**#\*\*Behavioral Cloning\*\***

##Writeup Template

## Rubric Points

###Here I will consider the [rubric points](https://review.udacity.com/#!/rubrics/432/view) individually and describe how I addressed each point in my implementation.

###Files Submitted & Code Quality

####1. Submission includes all required files and can be used to run the simulator in autonomous mode

My project includes the following files:

\* model.py containing the script to create and train the model

\* **drive.py** for driving the car in autonomous mode

\* **“model.h5“** containing a trained convolution neural network

\* **“BehavioralCloning.pdf”** summarizing the results

####2. Submission includes functional code

Using the Udacity provided simulator and my drive.py file, the car can be driven autonomously

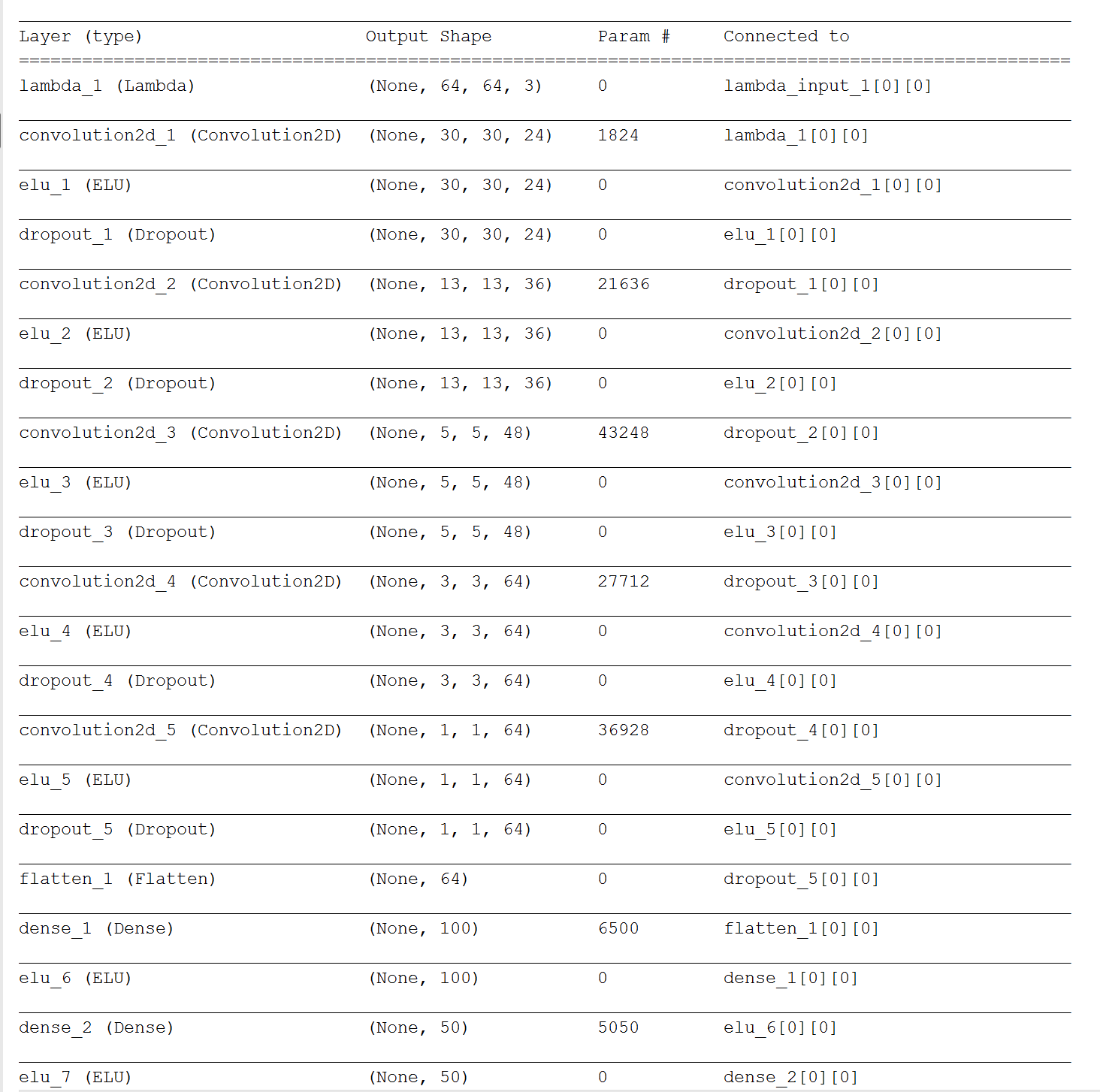
####3. Submission code is usable and readable

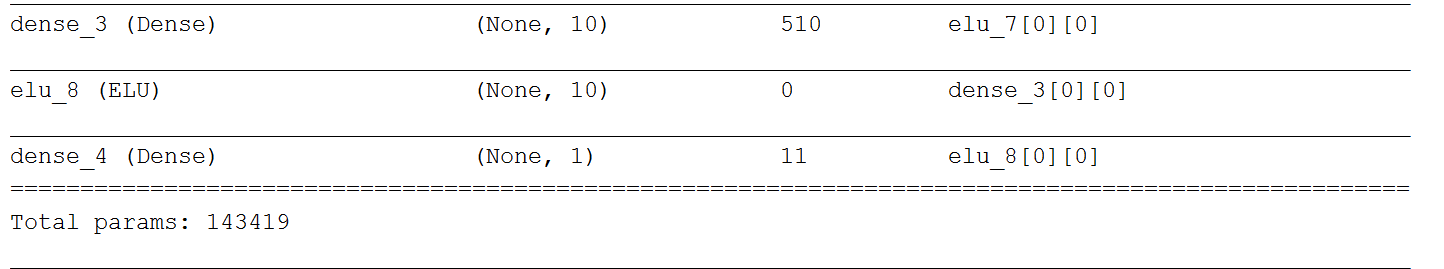
The model.py file contains the code for training and saving the convolution neural network. The file shows the pipeline I used for training and validating the model, and it contains comments to explain how the code works.

###Model Architecture and Training Strategy

####1. An appropriate model architecture has been employed

Similar with NVIDA model, My model consists of a convolution neural network with **three times 5x5 filter and 2 times 3x3 filter** (model.py **lines 299-333**) **. See network structure summary in below:**





The model includes ELU layers to introduce nonlinearity , and the data is normalized in the model using a Keras lambda layer (code line **300**).

####2. Attempts to reduce overfitting in the model

The model contains dropout layers in order to reduce **overfitting (model.py lines between 305 and 318).**

The model was trained and validated on **different data sets** to ensure that the model was not overfitting (code line **between 27 and 91** ).

####3. Model parameter tuning

The model used an adam optimizer, so **the learning rate was manually tuned to 0.0001.** (model.py **line 327**).

####4. Appropriate training data

Training data was chosen to keep the vehicle driving on the road. I used a combination of center lane driving, recovering from the left and right sides of the road.

**0.25 is added to the steering angle for the images from left camera.**

**0.25 is reduced to the steering angle for the images from right camera. See model.py line 87 to 91.**

Besides, **three different data sets** **are used for training including udacity data, normal driving data and data with opposite driving direction**.