Behavioral Cloning

Said Zahrai

August 8, 2019

Figure 1: Simulations performed at three speeds: default in drive.py with speed set to 9 MPH (left), driving speed set to 20 MPH (middle) and speed set to 30 MPH (right).

1 Introduction

The goal of this project is to clone the behavior of a driver driving a car in a Unity game. A player needs to play first and record data. While the player is playing, the environment is observed with three cameras mounted in the middle of the car, and in the two left and right corners.

2 Data

2.1 Camera recorded images

Figure 2: Camera recorded images. Left, seen from left camera, center from center camera and right from right camera.

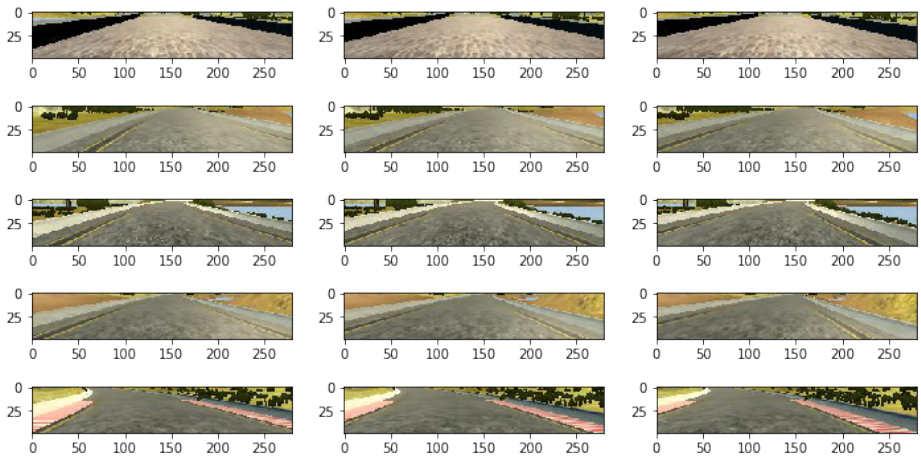

Figure 3: Corped images with focus on interesting region. Left, seen from left camera, center from center camera and right from right camera.

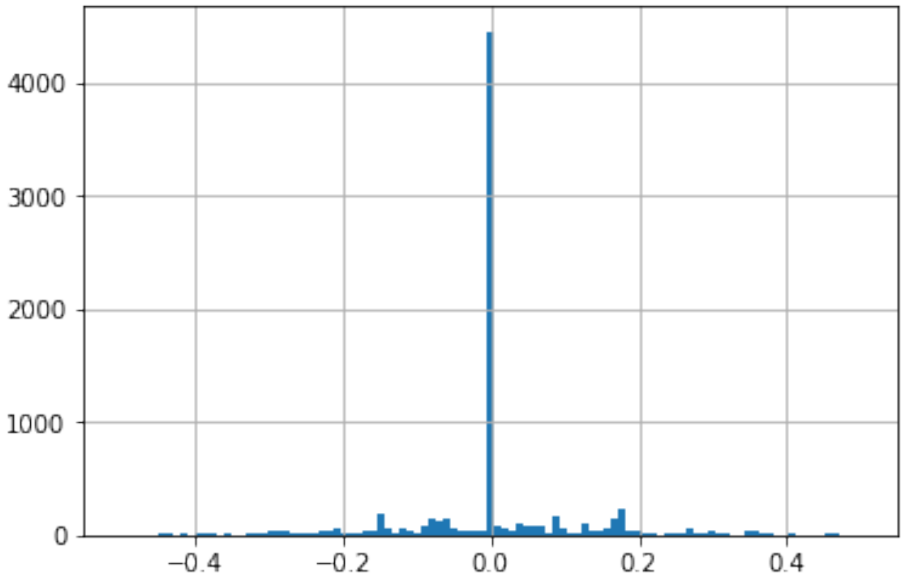

Figure 4: Distribution of steering values.

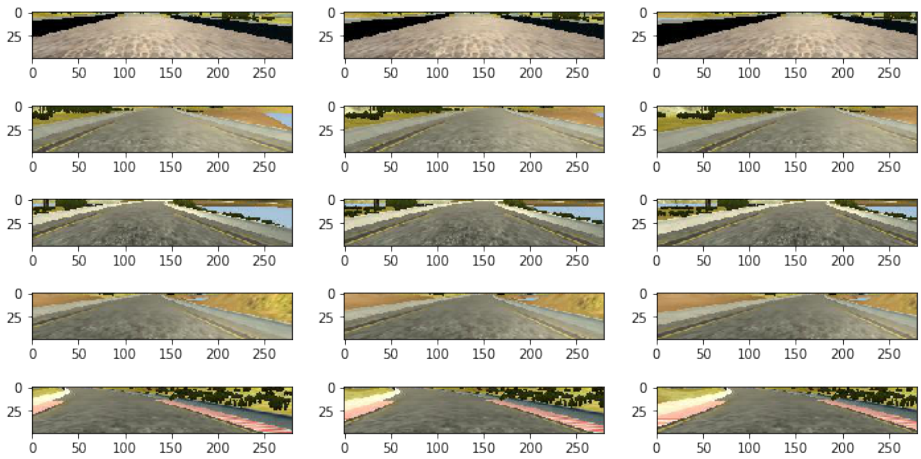

Figure 5: Shifted images: Left, shifted 20 pixels to left, center, original image and right, shifted 20 pixels to right.

