Instructions of Simulation and Testing of Model - Eshaan Agarwal

Index :

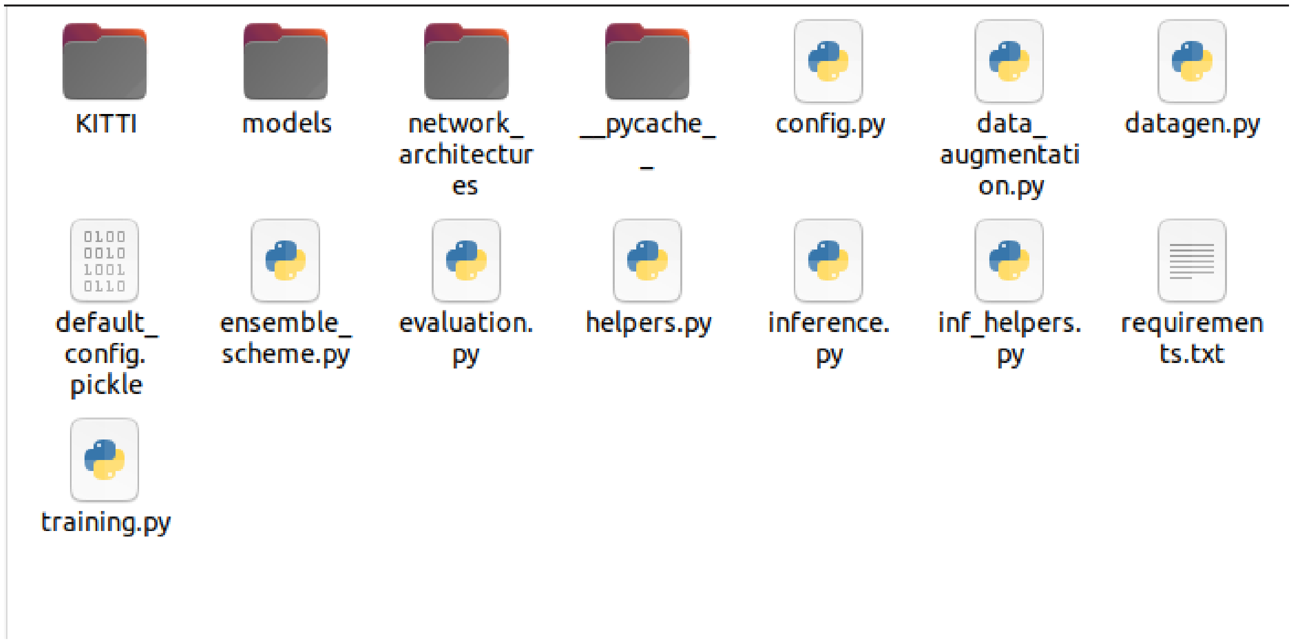
Code has been extensively documented and can be checked out to get a deeper understanding of the function and purpose.

File Structure:

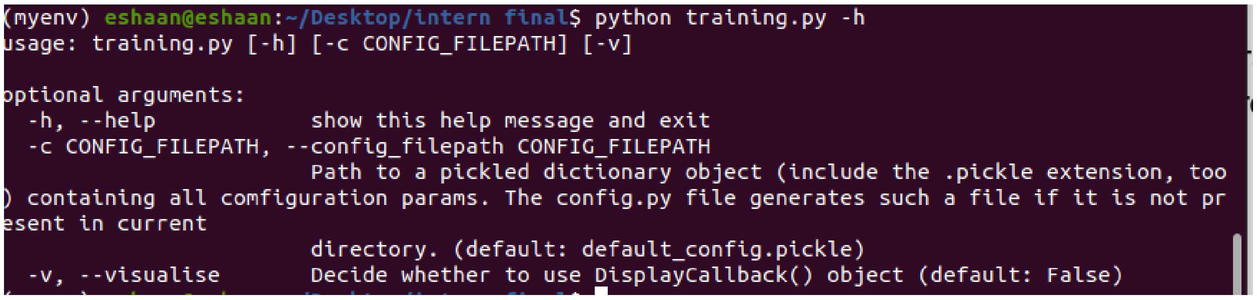
- **KITTI/** - directory contains all the visual data required for the training and prediction stages.
  - **data_road/** -
    - **enc_gt_images_1/** - All annotated labels including the augmented ones
    - **new_augs/-** folder includes all augmented version of the original KITTI road images
    - **uum_road/ -** Path to the testing images
    - **um_road/ -** Path to the training images

- **models/** - contain the pretrained models, Each saved model is stored with additional log information - config.txt, used_images.txt and loss_and_accuracy.pdf.

