Behavioral Cloning (writeup Template)

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

**Image Reference : Udacity Sample data

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

My project includes the following files:
* model1.py containing the script to create and train the model
* drive.py for driving the car in autonomous mode
* model1.h5 containing a trained convolution neural network
* writeup

2. Submission includes functional code

Using the Udacity provided simulator and my drive.py file, the car can be driven autonomously aro
und the track by executing

python drive.py model1.h5

3. Submission code is usable and readable

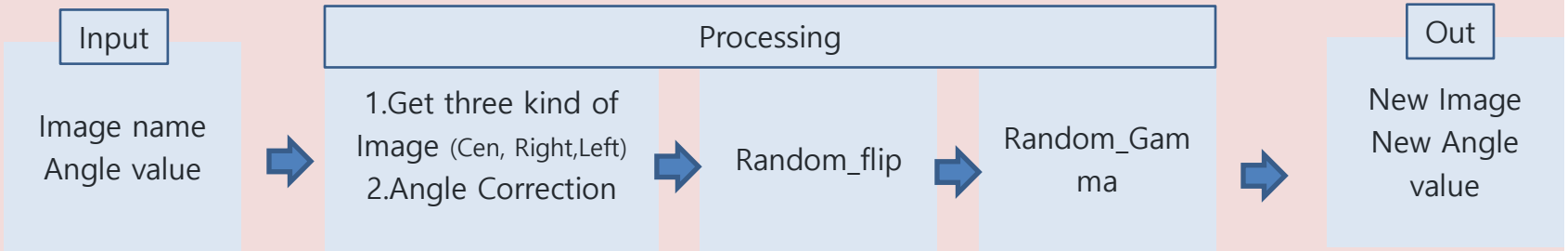
The model.py file contains the code for training and saving the convolution neural network. The file shows the pipeline I used for training and validating the model, and it contains comments to explain how the code works.

Data Capturing

1. Image Data

- I use sample_datas from Udacity (Its difficult for me to train myself in my computer)
- Datas consist of 24, 111 images (8037 * 3 : center, left, right cameras).
- I split datas for train(90% : 21699) and validation(10%:2412)
- And I modify steering_angle value for left, right cameras
  : Steering Correction value is 0.226

2. Image Processing



※ Get three kind of Image

```
name = image_file + batch_sample[0].split('/')[-1]  for center camera
name = image_file + batch_sample[1].split('/')[-1] for left camera
name = image_file + batch_sample[2].split('/')[-1] for right camera
```

2. Image Processing

※ Angle Correction

Center angle = Center angle
Left angle = Center angle + Correction value (0.226)
Right angle = Center angle - Correction value (-0.226)

※ Random_flip

Using random.randint(0,1)
if the value is 1, it will return flip Image (need steering angle correction)

