Autonomous Car Driving using Image Segmentation

PGP AIML July 2019: Capstone Interim Report

Team 6 : The Gaussians

3/10/2020

Contents

[1. Proposed Solution 3](#_Toc34766261)

[2. Evaluation metrics 4](#_Toc34766262)

[3. Exploratory Data Analysis 4](#_Toc34766263)

[4. Exploratory visualization 6](#_Toc34766264)

[5. Summary of Initial Findings 7](#_Toc34766265)

[Data Generation Approach 8](#_Toc34766266)

[Our learning: 9](#_Toc34766267)

[Overall Approach 9](#_Toc34766268)

[6. Challenges 9](#_Toc34766269)

[7. Next Steps 10](#_Toc34766270)

[8. Appendix: 10](#_Toc34766271)

## Proposed Solution

Autonomous driving is a complex field that is still under research involving Artificial Intelligence and Machine Learning (AIML) to impart autonomous capabilities onto vehicles on the move. This capstone project is a small step in exploring ways to begin analyzing the traffic on a road, understanding the constituents of the images captured via vehicle sensors, figuring out the object of interest in the images captured and enabling the right decisions.

Given the challenges that was encountered in capturing the images on the road(detailed in sections to follow) the data generation will rely on using a simulator that is capable of launching a town, deploying vehicles on the road and generating images via the sensors configured on the vehicle of interest to us. The simulator used here is CARLA ([http://www.carla.org](http://www.carla.org/)) which is capable of these features and much more.

Following are the broad level steps for the proposed solution:

* Launch Carla UE4 client, spawn 60 to 150 vehicles and set the sensor ticks to capture an image every 2 seconds
* Prepare the color map that can be used for annotation during the image segmentation
* The output is generated as RGB image and corresponding Segmented image as below:

|  |  |
| --- | --- |
| 000995.png | 000994.png |
| 000997.png | 000998.png |

* Once we have sufficient number (say 10000) of such RGB & segmented images, the data will be partitioned into training, test & validation set.
* The RGB images will be used as **features** & segmented images will be used as **labels** for the training
* The training set will be fed onto FCN (Fully Convolutional Networks) which are adept at learning to classify pixel-wise.
* The technique involves passing the image through several convolutional layers to understand the features and then instead of following it up with a dense layer(as it happens conventionally) use a 1x1 convolution layer to get the segmentation value which is then enlarged using upsampling to match the input size.
* Once the training is done the model is validated & tested using the validation & test set to validate & fine tune subsequently.
* In a real world scenario, once the segmented image is available, the road is identified using the corresponding segmentation and distance from the vehicle in front(if any) is calculated. If this distance is higher than the threshold then it is direction to ‘Go’ (As it can be observed from the image samples above the distance from the vehicle in front is very minimal in the first set and the decision would be ‘No Go’ in this case and the decision in the second set would ‘Go’ considering that there is ample distance i.e. beyond the set threshold).

The exploration will begin using the Google Colab resources and reconsidered either for workarounds if there are challenges in completing the classification or exploration of alternative compute resources.

## Evaluation metrics

Our solution contains two parts- Image segmentation and Driving assistance based on segmented image,

* mPA ( mean pixel accuracy) and mCIoU(mean class intersection over union) with the ground truth.
* Collision (target colliding with other object) rate for decision made on each frame.

**Mean Pixel accuracy:**  In segmentation each pixel of a given image is being classified, hence in order to evaluate the performance of a model we need to keep track of the result for each pixel.

Pixel accuracy 🡪 PA = (no. of pixels classified correctly for an image/ total no. of pixels for an image)

Mean PA 🡪mPA = (total pixels classified correctly for each image/ total no. of pixels)\*(1/n)

n🡪 size of the dataset/total no. of images

**Mean Class Intersection over Union**: In order to get the performance for each target class of a given dataset we use mCIoU. Intersection means, the overlap area of a class’s predicted and ground truth whereas union means the combined area of class’s both predicted and ground truth.

