Homework 5

[starter code]

In this homework, we will train a CNN to do vision-based driving in SuperTuxKart.

This assignment should be solved **individually**. No collaboration, sharing of solutions, or exchange of models is allowed. Please, do not directly copy existing code from anywhere other than your previous solutions, or the previous master solution. We will check assignments for duplicates. See below for more details.

We will design a simple low-level controller that acts as an auto-pilot to drive in SuperTuxKart. We then use this auto-pilot to train a vision based driving system. To get started, first download and install SuperTuxKart on your machine. If you are working on colab, be sure that you have the `GPU` hardware accelerator enabled. To enable the `GPU` hardware accelerator go to `Runtime > Change Runtime Type > GPU > Save` your colab notebook will then restart with GPU enabled.

Once you have GPU enabled use the following to install SuperTuxKart:

```
%pip install -U PySuperTuxKart
```

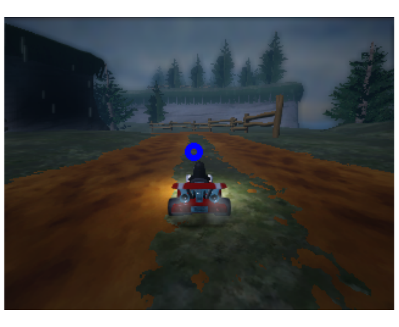
When running the simulator, if you encountered errors about irrlicht devices, make sure you have EGL installed. You can install EGL using `conda install -c anaconda mesa-libegl-cos6-x86_64` if you are using conda.

If installing on a Windows machine it is strongly suggested that you install PySuperTuxKart in a Anaconda environment. Anaconda will handle all of the compilation details that can be quite challenging on a Windows machine. To accomplish this: first create a conda environment for hw5, install the dependencies listed in `environment.yml` (except `PySuperTuxKart` as it is not available through Anaconda), and finally install `PySuperTuxKart` using `pip` as shown above.

If you encounter any issues installing this package, please post them in Piazza.

Controller

In the first part of this homework, you will write a low-level controller in `controller.py`. The controller function takes as input an aim point and the current velocity of the car. The aim point is a point on the center of the track 15 meters away from the kart, as shown below.



We use screen coordinates for the aim point: [-1..1] in both x and y directions. The point (-1,-1) is the top left of the screen and (1, 1) the bottom right.

In the first part of this assignment, we will use a ground truth aim point from the simulator itself. In the second part, we remove this restriction and predict the aim point directly from the image.

The goal of the low-level controller is to steer towards this point. The output of the low-level controller is a `pystk.Action`. You can specify:

- `pystk.Action.steer` the steering angle of the kart normalized to -1 … 1
- `pystk.Action.acceleration` the acceleration of the kart normalized to 0 … 1
- `pystk.Action.brake` boolean indicator for braking
- `pystk.Action.drift` a special action that makes the kart drift, useful for tight turns
- `pystk.Action.nitro` burns nitro for fast acceleration

Implement your controller in the `control` function in `controller.py`. You don't need any deep learning to design this low-level controller. You may use numpy instead of pytorch if you wish.

Once you finish, you could test your controller using

```
python3 -m homework.controller [TRACK_NAME] -v
```

You should tune the hyper-parameters of your controller. You may want to consider gradient-free optimization or exhaustive search. The reference controller completes each level relatively efficiently: `zengarden` and `lighthouse` in under 50 sec, `hacienda` and `snowtuxpeak` in under 60 sec, `cornfield_crossing` and `scotland` in under 70 sec. Note that these times are in-game times and will generally be less than the total computational runtime.

Grade your controller using

```
python3 -m grader homework
```

Hint: Skid if the steering angle is too large.

Hint: Target a constant velocity.

Hint: Steering and relative aim point use different units. Use the aim point and a tuned scaling factor to select the amount of normalized steering.

Hint: Make sure that your controller is able to complete all levels before proceeding to the next part of the homework because you will use your controller to build the training set for your planner.

Planner

In the second part, you will train a planner to predict the aim point. The planner takes as input an image and outputs the aim point in the image coordinate. Your controller then maps those aim points to actions.

Data

Use your low-level controller to collect a training set for the planner.

```
python3 -m homework.utils zengarden lighthouse hacienda snowtuxpeak cornfield_crossing scotland
```

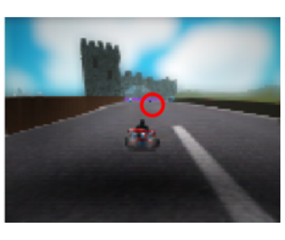
We highly recommend you limit yourself to the above training levels, adding additional training levels may create an unbalanced training set and lead to issues with the final test_grader.

This function creates a dataset of images and corresponding aim points in `drive_data`. You can visualize the data using

```
python3 -m homework.visualize_data drive_data
```

Below are a few examples from the master-solution controller.



Model

Implement your planner model in `Planner` class of `planner.py`. Your planner model is a `torch.nn.Module` that takes as input an image tensor and outputs the aiming point in image coordinates (x: `0..127`, y: `0..95`). We recommend using an encoder-decoder structure to predict a heatmap and extract the peak using a spatial argmax layer in `utils.py`. Complete the training code in `train.py` and train your model using `python3 -m homework.train`.

Vision-Based Driving

Once you completed everything, use

```
python3 -m homework.planner [TRACK_NAME] -v
```

to drive with your CNN planner and controller.

The red circle in the image below is being predicted using the trained master-solution planner network as a substitute for the ground truth aim point used previously.