2.2 Data augmentation

```
x.append(left_image_address)
y.append(steering_value+0.25)
x.append(left_image_address)
y.append(steering_value-0.25)
```

Group A:

```
img = plt.imread(image_file_address)
if (abs(batch_sample[1])<0.01):
    r = random.randint(0,400)
    if (r<20):
      images.append(img)
      angles.append(steering_value)
      images.append(np.fliplr(img))
      angles.append(-steering_value)
    elif (data_augmentation):
      r =random.randint(-40,40)
      if (abs(r)<21):
        M = np.float32([[1,0, r],[0,1,0]])
        images.append(cv2.warpAffine(img,M,(cols,rows)))
        angles.append(steering_value+0.01*r)
```

Group B:

```
img = plt.imread(image_file_address)
if (abs(batch_sample[1])<0.01):
    r = random.randint(0,400)
```

```
if (r<20):
  images.append(img)
  angles.append(steering_value)
  images.append(np.fliplr(img))
  angles.append(-steering_value)
elif (data_augmentation):
  r =random.randint(10,20)
  M = np.float32([[1,0, r],[0,1,0]])
  images.append(cv2.warpAffine(img,M,(cols,rows)))
  angles.append(batch_sample[1]+0.02*r)
  r =random.randint(-20,-10)
  M = np.float32([[1,0, r],[0,1,0]])
  images.append(cv2.warpAffine(img,M,(cols,rows)))
  angles.append(steering_value+0.02*r)
```