CIoU = Intersection/ Union

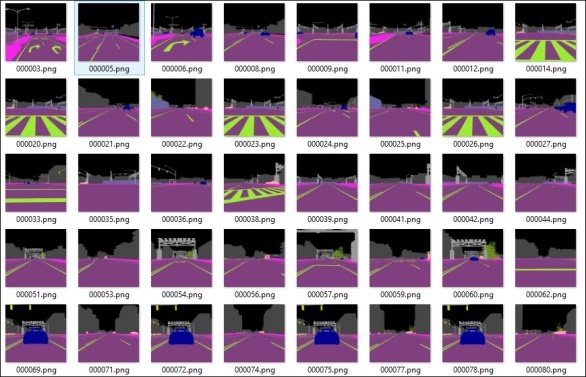
mCIoU = Summation{1:n}[( Intersction/union)]\*(1/n)

n🡪 size of the dataset/total no. of images

## Exploratory Data Analysis

About 1000 image samples were generated using CARLA (0.9.5) that included both the RGB images and their corresponding segmented views. These images are being analyzed for class-imbalance and redundant images/overlapping images.

**RGB Image Segmented Image**



**Determination of Class Imbalance:**

**Step1**: Upload images on to Google drive & use Google Colab with processor type set to TPU

**Step2:** Import the required libraries, define the label values and label the classes to identify the each of the segmented images (color map information).

**Import glob**

data=[]

path="/content/drive/My Drive/dataset/carla seg\_images/seg/"

label\_value=[[0,0,0],[70,70,70],[153,153,190],[160,170,250],[60,20,220],[153,153,153],[50,234,157],[128,64,128],[232,35,244],[35,142,107],[142,0,0],[156,102,102],[0,220,220]]

label\_class=['Unlabeled','Building','Fence','Other','Pedestrian','Pole','Road line','Road','Sidewalk','Vegetation','Car','Wall','Traffic sign']

**Step3:** Read each of the image color and append to data for identifying the shape

**For** filename **in** glob.glob(path+"\*.png"):

Img = cv2.imread(filename,cv2.IMREAD\_COLOR)

data.append(img)

**Step4:** Convert the data into a Numpy array to make the computation faster and get each pixel value of the whole dataset.

data=np.array(data)

print(data.shape)

result\_=[]

**for** I **in** range(data.shape[0]):

**for** j **in** range(data.shape[1]):

**for** k **in** range(data.shape[2]):

result\_.append(data[i,j,k])

**Output**: (500, 512, 512, and 3)

**Note**: Since the height and width of the images have been defined in CARLA there is no need to reshape it.

**Step5:** Identify the unique set of pixel values (of corresponding classes)

class\_pixels=np.unique(result\_,axis=0,return\_counts=**True**)

class\_pixels\_value,class\_pixels\_count=np.array(class\_pixels[0]).tolist(),np.array(class\_pixels[1]).tolist()

print(class\_pixels)

**Output**:

(array([[ 0, 0, 0],

[ 0, 220, 220],

[ 35, 142, 107],

[ 50, 234, 157],

[ 70, 70, 70],

[128, 64, 128],

[142, 0, 0],

[153, 153, 153],

[153, 153, 190],

[156, 102, 102],

[160, 170, 250],

[232, 35, 244]],

dtype=uint8),array([26213866, 95067,12736903,942411, 9220968,35962906,32503257, 1330844, 115822,8655522,129932, 3164502]))

**Step6:** Decode the pixel values to get their respective class and their counts

class\_counts={}

*#comp = map(lambda x,y: x-y, val1,val2)*

**for**val1**in**class\_pixels\_value:

**for**val2**in**label\_value:

comp=map(**lambda**x,y:abs(x-y),val1,val2)

*#comp = abs(val1-val2)*

**if**max(comp)==0:

i,j=class\_pixels\_value.index(val1),label\_value.index(val2)

