**Quality of Code**

**The model provided can be used to successfully operate the simulation.**

Amazing job building a model to operate the simulation.

**The code in model.py uses a Python generator, if needed, to generate data for training rather than storing the training data in memory. The model.py code is clearly organized and comments are included where needed.**

Great work here using a [Python generator](https://jeffknupp.com/blog/2013/04/07/improve-your-python-yield-and-generators-explained/) and you did a well job to comment the code. 

**Suggestions and Comments:**

* For more details, about quality of code in python, please refer to [Google Python Style Guide](https://google.github.io/styleguide/pyguide.html) to see how to improve the code quality in order to know more about coding industrial standards.
* Data augmentation will help to extract as much information from data as possible. You can refer to [this document](https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html) to see how to build powerful image classification models using very little data.
* Awsome job.

**Model Architecture and Training Strategy**

**The neural network uses convolution layers with appropriate filter sizes. Layers exist to introduce nonlinearity into the model. The data is normalized in the model.**

Fantastic work here your neural network uses 5 convolution layers, 3 fully connected layers and you normalized the data using [lambda layer](https://keras.io/layers/core/#lambda).

**Suggestions and Comments:**

* Great implementation of the [Nvidia Model](https://devblogs.nvidia.com/parallelforall/deep-learning-self-driving-cars/" \t "_blank) with some adjustment.
* [Normalization Layers](https://keras.io/layers/normalization/) is another way to normalize the images dataset by using a Keras.

**Train/validation/test splits have been used, and the model uses dropout layers or other methods to reduce overfitting.**

You are doing a good work by splitting the training dataset into training and validation subsets. I am also happy to see that, you used dropout in the model architecture to reduce overfitting.

**Learning rate parameters are chosen with explanation, or an Adam optimizer is used.**

That's right! **Adam optimizer** [a method for stochastic optimization](https://arxiv.org/abs/1412.6980v8) was used.

**Training data has been chosen to induce the desired behavior in the simulation (i.e. keeping the car on the track).**

Nice work! Your model is very robust, it keeps the car in all the track.

**Architecture and Training Documentation**

**The README thoroughly discusses the approach taken for deriving and designing a model architecture fit for solving the given problem.**

The README looks good, but there are ways it can be improved upon. It's good because:

1. It clearly defines the problem at hand.
2. It includes a discussion of the model used to train the driving agent.
3. It outlines a problem faced during implementation of the solution, and how this problem was addressed.
4. The results and how to run the code

I greatly appreciated your discussion in the following terms: *"I used Nvidia model architecture directly since it was mentioned in one of the module that is a standard model being used by Nvidia and also from the old student Paul Heraty as mentioned in the Udacity module. To reduce overfitting I used a single dropout layer with probability of 0.6"*. Great job 

**Suggestions and Comments:**

The following link presents [14 design patterns improve convolutional neural network cnn architecture](https://review.udacity.com/(https:/www.topbots.com/14-design-patterns-improve-convolutional-neural-network-cnn-architecture/)

**The README provides sufficient details of the characteristics and qualities of the architecture, such as the type of model used, the number of layers, the size of each layer. Visualizations emphasizing particular qualities of the architecture are encouraged.**

* Great job here describing your architecture!
* I appreciated the provided visualization and comments about adjustment.

**The README describes how the model was trained and what the characteristics of the dataset are. Information such as how the dataset was generated and examples of images from the dataset must be included.**

You did a nice job in describing how did you train the model, especially a discussion was conducted on how dataset was generated.

**Suggestions and Comments:**

The writeup mentioned that, *I augmented data to generate 3 additional image for each image*. The following points are to be included in the writeup:

* More information on the number of epochs chosen, why you chose this number of epochs, etc.
* Examples of images from the dataset and generated data. Making things will be of great help here.

**Simulation**

**No tire may leave the drivable portion of the track surface. The car may not pop up onto ledges or roll over any surfaces that would otherwise be considered unsafe (if humans were in the vehicle).**

You made it! The car is able to navigate correctly on test data. The car drives successfully on the first (left) track in the simulator. However, the car gets dangerously close to the roadside at certain points(if humans were in the vehicle). I provided two related links for improvement.

**Suggestions and Comments:**

I suggest you these greats posts

* <https://medium.com/@ValipourMojtaba/my-approach-for-project-3-2545578a9319>,
* <https://chatbotslife.com/learning-human-driving-behavior-using-nvidias-neural-network-model-and-image-augmentation-80399360efee>  
  on medium to improve the work and allow the car to drive in the second track by using some specific preprocessing techniques.