Grading

We will grade both your controller and planner on the following 6 tracks

- hacienda
- lighthouse
- cornfield_crossing
- scotland
- zengarden
- snowtuxpeak

Your controller/planner should complete each track within a certain amount of time (see grader for details). You receive 5% of your grade by completing each track with your low-level controller. You receive 10% of your grade by completing each track with your image-based agent (i.e., planner/controller pair). You may train on all the above testing track.

For the last 10%, you'll need to complete an unseen test track. We chose a relatively easy test track. You can test your solution against the grader by

```
python3 -m grader homework
```

Extra credit (up to 10pt)

We will run a little tournament with all submissions, the top 9 submissions will receive 10, 9, 8, … extra credit respectively. The tournament uses several unreleased test tracks.

Submission

Once you finished the assignment, create a submission bundle using

```
python3 bundle.py [YOUR UT ID]
```

and submit the zip file online. If you want to double-check that your zip file was properly created, you can grade it again

```
python3 -m grader [YOUR UT ID].zip
```

Grading

The test grader we provide

```
python3 -m grader homework -v
```

will run a subset of test cases we use during the actual testing. The point distributions will be the same, but we will use additional test cases. More importantly, we evaluate your model on the test set. The performance on the test grader may vary. Try not to overfit to the validation set too much.

Submission

Once you finished the assignment, create a submission bundle using

```
python3 bundle.py homework [YOUR UT ID]
```

and submit the zip file on canvas. Please note that the maximum file size our grader accepts is **20MB**. Please keep your model compact. Please double-check that your zip file was properly created, by grading it again

```
python3 -m grader [YOUR UT ID].zip
```

Online grader

We will use an automated grader through canvas to grade all your submissions. There is a soft limit of **5** submisisons per assignment. Please contact the course staff before going over this limit, otherwise your submission might be counted as invalid.

The online grading system will use a slightly modified version of python and the grader:

- Please do not use the `exit` or `sys.exit` command, it will likely lead to a crash in the grader
- Please do not try to access, read, or write files outside the ones specified in the assignment. This again will lead to a crash. File writing is disabled.
- Network access is disabled. Please do not try to communicate with the outside world.
- Forking is not allowed!
- `print` or `sys.stdout.write` statements from your code are ignored and not returned.

Please do not try to break or hack the grader. Doing so will have negative consequences for your standing in this class and the program.

Your latest submission always overrides any previous attempt and counts to wards your final grade (this includes late days).

Running your assignment on google colab

This assignment requires a slightly different setup than the previous colab starters.
IMPORTANT: download this ipynb.

Next, upload the notebook to http://colab.research.google.com then follow the instructions in the notebook.

Honor code

This assignment should be solved **individually**.

What interaction with classmates is allowed?

- Talking about high-level concepts and class material
- Talking about the general structure of the solution (e.g. You should use convolutions and ReLU layers)
- Looking at online solutions, and pytorch samples **without** directly copying or transcribing those solutions (rule of thumb, do not have your coding window and the other solution open at the same time). Always cite your sources in the code (put the full URL)!
- Using any of your submissions to prior homework
- Using the master solution to prior homework
- Using ipython notebooks from class

What interaction is *not* allowed?

- Exchange of code
- Exchange of architecture details
- Exchange of hyperparameters
- Directly (or slightly) modified code from online sources
- Any collaboration
- Putting your solution on a public repo (e.g. github). You will fail the assignment if someone copies your code.

Ways students failed in past years (do **not** do this):

- Student A has a GPU, student B does not. Student B sends his solution to Student A to train 3 days before the assignment is due. Student A promises not to copy it but fails to complete the homework in time. In a last-minute attempt, Student A submits a slightly modified version of Student B's solution. Result: Both students fail the assignment.
- Student A struggles in class. Student B helps Student A and shows him/her his/her solution. Student A promises to not copy the solution but does it anyway. Result: Both students fail the assignment.
- Student A sits behind Student B in class. Student B works on his homework, instead of paying attention. Student A sees Student B's solution and copies it. Result: Both students fail the assignment.
- Student A and B do not read the honor code and submit identical solutions for all homework. Result: Both students fail the class.

Installation and setup

Installing python 3

Go to https://www.python.org/downloads/ to download python 3. Alternatively, you can install a python distribution such as Anaconda. Please select python 3 (not python 2).

Installing the dependencies

Install all dependencies using

```
python3 -m pip install -r requirements.txt
```

Note: On some systems, you might be required to use `pip3` instead of `pip` for python 3.

If you're using conda use

```
conda env create environment.yml
```

The test grader will not have any dependencies installed, other than native python3 libraries and libraries mentioned in `requirements.txt`. This includes packages like `pandas`. If you use additional dependencies ask on piazza first, or risk the test grader failing.

Manual installation of pytorch

Go to https://pytorch.org/get-started/locally/ then select the stable Pytorch build, your OS, package (pip if you installed python 3 directly, conda if you installed Anaconda), python version, cuda version. Run the provided command. Note that cuda is not required, you can select cuda = None if you don't have a GPU or don't want to do GPU training locally. We will provide instruction for doing remote GPU training on Google Colab for free.

Manual installation of the Python Imaging Library (PIL)

The easiest way to install the PIL is through `pip` or `conda`.

```
python3 -m pip install -U Pillow
```

There are a few important considerations when using PIL. First, make sure that your OS uses `libjpeg-turbo` and not the slower `libjpeg` (all modern Ubuntu versions do by default). Second, if you're frustrated with slow image transformations in PIL use `Pillow-SIMD` instead:

```
CC="cc -mavx2" python3 -m pip install -U --force-reinstall Pillow-SIMD
```

The `CC="cc -mavx2"` is only needed if your CPU supports AVX2 instructions. `pip` will most likely complain a bit about missing dependencies. Install them, either through `conda`, or your favorite package manager (`apt`, `brew`, …).