

# User Manual

The instructions described in this chapter are related to the Windows OS. They are pretty straightforward given that the user has already downloaded the necessary project files and meets the dependencies and requirements mentioned in the maintenance manual. One can obtain the program scripts from our [GitHub repository](#).

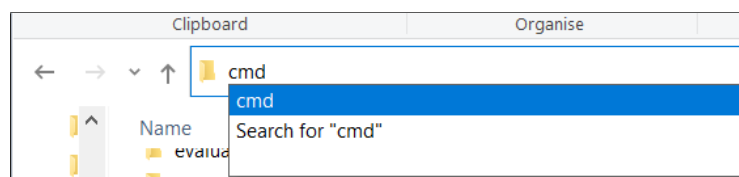
## 1 Data Preprocessing

As mentioned in the paper, the comma.ai dataset is split into 3 types of raw recordings prior to being preprocessed. This process is carried out manually and no script is developed for the purpose. We present the actions required to preprocess these 3 types of recording data. In contrast, the Udacity dataset is preprocessed on the go while training the models via a batcher function thus no additional steps are required.

*Step 1:* Enter a command prompt terminal window;

*Step 2:* Navigate to the directory of the comma.ai dataset model training scripts by using the `cd` command;

Alternatively, one could go to the model training program files directory via Windows File Explorer, use the `Ctrl + L` keyboard shortcut to go to the address bar and enter `cmd`, as shown in Figure 1, to launch a command prompt from the current file location;



**Figure 1:** Entering 'cmd' into the File Explorer address bar.

*Step 3:* Run the data preprocessing script by entering the following command line arguments into the terminal:

```
python preprocess_data.py rec_type [batch_size]
```

The `rec_type` should be set to the value of either 'sunny', 'cloudy' or 'night' since we have 3 types of video files. On the other hand, the optional `batch_size` argument above can be replaced with the desired numerical size value of each batch. It is set to 64 by default.

## 2 Animating the Comma.ai Dataset Files

This process is employed in creating animations from the HDF5 files. These are then played to determine the type of each comma.ai recording file. Consequently, this allows the dataset to be split into subfolders depending on the observed animation features such as sunny or cloudy weather etc.

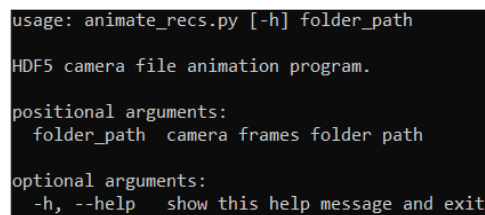
*Step 1:* Enter a command prompt terminal window;

*Step 2:* Navigate to the directory of the Udacity/comma.ai dataset model training scripts by using the `cd` command.

*Step 3:* Run the `train_model.py` script by entering the following command line arguments into the terminal:

```
python animate_recs.py folder_path
```

The `folder_path` argument is set to the path of the original, unprocessed comma.ai dataset camera frames folder. Using the `-h` generates the output portrayed in Figure 2.



```
usage: animate_recs.py [-h] folder_path
HDF5 camera file animation program.
positional arguments:
  folder_path camera frames folder path
optional arguments:
  -h, --help show this help message and exit
```

**Figure 2:** Animation source code file help message.

## 3 Model Training

In the present section we instruct the user about the two neural network training procedures. We analyse the required steps for running the model training scripts of the Udacity self-driving car simulator and comma.ai datasets hereunder.

*Step 1:* Enter a command prompt terminal window;

*Step 2:* Navigate to the directory of the Udacity/comma.ai dataset model training scripts by using the `cd` command.

*Step 3:* Run the `train_model.py` script by entering the following command line arguments into the terminal:

```
python train_model.py model [-unseeded]
```

Using the `-h` optional arguments produces the output depicted in Figure 3. The `model` argument should be replaced by the architecture type the user wants to train. The list of accepted model names corresponding to the CNN, CNN-NCP and CNN-DNCP and its subsequent version architectures are like so: `cnn`, `nncp`, `dncp`, `dncp2`, `dncp3`

```
usage: train_model.py [-h] [-unseeded] model

Model training program.

positional arguments:
  model      model type

optional arguments:
  -h, --help  show this help message and exit
  -unseeded  leave the random script values unseeded
```

**Figure 3:** Help message output of the model training script.

and dncp4. The `-unseeded` argument, which is optional, can also be entered in case the user desires to leave the random script values 'unseeded' before training. By default the models are trained with 'seeded' scripts for reproducibility intentions.

