**Self Driving/Behavioural Cloning Car Readme**

This is a self driving/behavioural cloning car trained using a Convolutional Neural Network (CNN).The Model used to train the self driving car is Nvidia’s End-to-End Deep Learning for Self Driving Cars Model (<https://developer.nvidia.com/blog/deep-learning-self-driving-cars/>). This model was used by Nvidia to train a real car, so it was optimal for a car running in a simulation environment. It uses Udacity’s open source Self Driving car simulator also used in their Self driving car nanodegree program. It is a supervised regression problem between the car steering angles and the road images in real-time from the cameras the car.

**Setup & Installation Requirements**

-Python 3.7.6

-Pycharm IDE

-Udacity Self Driving Car Sim (included in repo) : <https://github.com/udacity/self-driving-car-sim>

The following specific libraries need to be added in the Pycharm virtual environment

* opencv-python=4.2.0.34
* pandas=1.0.4
* numpy=1.18.5
* matplotlib=3.2.1
* sklearn=0.0
* tensorflow=2.2.0
* imgaug=0.4.0
* python-socketio=4.2.1
* eventlet=0.33.3
* Flask=2.2.5

**References:**

<https://towardsdatascience.com/how-to-train-your-self-driving-car-using-deep-learning-ce8ff76119cb>

<https://developer.nvidia.com/blog/deep-learning-self-driving-cars/>

<https://developer.nvidia.com/blog/explaining-deep-learning-self-driving-car/>

<https://www.codeproject.com/Articles/1273179/A-Complete-guide-to-self-driving-Car>

<https://www.computervision.zone/courses/self-driving-simulation-using-nvidias-model/>

<https://github.com/naokishibuya/car-behavioral-cloning>

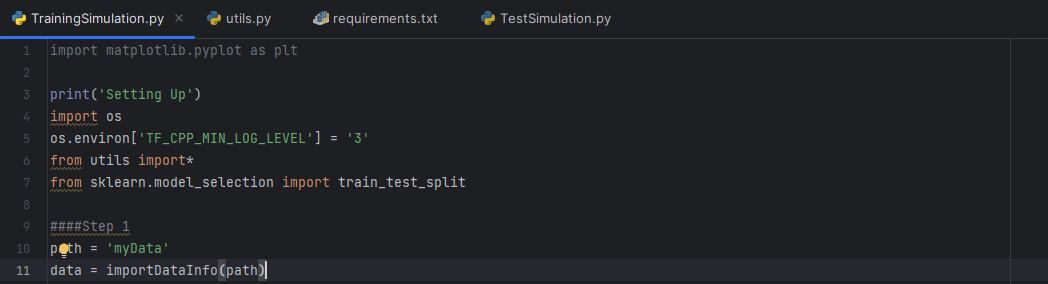
<https://kikaben.com/introduction-to-udacity-self-driving-car-simulator/>

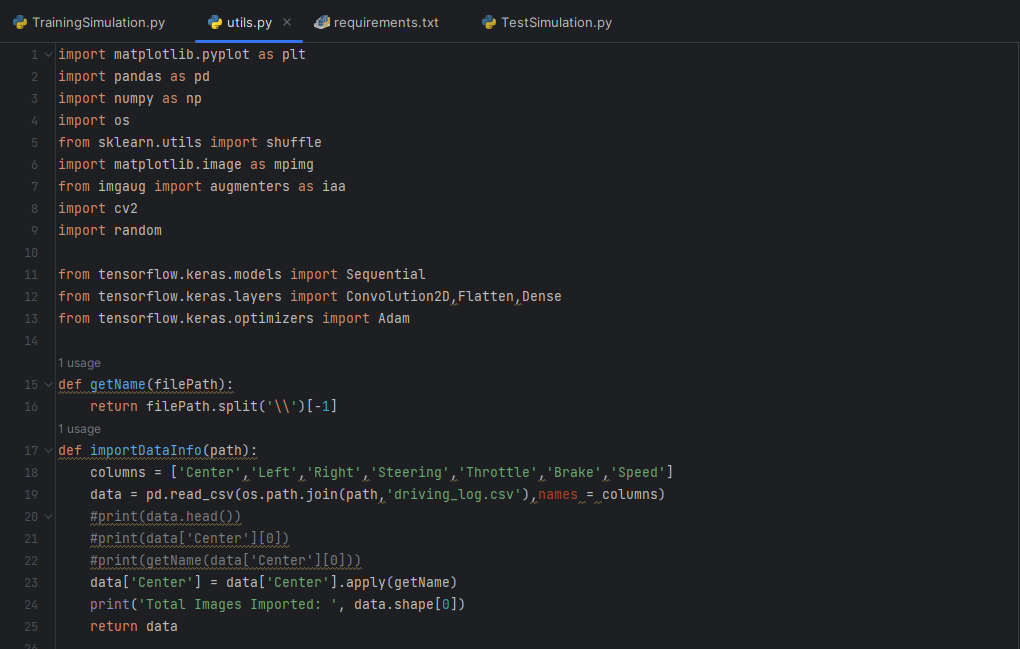
<https://github.com/udacity/self-driving-car-sim>

