Ramakrishna Mission Vivekananda Educational & Research Institute

Belur Math, Howrah, West Bengal

School of Mathematical Sciences, Department of Computer Science

Assignment - 2 (Multi-Class Semantic Image Segmentation using COMMA-1K images)

M.Sc. Computer Science and Big Data Analytics Date: 13 March 2024

Course : **CS411: Applications of Computer Vision and Deep Learning**

Deadline: 5th-April-2024, 11:59 P.M.

Instructor: Jimut Bahan Pal Max marks: 140

Instructions: **Attempt all the questions**. Submit the code as **Name_ROLL.zip** and send it to the reply email on **jimutbahanpal@yahoo.com** AND **jpal.cs@gm.rkmvu.ac.in** as Carbon Copy (CC). You may also CC it to your other personal email address to check whether the email has gone to the address. **Submission after the deadline will fetch you -5 marks per day, so start early!**

1. In the Image Segmentation lecture, you explored the process of single-class segmentation for skin lesions. This assignment, however, introduces a captivating challenge centered around multi-class segmentation. Your task involves the segmentation of scenes captured by front-view cameras in cars, which are mounted through a mobile application. The objective is to delineate six distinct classes within the images by doing pixel-level classification using standard image segmentation techniques. Drawing inspiration from the methodology employed by COMMA.ai, where a phone-based system was ingeniously utilized to enhance existing cars, enabling autonomous driving by extracting key features. Notably, this approach allows the integration of advanced driving capabilities into a wide range of vehicles, eliminating the need for expensive alternatives such as Tesla. The assignment aims to emulate or design a segment of their image segmentation pipeline. Upon completion, you will possess the skills to proficiently conduct multi-class segmentation tasks. (140)

   We are using a small set of images, having 1000 images from the original COMMA_10K dataset hosted in github. You must refer to these two notebooks, i.e., **UNet_Architecture.ipynb** and **UNET_SEGMENTATION_Tutorial.ipynb** for getting a hook of the pipeline used there. The COMMA1K folder is divided into subfolders (**IMAGES, MASKS and MASKS_COLOR**) and files (**COLORMAP.txt**). **COLORMAP.txt** has the colormap, the **IMAGES** has the un-normalized images having intensity between 0 and 255, **MASKS** has class (from 1 to 5), our dataset has only five classes, as compared to the **COMMA-10K**, so, use 6 classes, just the first class will have no-relevance during training (please make suitable assumptions by looking at the data, and visiting the COMMA_10K github repository). The **MASKS_COLOR** folder has the colored masks for creating an overlay with the un-normalized intensity images as shown in Figure 1. There are a wide variety of images present in the dataset, images having fog, occlusion, haziness, different texture of roads, countryside roads, day and night images etc. It will be good to make robust segmentation algorithms using the available deep learning architectures, which are robust to all these changes, and perform well on any set of given images. The default colormap is (copied from the COMMA-10K repository):

   [0, 0, 0] := zeroth index, object of no relevance, just dummy placeholder for better processing, not present in the dataset

   [64, 32, 32] := road (all parts, anywhere nobody would look at you funny for driving)

   [255, 0, 0] := lane markings (don't include non lane markings like turn arrows and crosswalks)

   [128, 128, 96] := undrivable

   [0, 255, 102] := movable (vehicles and people/animals)

   [204, 0, 255] := my car (and anything inside it, including wires, mounts, etc. No reflections)

   [0, 204, 255] := movable in my car (people inside the car, imgs only)

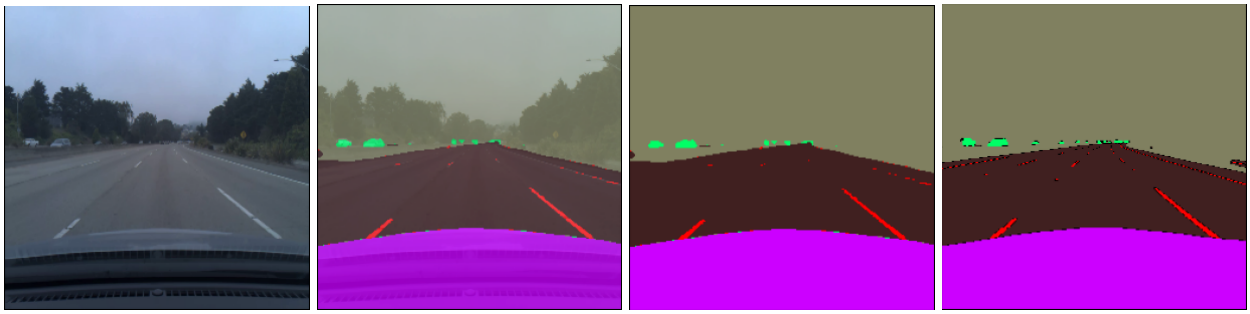
   Use the above colormap to visualize the output segmentation from the model.

Note: You must use the Google Colaboratory platform or any other GPU compute resource and submit the notebook for this task. You can tune the model's parameters to run it in decent-sized images without further hurting its performance. Everything in this assignment is left to the doer's choice to tune and play with for a better learning experience. Investing at least 10-15 hours for this assignment is required. The notebook should be self-explanatory, and you must demo the assignment in class using a loaded model that you will train and save over time. The focus should be on visualizing the qualitative results and easy demo within 5-10 minutes. Please practice the demo 1-2 times before the final one. You can refer to a friend and discuss the logic for this task, but you should code individually and not copy from another person's work.