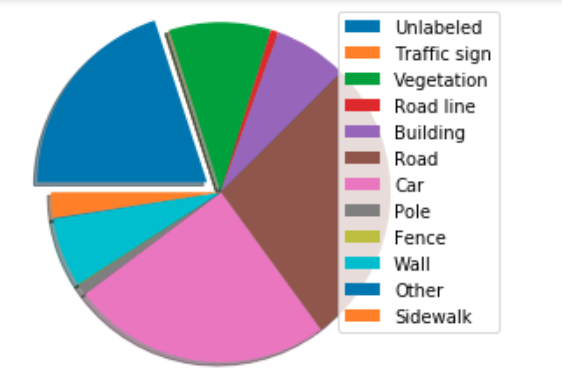
class\_counts.update({label\_class[j]:class\_pixels\_count[i]})

**Output:** {'Unlabeled': 26213866, 'Traffic sign': 95067, 'Vegetation': 12736903, 'Road line': 942411, 'Building': 9220968, 'Road': 35962906, 'Car': 32503257, 'Pole': 1330844, 'Fence': 115822, 'Wall': 8655522, 'Other': 129932, 'Sidewalk': 3164502}

## Exploratory visualization

**Class Imbalance Visualization**

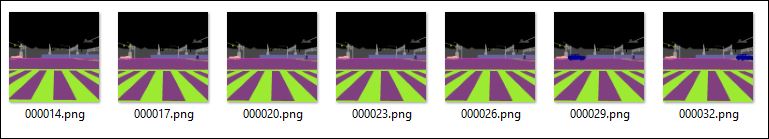
As per the pie chart representation below, current dataset has 70% of the label belonging to the classes (Road, Car & Unlabeled) other 30% of the labels contain 9 different classes. This clearly says that the dataset is multi-class imbalanced. Due to multi-class imbalance, most of the classified pixel lies in these 3 dominant classes. To overcome the above problem loss function will be customized to add weights to the classes based on their frequency.



**Redundant Data Visualization:**

Redundant image are similar to multiple duplicate images where each pixel values of given images are similar to each other at the corresponding coordinates between them.

**For example:**



Exploring following workarounds to overcome redundant data issue:

1. *Sampling :* Experiment with image capture duration (say 1 image every 3 seconds)
2. *Manual Drop:* Notwithstanding the sampling step above there still could be redundant images. In such cases manual review will be done to drop the images from the samples

## Summary of Initial Findings

Towards achieving our objective (semantic segmentation for driving assistance) the progress made has been listed below:

|  |  |
| --- | --- |
| Scope | Planned to implement in multiple phases, based on our progress the scope gets widened. Initial scope is defined to cover Phase 1 [Look for the phase details at last page ]. |
| Some Research Papers/Artifacts went through | **For data:**  IIIT Hyderabad and Intel Release World’s First Dataset for Driving in India  [https://www.intel.ai/iiit-hyderabad-and-intel-release-worlds-first-dataset-for-driving-in-india/#gs.yxhsdj]  Carla Simulator  [<http://carla.org/>]  "City Car Driving" Simulator  [https://citycardriving.com/]  Waymo  [https://waymo.com/]  **Model for segmentation:**  FCN for semantic segmentation  [[https://people.eecs.berkeley.edu/](https://people.eecs.berkeley.edu/fcn.com)]  **For implementation:**  Using FCN architecture to implement semantic segmentation  [<https://medium.com/coinmonks/semantic-segmentation-deep-learning-for-autonomous-driving-simulation-part-1-271cd611eed3>] |
| Data Generation Approach | Tried various options and finalized with a simulator. |
| Data Generation | In Progress – So far, around 200 Images are generated. However, we need to regenerate the images by fixing few parameters to ensure the right outcome of the model. |
| Data Analysis | Analyzed the data for no. of classes, class imbalance, redundancy, etc. |
| Model architecture | FCN architecture ( built on top of VGG-16 ) |
| Develop model | Yet to start |
| Model Testing Strategy | Finalized |
| Model Implementation Strategy | Finalized |
|  |  |

### Data Generation Approach

Explored below approaches for data Generation:

1. ***Manually recording through dashboard camera:***

Recording the data by manually driving in public road using the dashboard camera, this approach was heavily time consuming and the recorded data found to have lots of noise present due to our imprecision in recording. It requires a higher supervision to produce a good data through this approach.