- **network_architectures/** scripts for the initialisation of the neural networks. Each saved model is stored with additional log information - config.txt, used_images.txt and loss_and_accuracy.pdf.

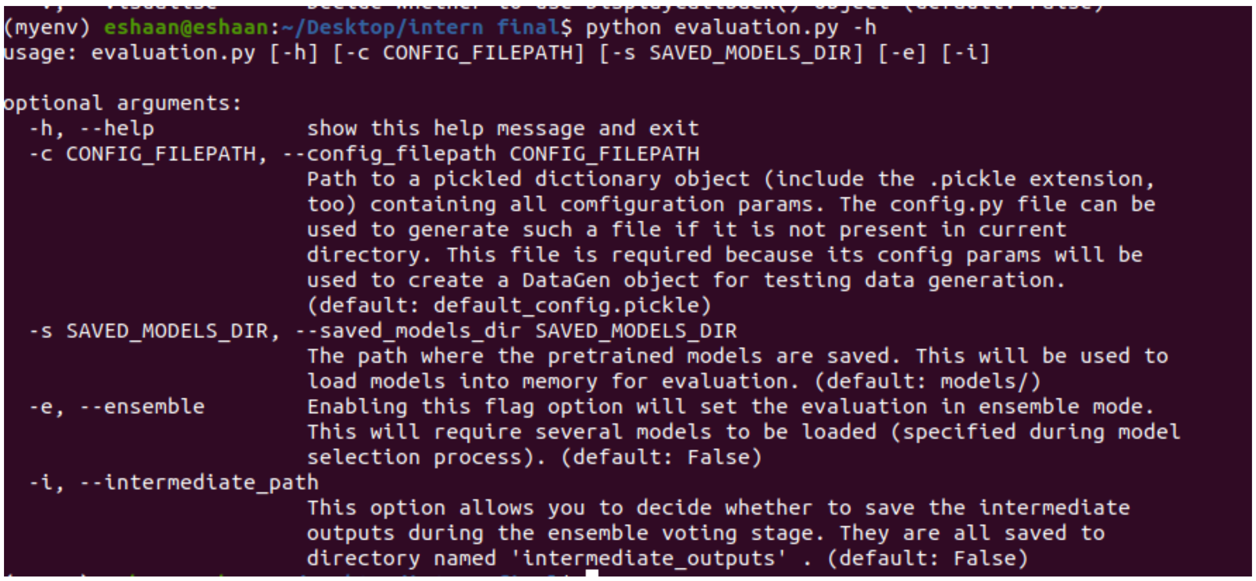


- **training.py -** This script is executed when a model needs to be trained.



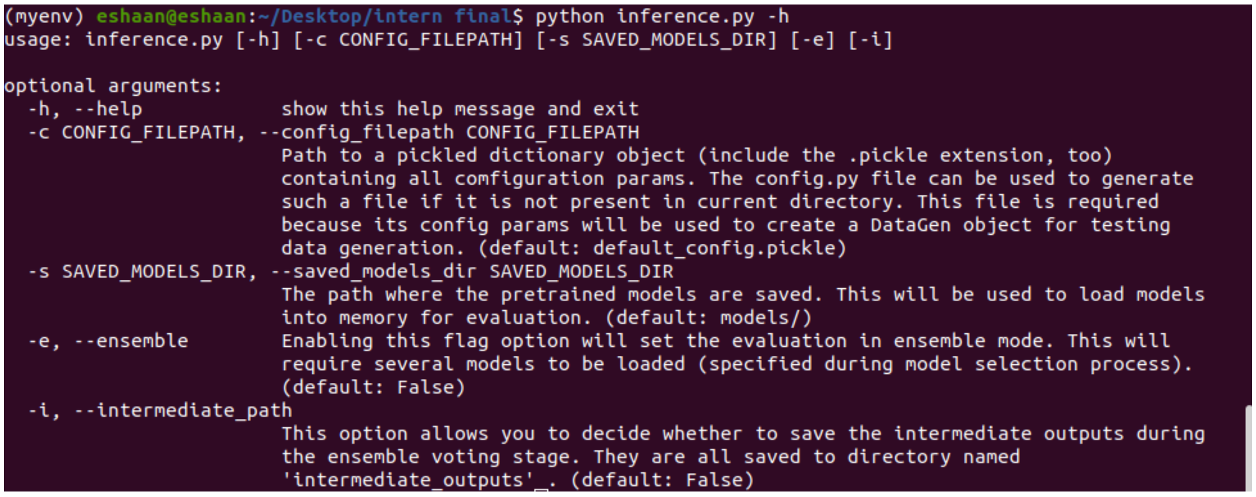
```
(myenv) eshaan@eshaan:~/Desktop/intern final$ python training.py -h
usage: training.py [-h] [-c CONFIG_FILEPATH] [-v]

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG_FILEPATH, --config_filepath CONFIG_FILEPATH
                        Path to a pickled dictionary object (include the .pickle extension, too
) containing all configuration params. The config.py file generates such a file if it is not pr
esent in current
                        directory. (default: default_config.pickle)
  -v, --visualise       Decide whether to use DisplayCallback() object (default: False)
```