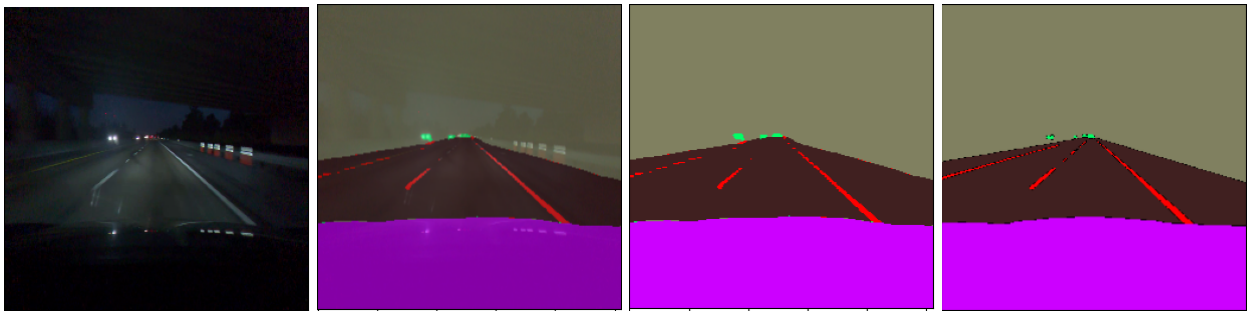
Accessing the data inside Google Colab: You can access the data here: `https://drive.google.com/file/d/1pvs-FFEiHtsTLnpDmct6p19JLvEpkmll/view`. Or you can use this command as it is in colab.

```
! pip install --upgrade --no-cache-dir gdown
! gdown 1pvs-FFEiHtsTLnpDmct6p19JLvEpkmll
! unzip -qq COMMA1K.zip
```

Here are some of the visualizations created from DeepLabV3+ architecture as shown in Figure 1. Use the same color styles for visualization for the given tasks.



(a) Image - Day (b) Overlay - Prediction (c) Prediction (d) Ground Truth



(e) Image - Night (f) Overlay - Prediction (g) Prediction (h) Ground Truth

Figure 1: Check how the lanes etc. are segmented properly even during night, this however was done using 10K images which achieved a Dice of 98%; but even using 700 images during training is expected to give good performance.

The current assignment is divided into the following parts:

(a) Please split the dataset into train, validation and testing by 70%-10%-20%, i.e., the first 700 image mask pairs will be used for training, the next 100 image mask samples will be used for validation and the last 20% of the image mask pairs will be used for testing. Please plot the first image mask samples from each of the splits and check they are consistent with the IDs as in the dataset. **(5)**

(b) Make proper amendments to the data loader and data generator to create image mask pairs which have one hot representation in terms of masks and normalized image tensor (between 0 and 1) as return type. Use standard NxN image size, like 256x256 or 128x128 but don't go

below 64x64. Find proper size which will run fast in google colaboratory and also complete full training using the available decent runtime for 100 epochs. This image will be passed on to a standard segmentation model, like U-Net and a proper mask will be generated at the other end of the model, which will be compared against the ground truth and loss will be calculated accordingly. **(15)**

(c) Please refer to the U-Net architecture as discussed in the class. Use the same model, except the concatenation across the symmetric encoder decoder layers, hence, now this model will perform like an Autoencoder. Plot the model and calculate its parameters using torch summary as done in the Notebooks. Use softmax as the final activation layer and the number of classes, i.e., 6 (here) will be the depth (thickness) of the final layer. Hence, the final layer now will output a one hot representation of the predicted mask, which needs to be mapped to class values during visualization. For example, the first element of the resulting activation from the feature map of dimension HxWxD will be [1,0,0,0,0,0] for class 1 (road), [0,1,0,0,0,0] for class 2 (lane markings), [0,0,1,0,0,0] for class 3 (u n-drivable) till [0,0,0,0,0,1] for class 6 (movable inside car). **(15)**

(d) Extend the train-validation-test pipeline to train-validate-test this new multi-class segmentation task, using proper loss function, like categorical cross entropy. Use another kind of loss function to see if the things are working better or not. Plot the train and validation loss for different learning rates, try at least 3 different learning rate, and plot the values for 100 (epochs at-least!). **(20)**

(e) Use multi-class dice coefficient, multiclass jaccard (IoU) metrics for recording the variation of training and validation. **(15)**

(f) Now, use a standard U-Net, with proper concatenation layers (just use the same model as used in the lecture for single class segmentation), and use the pipeline you created so far. **(5)**

(g) Find the final Dice Coefficient, Multi-Class Jaccard for the Autoencoder and U-Net and compare side by side the predictions that are obtained on the un-touched test dataset. You should map the final class to the colormap and visualize accordingly like Figure 1. **(20)**

(h) Report the test metrics across the untouched 200 images and display the first 10 images from each class. Use the saved model to load and dump the test folder's prediction by feeding the test images into the model. **(20)**

(i) You should dump all the predictions from the test dataset and compare it against the ground truth. Use proper mapping in the one-hot representation in both the GT and Predicted map to compute the actual metrics (class wise). You can represent the final score of the model on the test set by mean $\pm$ standard deviation of the performance for the two models. **(20)**

(j) Summarize what you learnt from the assignment, the challenges faced and any other important observation. **(5)**