1. ***Carla Driving Simulator :***

A real world driving simulator designed especially for autonomous driving data generation. It gives the user full control over the data generation: determining no. of actors (cars, trucks, bikes & pedestrians), customizing the environment (for e.g. weather), configuring different sensors (cameras, LIDAR, collision sensor), controlling the spectators etc. These customizable options that CARLA provides help to generate the required RGB images and the corresponding segmented images. This additional capability eases out the otherwise mammoth exercise of manual data annotation.

### Our learning:

1. Generating a smooth data (with less noise) for a given process.
2. Analyzing a segmented image.
3. Advantages of using convolutional layers over fully connected layers.
4. Dealing with class imbalance in segmented images.

### Overall Approach

1. Generate data (RGB images).
2. Get the pixel wise annotations for semantic segmentation using CARLA.
3. Analyze the data for class imbalance and redundancy.
4. Build the model (FCN architecture – uses pre-trained VGG16) which takes in RGB image as an input and classifies each pixel resulting a segmented image.
5. From the output segmented image mask out only the road class and use vertical edge detection to identify the current lane.
6. Conduct spatial analysis on the lane information (identifying obstacles, turns) and provide driving assistance for each given frame (slow, accelerate, stop, driving angle).

## Challenges

Following are some of the challenges faced in the project this far:

1. Data: The real world data collected through video camera was shaky and thereby requiring a lot more recording and manual sifting through for good quality images that can be used for training and testing
2. Time/effort for experimentation in selecting the right approach for data generation – as model heavily relies on the input data, explored multiple options and evaluated the pros and cons of each approach
3. Labeling the images: Manually labeling 1000 - 3000 images might be difficult. However, our focus is on the surface of the road and hence exploring Semantic Segmentation.
4. Computation: Since we chose the image size (512\*512 since it is synthetic data) the time for processing the data is little high.

Here are the risks/challenges we continue to track and monitor

1. As our understanding of CNN through in class explanation, the size of the images used, the number of images used for training, the number of layers in the model, the number of epochs, and the amount of time taken for training.
2. Building and testing models is an extremely time-consuming process and requires many experiments with hyper-parameters.

## Next Steps

Immediate next steps based on the approach:

* Data Generation – Already finalized the data generation approach. It would require standardization of parameters and optimize it to get the input required for the model
* EDA – Identified the approach for doing EDA and ensuring the actions required to clean up the data before using for further processing
* Data Augmentation – Creating different possible images with the help of existing data, to overcome overfitting.
* Build and Train the model – Research on the model is completed. Next step is to build the model, utilize the segmented data and input data to train and test
* Improving accuracy – Fine-tune the model with CNN and NN to get the required accuracy
* Building algebraic algorithms – To take driving decisions.

## Appendix:

**Phase-wise roadmap:**

**Phase 1:**

* This scope is designed for the given timeframe.
* Model to predict stop/continue driving
* Indicate driving angle

As a stretch goal beyond phase 1 following aspects will be explored and considered for extending the scope subject to completion of each phase prior to it.

**Phase 2:**

This will be an addition to the scope if Phase 1is completed. Otherwise, Phase 1 will be considered as the project submission

* Decide the distance to apply the brake

**Phase 3:**

This will be an addition to the scope if Phase 2 is completed. Otherwise, Phase 2 will be considered as the project submission

* Scenarios to include vehicle coming from back and other sides