- **evaluation.py -** This script is executed to test a trained model



```
(myenv) eshaan@eshaan:~/Desktop/intern final$ python evaluation.py -h
usage: evaluation.py [-h] [-c CONFIG_FILEPATH] [-s SAVED_MODELS_DIR] [-e] [-i]

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG_FILEPATH, --config_filepath CONFIG_FILEPATH
                        Path to a pickled dictionary object (include the .pickle extension,
                        too) containing all configuration params. The config.py file can be
                        used to generate such a file if it is not present in current
                        directory. This file is required because its config params will be
                        used to create a DataGen object for testing data generation.
                        (default: default_config.pickle)
  -s SAVED_MODELS_DIR, --saved_models_dir SAVED_MODELS_DIR
                        The path where the pretrained models are saved. This will be used to
                        load models into memory for evaluation. (default: models/)
  -e, --ensemble        Enabling this flag option will set the evaluation in ensemble mode.
                        This will require several models to be loaded (specified during model
                        selection process). (default: False)
  -i, --intermediate_path
                        This option allows you to decide whether to save the intermediate
                        outputs during the ensemble voting stage. They are all saved to
                        directory named 'intermediate_outputs' . (default: False)
```

- **inference.py -** The script is executed to test our results on random images without any ground_truth_mask

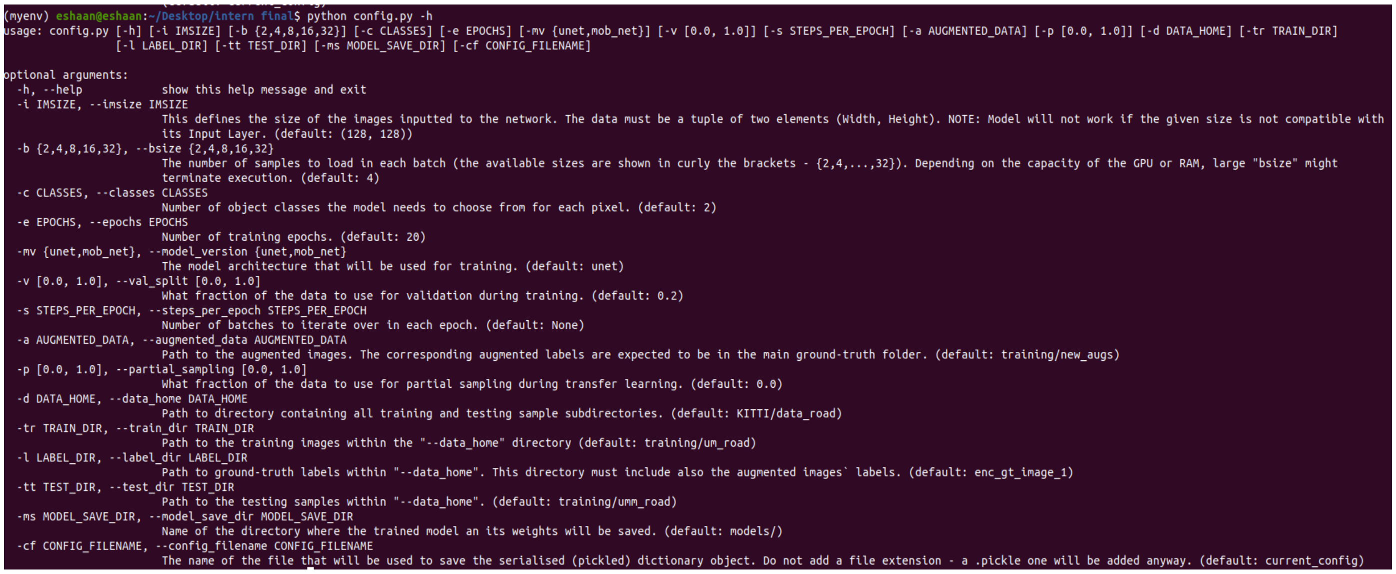


```
(myenv) eshaan@eshaan:~/Desktop/intern final$ python inference.py -h
usage: inference.py [-h] [-c CONFIG_FILEPATH] [-s SAVED_MODELS_DIR] [-e] [-i]

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG_FILEPATH, --config_filepath CONFIG_FILEPATH
                        Path to a pickled dictionary object (include the .pickle extension, too)
                        containing all configuration params. The config.py file can be used to generate
                        such a file if it is not present in current directory. This file is required
                        because its config params will be used to create a DataGen object for testing
                        data generation. (default: default_config.pickle)
  -s SAVED_MODELS_DIR, --saved_models_dir SAVED_MODELS_DIR
                        The path where the pretrained models are saved. This will be used to load models
                        into memory for evaluation. (default: models/)
  -e, --ensemble        Enabling this flag option will set the evaluation in ensemble mode. This will
                        require several models to be loaded (specified during model selection process).
                        (default: False)
  -i, --intermediate_path
                        This option allows you to decide whether to save the intermediate outputs during
                        the ensemble voting stage. They are all saved to directory named
                        'intermediate_outputs'_. (default: False)
```