## 4 Evaluation

To evaluate a given model we go through the following steps:

*Step 1:* Enter a command prompt terminal window;

*Step 2:* Navigate to the directory of the Udacity/comma.ai dataset scripts by using the `cd` command.

*Step 3:* Run the `evaluate_model.py` program file by entering the following command line arguments into the terminal:

Udacity dataset:

```
python evaluate_model.py model [-unseeded] [-path]
```

Comma.ai dataset:

```
python evaluate_model.py model gen [-unseeded] [-path]
```

```
usage: evaluate_model.py [-h] [-unseeded] [-path PATH] model

Model evaluation program.

positional arguments:
  model      model type

optional arguments:
  -h, --help  show this help message and exit
  -unseeded  leave the random script values unseeded
  -path PATH enable the user to define a pretrained model path
```

(a) Udacity dataset.

```
usage: evaluate_model.py [-h] [-unseeded] [-path PATH] model gen

Model evaluation program.

positional arguments:
  model      model type
  gen        evaluation data generator type

optional arguments:
  -h, --help  show this help message and exit
  -unseeded  leave the random script values unseeded
  -path PATH enable the user to define a pretrained model path
```

(b) Comma.ai dataset.

**Figure 4:** Command line help messages of the evaluation scripts.

The `model` argument should be replaced by the architecture type the user wants to train. The list of accepted model names corresponding to the CNN, CNN-NCP and CNN-DNCP and its subsequent version architectures are like so: `cnn`, `nnp`, `dncp`, `dncp2`, `dncp3` and `dncp4`. Adding the `-path` optional command line argument enables the user to perform unseeded script evaluation of the models. The `-path` optional argument can be included if a different pretrained model path needs to be specified apart from the ones included in the model evaluation program script (the checkpoints folder). Moreover, the `gen` argument is used to choose between the cloudy

weather and night time evaluation data generators. To evaluate the models on the cloudy weather or night time recordings, the user is required to set the value of the optional argument to either setting the argument value to either cloudy or night respectively.

## 5 Using the Udacity Simulator Environment

The Udacity self-driving car simulator provides an autonomous mode for testing the performance of pretrained models. Sample demo run lakeside and jungle track evaluations are available at our [Youtube channel playlist](#). The steps to achieve an example model evaluation are listed below.

*Step 1:* Launch the Udacity simulator environment application;

*Step 2:* Enter a command prompt terminal window;

*Step 3:* Navigate to the directory where the autonomous car driving script is located;

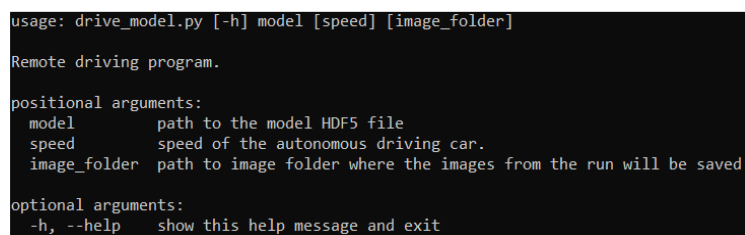
*Step 4:* Run a pretrained model driving simulation by entering the following terminal command:

```
python driving_model.py model [speed] [image_folder]
```

For instance, let's say that the pretrained model, which is located in the 'checkpoints' folder, is called 'cnn.h5'. Furthermore, we want to set the desired car speed to 25 and store the run recording in a file named 'cnn\_run'. Therefore, the command would look like this:

```
python drive_model.py checkpoints\cnn.h5 25 cnn_run
```

A help message output is displayed in Figure 5.



```
usage: drive_model.py [-h] model [speed] [image_folder]

Remote driving program.

positional arguments:
  model      path to the model HDF5 file
  speed      speed of the autonomous driving car.
  image_folder path to image folder where the images from the run will be saved

optional arguments:
  -h, --help  show this help message and exit
```

**Figure 5:** Help message output of the autonomous driving script.