**esources For Completing the Project**

You'll need a few files to complete the Behavioral Cloning Project.

The [**GitHub repository**](https://github.com/udacity/CarND-Behavioral-Cloning-P3) has the following files:

* drive.py: a Python script that you can use to drive the car autonomously, once your deep neural network model is trained
* writeup\_template.md: a writeup template
* video.py: a script that can be used to make a video of the vehicle when it is driving autonomously

The simulator contains two tracks.

We encourage you to drive the vehicle in training mode and collect your own training data, but we have also included sample driving data for the first track, which you can optionally use to train your network. You may need to collect additional data in order to get the vehicle to stay on the road.

Here are links to the resources that you will need:

* [**GitHub Repository**](https://github.com/udacity/CarND-Behavioral-Cloning-P3)
* [**Sample Training Data**](https://d17h27t6h515a5.cloudfront.net/topher/2016/December/584f6edd_data/data.zip)
* [**Project Rubric**](https://review.udacity.com/#!/rubrics/432/view)

**Simulator Download**

* [**Linux**](https://d17h27t6h515a5.cloudfront.net/topher/2017/February/58ae46bb_linux-sim/linux-sim.zip)
* [**macOS**](https://d17h27t6h515a5.cloudfront.net/topher/2017/February/58ae4594_mac-sim.app/mac-sim.app.zip)
* [**Windows**](https://d17h27t6h515a5.cloudfront.net/topher/2017/February/58ae4419_windows-sim/windows-sim.zip)

NOTE \* On Windows 8 there is an issue where drive.py is unable to establish a data connection with the simulator. If you are running Windows 8 It is advised to upgrade to Windows 10, which should be free, and then you should be able to run the project properly.

Here are the newest updates to the simulator:

1. Steering is controlled via position mouse instead of keyboard. This creates better angles for training. Note the angle is based on the mouse distance. To steer hold the left mouse button and move left or right. To reset the angle to 0 simply lift your finger off the left mouse button.
2. You can toggle record by pressing R, previously you had to click the record button (you can still do that).
3. When recording is finished, saves all the captured images to disk at the same time instead of trying to save them while the car is still driving periodically. You can see a save status and play back of the captured data.
4. You can takeover in autonomous mode. While W or S are held down you can control the car the same way you would in training mode. This can be helpful for debugging. As soon as W or S are let go autonomous takes over again.
5. Pressing the spacebar in training mode toggles on and off cruise control (effectively presses W for you).
6. Added a Control screen
7. Track 2 was replaced from a mountain theme to Jungle with free assets , Note the track is challenging
8. You can use brake input in drive.py by issuing negative throttle values

If you are interested here is the source code for the [**simulator repository**](https://github.com/udacity/self-driving-car-sim)

### Center Driving

So that the car drives down the center of the road, it's essential to capture center lane driving. Try driving around the track various times while staying as close to the middle of the track as possible even when making turns.

In the real world, the car would need to stay in a lane rather than driving down the center. But for the purposes of this project, aim for center of the road driving.



Example of Center Lane Driving

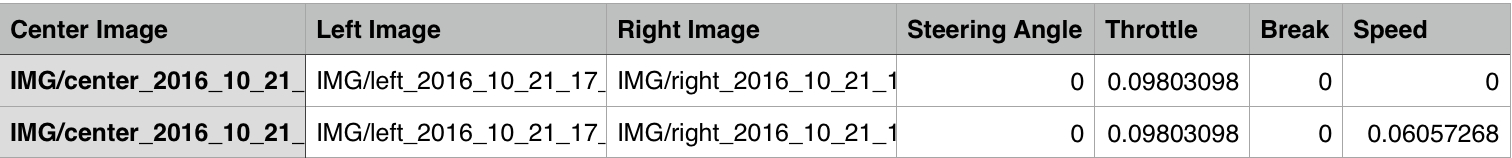
## Strategies for Collecting Data

Now that you have driven the simulator and know how to record data, it's time to think about collecting data that will ensure a successful model. There are a few general concepts to think about that we will later discuss in more detail:

* the car should stay in the center of the road as much as possible
* if the car veers off to the side, it should recover back to center
* driving counter-clockwise can help the model generalize
* flipping the images is a quick way to augment the data
* collecting data from the second track can also help generalize the model
* we want to avoid overfitting or underfitting when training the model
* knowing when to stop collecting more data

If everything went correctly for recording data, you should see the following in the directory you selected:

1. IMG folder - this folder contains all the frames of your driving.
2. driving\_log.csv - each row in this sheet correlates your image with the steering angle, throttle, brake, and speed of your car. You'll mainly be using the steering angle.