**Project workflow**

**Step #1:**

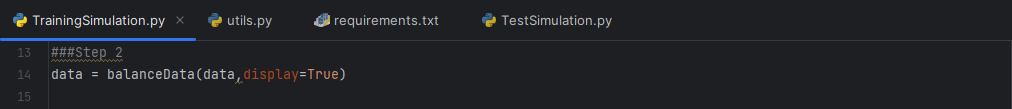
Import and process driving log data from the csv file, it only uses the center view of the camera ie. the center images





**Step #2:**

We balance the data that we import in step #1 so that our data isn’t biased and we remove redundant data. It is important that we do this to balance the data distribution by removing samples from overrepresented steering angle bins.

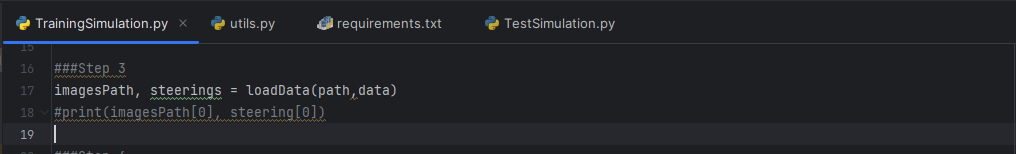


A screen shot of a computer program

Description automatically generated with low confidence

**Step #3:**

Prepare for loading & processing data for the model



A screenshot of a computer

Description automatically generated with medium confidence

**Step #4:**

Split dataset into Training and Validation sets.

This allows the model to train and evaluate. This then makes it possible to evaluate the model while training and make adjustments as necessary.

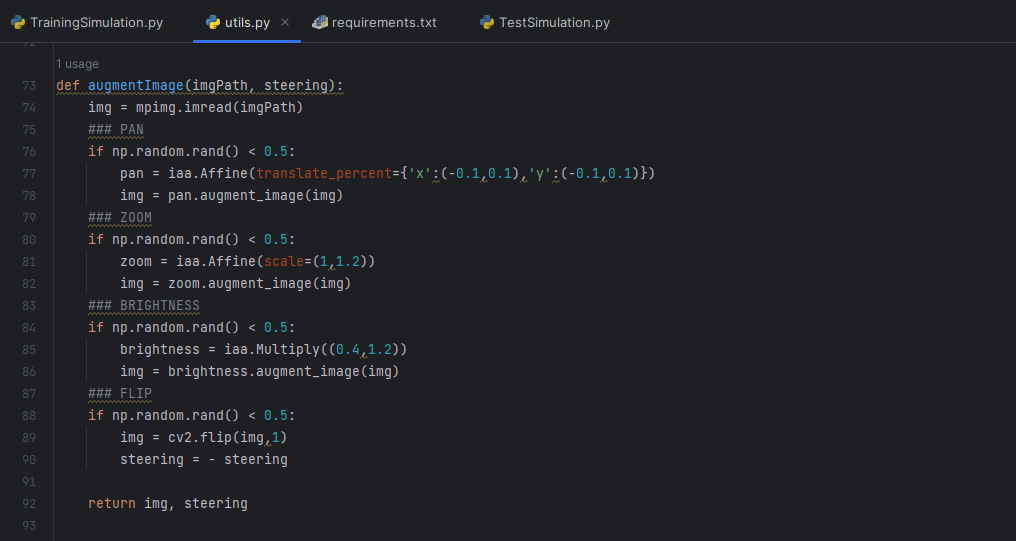
A screen shot of a computer

Description automatically generated with medium confidence

**Step #5:**

Augmentation of Images to Prepare for Training.

This is done to increase the diversity of training data fed to the neural network. It randomly translates the image horizontally and vertically. It randomly scales the image. It randomly adjusts the brightness of the image. It also randomly flips the image horizontally and adjusts the corresponding steering angle.



**Step #6:**

Pre-processing of input images before feeding into the neural network.

It crops the top and bottom parts of the image to remove irrelevant information. It converts the image from the RGB color space to the YUV color space. It applies gaussian blur to reduce noise and smooth the image

A screenshot of a computer

Description automatically generated with medium confidence

A screen shot of a computer

Description automatically generated with medium confidence

**Step #7:**

Batch generator. Generates batches of pre-processed or augmented image data and corresponding steering values. It is efficient as it generates image data with their corresponding steering data on the go, rather than loading the entire dataset at once.



