run an application, such as code, libraries, dependencies, and configuration.

Docker makes it easy to create, distribute, and run applications in containers by providing a set of tools and services that abstract away the underlying infrastructure and operating system. This means that developers can write code once and deploy it anywhere, regardless of the underlying environment.

A Docker image is a read-only template that contains all the files, libraries, and dependencies needed to run an application. It's created using a Dockerfile, which is a script that defines the steps to build an image. An image is like a snapshot of an application and can be used to create multiple containers with the same configuration.

A Docker container is a running instance of an image. It's created from an image using the docker run command, and it's isolated from other containers and the host system. Each container has its own file system, network interface, and process space, which makes it easy to run multiple applications on the same host without conflicts.

Containers are designed to be ephemeral, meaning that they can be started, stopped, and destroyed easily. This makes it easy to scale applications up or down based on demand, and to replace or upgrade containers without affecting other parts of the system.

Overall, Docker provides a powerful and flexible platform for building and deploying applications in containers, which can help developers streamline their workflow and make it easier to build and deploy software.

The first step to run the BE is to create a docker container from an image.

To interact with docker, is necessary that the desktop application must be opened

Instruction : Open docker desktop application

To install the required docker package, just use these commands at your command prompt :

Instruction : Create the docker container

The following commands will start docker and pull the image locally.

```
docker start
docker pull falheisen/dreamer
```

In the case you have a GPU with CUDA installed, run :

```
docker run --gpus all -d --name dreamer falheisen/dreamer
```

Otherwise :

```
docker run -d --name dreamer falheisen/dreamer
```

Instruction : Run the docker container

```
docker exec -it dreamer bash
```

Références

[1] https ://github.com/danijar/dreamer

[2] D. Hafner et al. 2019, Dream to control : learning behaviors by latent imagination `https://arxiv.org/pdf/1912.01603.pdf`

[3] D. Hafner et al. 2019 , Learning latent dynamics for Planning from pixels `https://arxiv.org/pdf/1811.04551.pdf`]

[4] D. Ha et al. 2018, World Models `https://arxiv.org/pdf/1803.10122.pdf`