## **Project Submission**

For this project, a reviewer will be testing the model that you generated on the first test track (the one to the left in the track selection options).

Whether you decide to zip up your submission or submit a GitHub repo, please follow the naming conventions below to make it easy for reviewers to find the right files:

* model.py - The script used to create and train the model.
* drive.py - The script to drive the car. You can feel free to resubmit the original drive.py or make modifications and submit your modified version.
* model.h5 - The saved model. Here is the [**documentation**](https://keras.io/getting-started/faq/#how-can-i-save-a-keras-model) explaining how to create this file.
* writeup\_report as a markdown or pdf file. It should explain the structure of your network and training approach. The writeup must also include examples of images from the dataset in the discussion of the characteristics of the dataset. While we recommend using English for good practice, writing in any language is acceptable (reviewers will translate). There is no minimum word count so long as there are complete descriptions of the problems and the strategies. See the [**rubric**](https://review.udacity.com/#!/rubrics/432/view) and the [**writeup\_template.md**](https://github.com/udacity/CarND-Behavioral-Cloning-P3/blob/master/writeup_template.md) for more details about the expectations.
* video.mp4 - A video recording of your vehicle driving autonomously at least one lap around the track.

**As a reminder all of the files we provide for this project (**[**rubric**](https://review.udacity.com/#!/rubrics/432/view)**, simulator, github repository, track 1 sample data) can be found in the lecture slide titled "Project Resources".**

#### Further Help

* Use a generator (such as the fit\_generator function provided by Keras). Here is **[some documentation](https://keras.io/models/model/" \t "_blank)** that will help.
* Paul Heraty, a student in the October cohort, has written [**a helpful guide**](https://slack-files.com/T2HQV035L-F50B85JSX-7d8737aeeb) for those of you looking for some hints and advice.
* You can use our sample data for track 1 (see the "Project Resources" lecture for the link)

#### Which track are you evaluated on?

You will only be evaluated on your performance in the first track (the one to the left in the options). The other track is there for your own self-assessment.

**Tips:**

1. How to use Python generators in Keras. This was critical as I was running out of memory on my laptop just trying to read in all the image data. Using generators allows me to only read in what I need at any point in time. Very useful.

2. Use a GPU. This should almost be a prerequisite. It is too frustrating waiting for hours for results on CPU. I must have run training 100 times over the past 3 weeks and it was driving me crazy. Using a GTX980M was around 20x faster in training that a quad­core Haswell CPU.

3. Use an analog joystick. This also should be a prerequisite. I'm not sure if its even possible to train with keyboard input. I think some have managed it, bu for me it's a case of garbage in, garbage out.

4. Use successive refinement of a 'good' model. This really saves time and ensures that you converge on a solution faster. So when you get a model working a little bit, say passing the first corner, then use that model as a starting point for your next training session (kinda like Transfer Learning). Generate some new IMG data, lower the learning rate, and 'fine tune' this model.

5. Use the 50Hz simulator. This generates much smoother driving angle data. Should be the default. You can find a link to download this on the Slack channel. Choose the fastest graphic quality and lowest screen resolution has helped the model to perform better

6. You need at least 40k samples to get a useful initial model. Anything less was not producing anything good for me.

7. Copy the Nvidia pipeline. It works :) And it's not too complex.

8. Re­size the input image. I was able to size the image down by 2, reducing the number of pixels by 4. This really helped speed up model training and did not seem to impact the accuracy.

9. I made use of the left and right camera angles also, where I modified the steering angles slightly in these cases. This helped up the number of test cases, and these help cases where the car is off center and teaches it to steer back to the middle.

10. Around 5 epochs seems to be enough training. Any more does not reduce the mse much if at all.

11. When you're starting out, pick three images from the .csv file, one with negative steering, one with straight, and one with right steering. Train a model with just those three images and see if you can get it to predict them correctly. This will tell you that your model is good and your turn­around time will be very quick. Then you can start adding more training data

. drive.py sends RGB images to the model; cv2.imread() reads images in BGR format!!!!

every time i fit neural networks, I test the following.

1­ Start with very little data, and over fit. If you cant overfit, your implementation is wrong. This is typically not done, but is actually a very powerful tool for checking your implementation is correct. We know if we have too little data, we will overfit. So its good to actually see if your network does it.