**Step #8:**

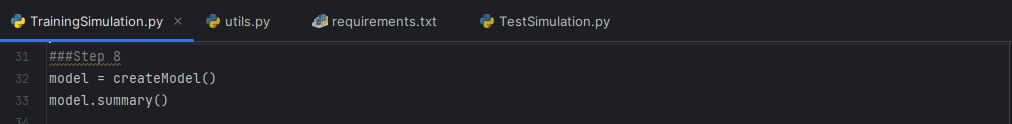
Creating the Nvidia Convolutional Neural Network model with fully connected layers.

The activation function used in this model is known as an Exponential Linear Unit(ELU), this type of activation function is often used in deep learning models. The Optimization function used in this model is known as Adam(Adaptive Moment Estimation) Optimization also commonly used in machine learning and deep learning. It’s advantages include adaptive learning rates, Momentum-based updates, Individual parameter updates and robustness to hyperparameters. The loss function used is know as the MSE(Mean Squared Error) function, it is widely used metric in regression problems in the machine learning and deep learning field.

A picture containing text, screenshot, design, receipt

Description automatically generated





**Step #9:**

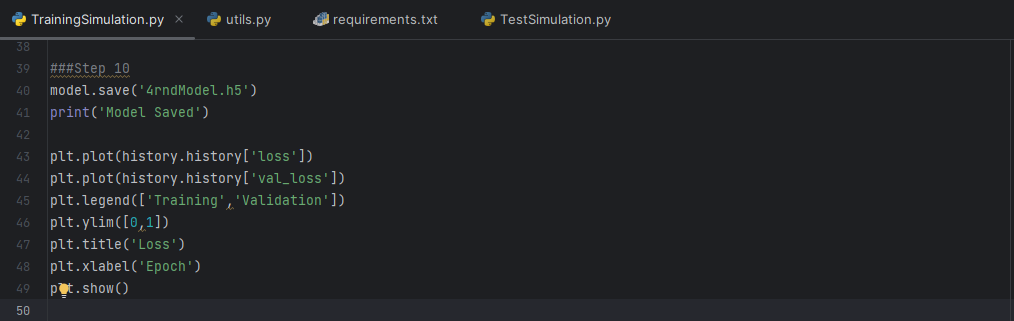
Training of model

A screenshot of a computer

Description automatically generated with medium confidence

**Step #10:**

Saving the model and plotting the loss graph



**Training the model**

A picture containing screenshot, diagram, text, line

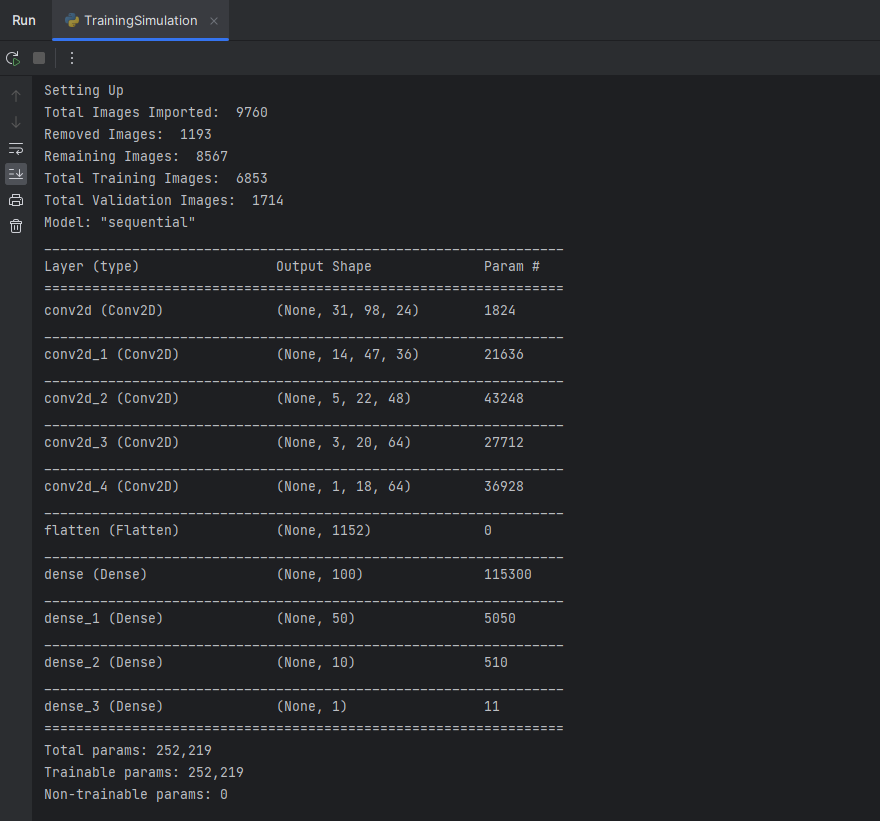
Description automatically generated

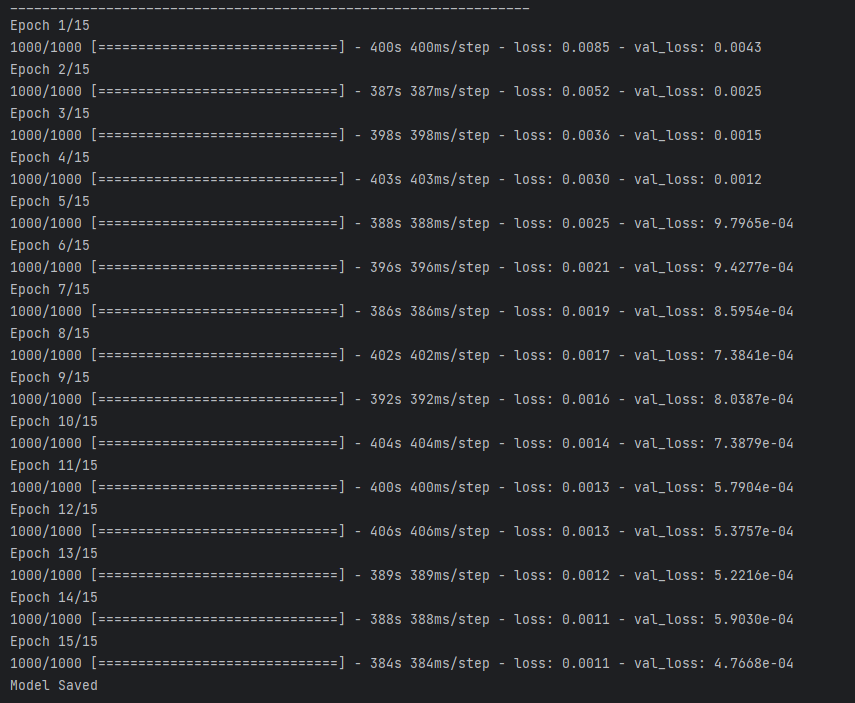
A picture containing screenshot, diagram, text, line

Description automatically generated

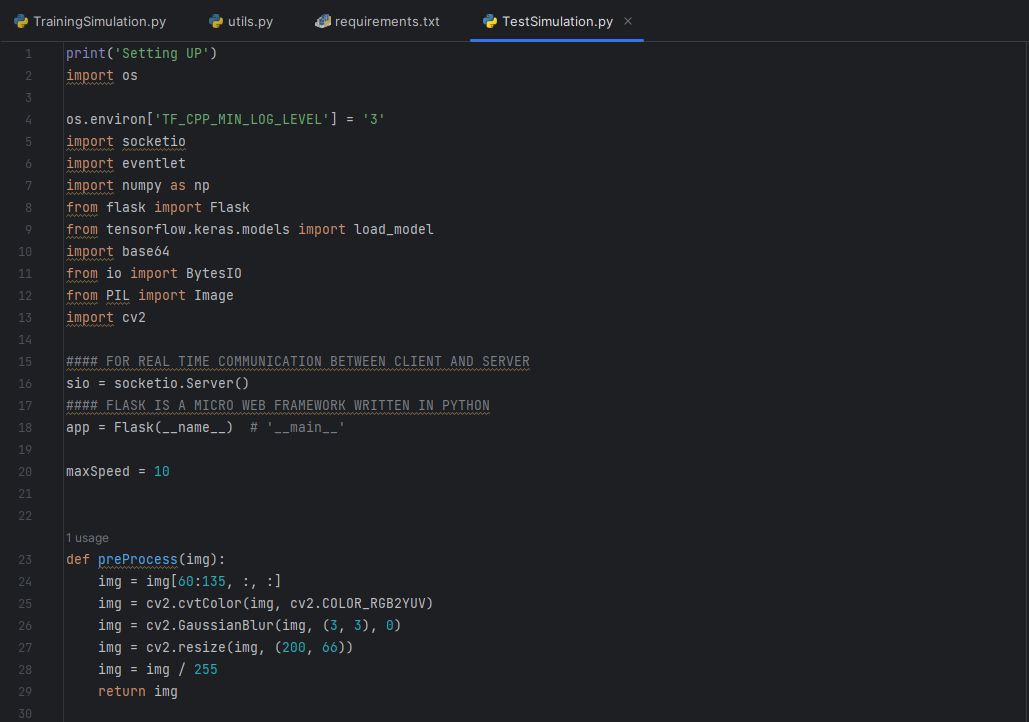
A picture containing text, screenshot, diagram, line

Description automatically generated





**Testing the model:**



A screen shot of a computer program

Description automatically generated with medium confidence