# Behavioral Cloning

## Writeup Template

### You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

---

**Behavioral Cloning Project**

The goals / steps of this project are the following:

- Use the simulator to collect data of good driving behavior
- Build, a convolution neural network in Keras that predicts steering angles from images
- Train and validate the model with a training and validation set
- Test that the model successfully drives around track one without leaving the road
- Summarize the results with a written report

## Rubric Points

### Here I will consider the rubric points 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
- writeup_report.md or writeup_report.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 around the track by executing

```
python drive.py model.h5
```

### 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

私は、モデル作成のために基本となる Training The Network(4 章)、LeNet(9 章)、NVIDIA Architecture(14 章)に基づく訓練を行い、各動作を確認しました。結果、最もうまく車両を動かせた NVIDIA Architecture を採用しました。

(補足：訓練方法の切り替えは Model.py の 115 行目に示す Lesson の値で選択できます。)

このモデルは、非線形性を導入するための RELU レイヤーを含み、データは Keras ラムダレイヤーを使用したモデルの正規化を行っています。またクロッピング処理を行い教師データから不要な背景を削除しました。

### 2. Model parameter tuning

モデル作成について、最適化のアルゴリズムには Adam を使用、損失評価には MSE（平均二乗誤差）を使用しました。

### 3. Appropriate training data

トレーニングデータには、センターカメラ、左カメラ、右カメラの映像を組み合わせました。.

トレーニングデータ作成方法の詳細については、次のセクションを参照してください。

## Model Architecture and Training Strategy

### 1. Solution Design Approach

まず、Project Resources（第1章）より、サンプルトレーニングデータを取得し、車両走行の基本とする教師データとして訓練に与えました（model.py の 19 行目 ubu_file01）。

次に実際に道路中央を逸脱しないよう正確にコースを周回して、理想的な教師データを2つ分追加しました（model.py の 21 行目 win_file01,22 行目 win_file02）。

さらににコースを逆周回した教師データを2つ分追加しました（model.py の 23 行目 win_file03,24 行目 win_file04）。

最後に路面の色が変化する「橋の上」での走行データを追加しました（model.py の 20 行目 ubu_file02）

センターカメラ、左カメラ、右カメラの映像を教師データに採用したため、データの数が多くなりました。そこで、私は操舵の小さい時に撮影した教師データを間引きました。これは直線よりもコーナを走行する際の車線追従性能を向上させる目的を兼ねています。

また、変化に乏しい同じような教師データではユニーク特徴がつかみにくく、訓練結果が過適合となる恐れがあると判断して、教師データを 200 個ずつスキップさせながら訓練する方法を採用しました。

さらに、各教師データと操舵データを反転させた教師データを追加し、データに含まれる各特徴を多様化させました。

結果、私の model.h5 を用いた車両は、自律的にコースを周回することができました。

## 2. Final Model Architecture

前述の 1. An Appropriate Model Architecture Has Been Employed に示すよう私のモデルは NVIDIA Architecture(14 章)に基づき作成しました。具体的なコードは Model.py の 198 行目～209 行目に示します。

※アーキテクチャを視覚化することは、プロジェクトのルーブリックに従ってオプションということなので詳細は省きます。

## 3. Creation of the Training Set & Training Process

私の検討した内容は前述の「An appropriate model architecture has been employed 回答」、「 Model parameter tuning 回答」、「Solution Design Approach 回答」に示したとおりです。詳細は省きます。