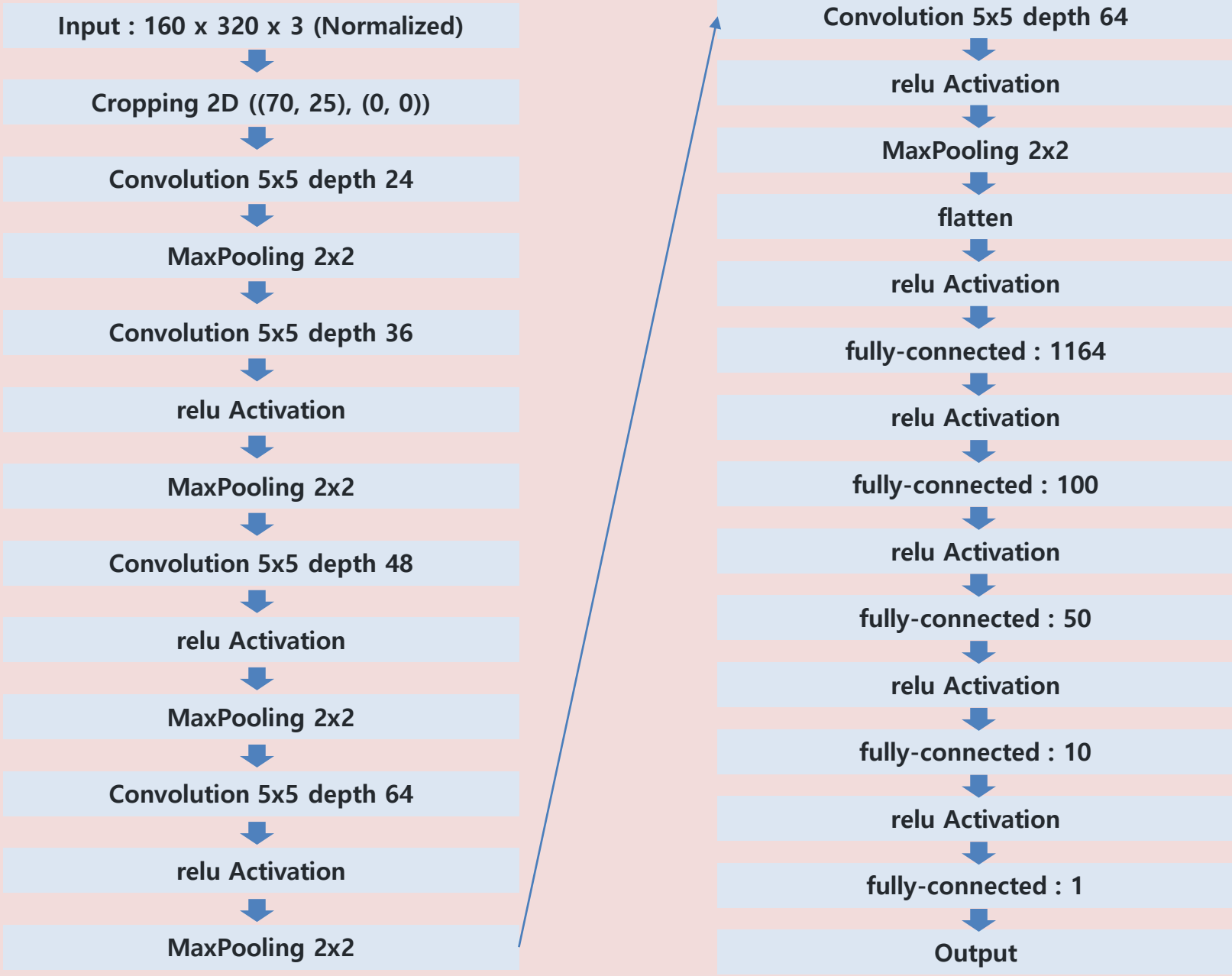
※ Random_gamma

apply randomly gamma correction

discussion

At first I tried to create my own training data. but its difficult for me to do that because of my computer. so I use sample datas a lot. Not only center images, but also left/right cameras images

Model Architecture



Model Architecture

※ Tune Parameter

- batch_size = 32
- epochs = 2

※ Create two generators for training and validation

- train_generator = generator (train_samples)
- validation_generator = generator (validation_samples)

Generator Function

```
num_samples = len(samples)
    while 1: # Loop forever so the generator never terminates
        shuffle(samples)
        for offset in range(0, num_samples, batch_size):
            batch_samples = samples[offset:offset+batch_size]

            images = []
            angles = []

            for batch_sample in batch_samples:
                # for center image
                angle = float(batch_sample[3])
                name = image_file + batch_sample[0].split('/')[-1]
                image = cv2.imread(name)
                processed_image, processed_angle = generate_new_image(image, angle)
                images.append(processed_image)
                angles.append(processed_angle)

                                                    ……

            X_train = np.array(images)
            y_train = np.array(angles)
            yield shuffle(X_train, y_train)
```

Model Architecture

discussion

- I tried to use only one kind of data among center, left, right using random function.
  But when run drive.py it didn't satisfy requires. So I use all of camera datas

- At firtst I use Dropout function to avoid overfitting. but when it works worse than
  I don't use dropout function. I think its from the lack of train data (I only use training data).
  and also

- When I select batch_size 64, training time is very much. so changed batch_size 8.
  It worked so fast but loss is too big. and then finally I choose batch_size 32

- And the last thing my model look like having a week about no lane images
  although there is curve, my model arent ready for turning (after a moment it turns)
  need more no lane images datas I think

Result

※ Tune Parameter
- batch_size = 32
- epochs = 2

※ Create two generators for training and validation
- train_generator = generator (train_samples)
- validation_generator = generator (validation_samples)

Generator Function

```
num_samples = len(samples)
    while 1: # Loop forever so the generator never terminates
        shuffle(samples)
        for offset in range(0, num_samples, batch_size):
            batch_samples = samples[offset:offset+batch_size]

            images = []
            angles = []

            for batch_sample in batch_samples:
                # for center image
                angle = float(batch_sample[3])
                name = image_file + batch_sample[0].split('/')[-1]
                image = cv2.imread(name)
                processed_image, processed_angle = generate_new_image(image, angle)
                images.append(processed_image)
                angles.append(processed_angle)

                                                    ……

            X_train = np.array(images)
            y_train = np.array(angles)
            yield shuffle(X_train, y_train)
```