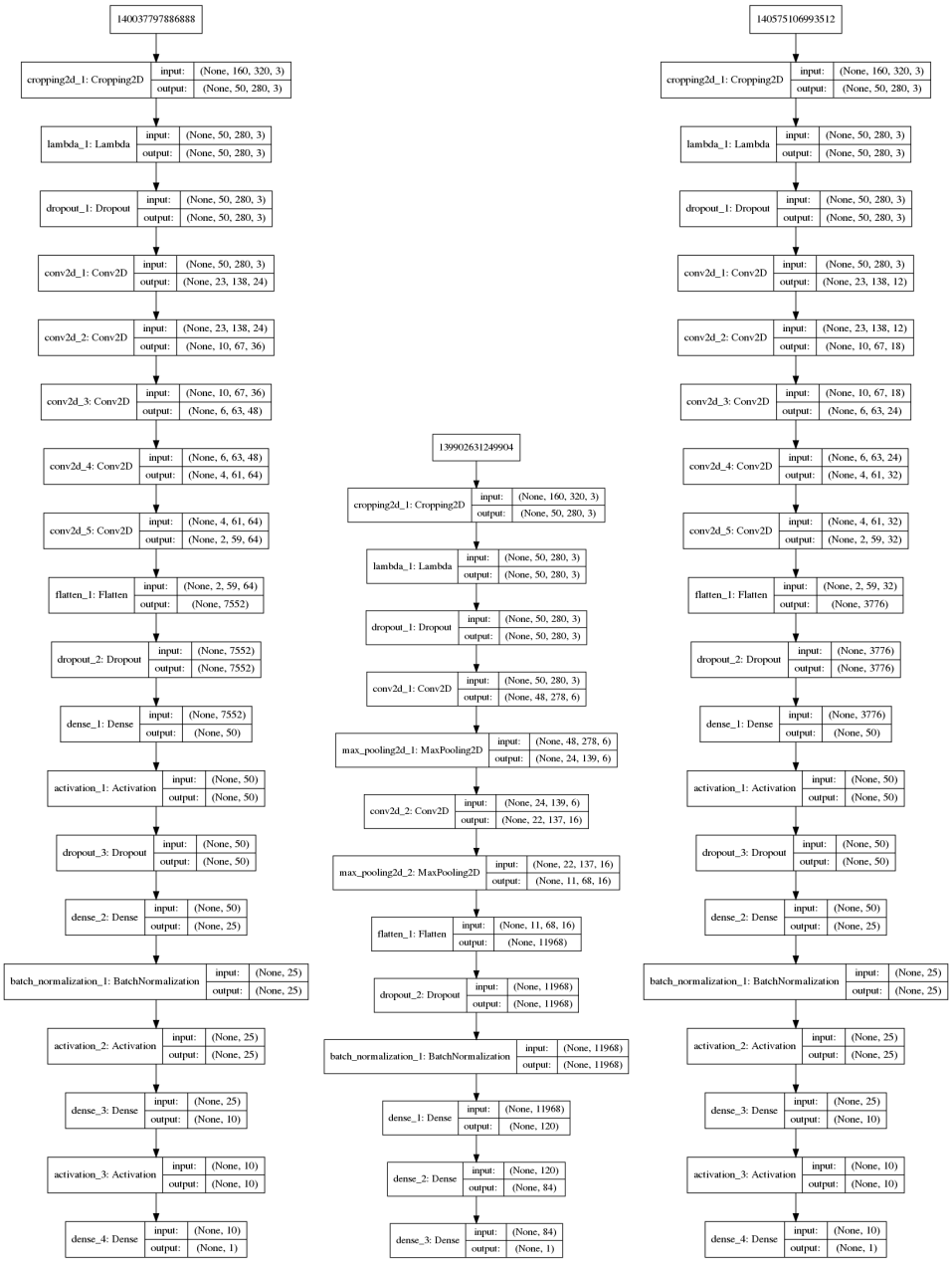

Figure 6: Model structure: Left, Nvidia's network, center, Lenet's network and right, reduced version of Nvidia's network.

	Nvidia	Lenet	Reduced Nvidia
Total params	510,644	1,495,449	223,842
Trainable params	510,594	1,471,513	223,792

Table 1: Model structure.

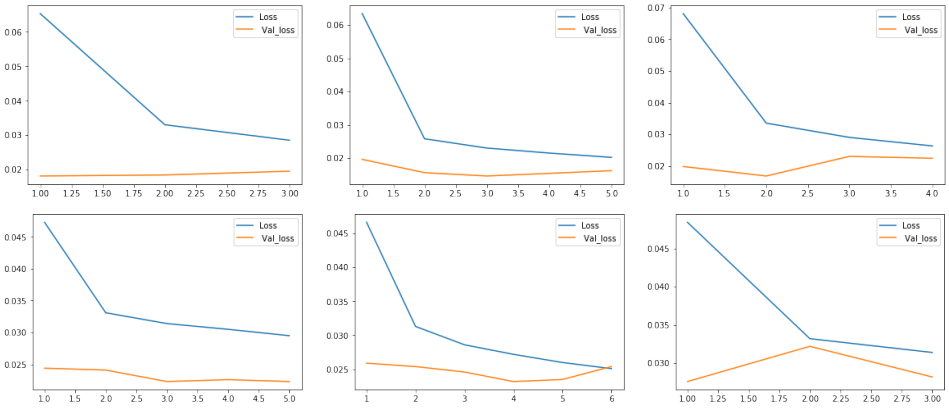

Figure 7: Convergence history for group A. Above row with data shared by Udacity and below with additional 2 more runs, i.e. 3 sets of data. Left, Nvidia's network, center, Lenet's network and right, reduced version of Nvidia's network.

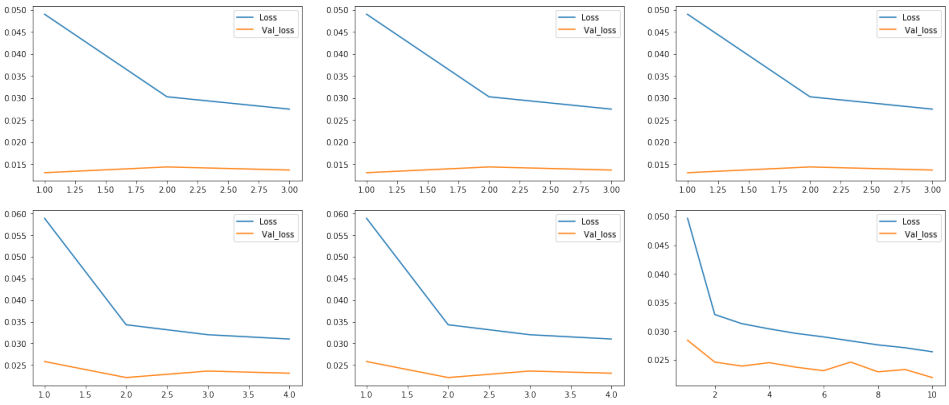

Figure 8: Convergence history for group B. Above row with data shared by Udacity and below with additional 2 more runs, i.e. 3 sets of data. Left, Nvidia's network, center, Lenet's network and right, reduced version of Nvidia's network.

Speed	Nvidia	Lenet	Reduced Nvidia
9 MPH	OK	OK	OK
20 MPH	OK	Touches lane borders	Touches lane borders
30 MPH	OK	Passes lane borders	Drives out from track