- **helpers.py -** Most of the methods in this file are used for image visualization or during the evaluation stage

- **inf_helpers.py** - Most of the methods in this file are used for image visualization or during the inference stage

- **config.py** - Here, we create a dictionary object containing configuration parameters and export it to the disk as a .pickle file.



```
(myenv) eshaan@eshaan:~/Desktop/intern final$ python config.py -h
usage: config.py [-h] [-i IMSIZE] [-b {2,4,8,16,32}] [-c CLASSES] [-e EPOCHS] [-mv {unet,mob_net}] [-v [0.0, 1.0]] [-s STEPS_PER_EPOCH] [-a AUGMENTED_DATA] [-p [0.0, 1.0]] [-d DATA_HOME] [-tr TRAIN_DIR]
                 [-l LABEL_DIR] [-tt TEST_DIR] [-ms MODEL_SAVE_DIR] [-cf CONFIG_FILENAME]

optional arguments:
  -h, --help            show this help message and exit
  -i IMSIZE, --imsize IMSIZE
                        This defines the size of the images inputted to the network. The data must be a tuple of two elements (Width, Height). NOTE: Model will not work if the given size is not compatible with
                        its Input Layer. (default: (128, 128))
  -b {2,4,8,16,32}, --bsize {2,4,8,16,32}
                        The number of samples to load in each batch (the available sizes are shown in curly the brackets - {2,4,...,32}). Depending on the capacity of the GPU or RAM, large "bsize" might
                        terminate execution. (default: 4)
  -c CLASSES, --classes CLASSES
                        Number of object classes the model needs to choose from for each pixel. (default: 2)
  -e EPOCHS, --epochs EPOCHS
                        Number of training epochs. (default: 20)
  -mv {unet,mob_net}, --model_version {unet,mob_net}
                        The model architecture that will be used for training. (default: unet)
  -v [0.0, 1.0], --val_split [0.0, 1.0]
                        What fraction of the data to use for validation during training. (default: 0.2)
  -s STEPS_PER_EPOCH, --steps_per_epoch STEPS_PER_EPOCH
                        Number of batches to iterate over in each epoch. (default: None)
  -a AUGMENTED_DATA, --augmented_data AUGMENTED_DATA
                        Path to the augmented images. The corresponding augmented labels are expected to be in the main ground-truth folder. (default: training/new_augs)
  -p [0.0, 1.0], --partial_sampling [0.0, 1.0]
                        What fraction of the data to use for partial sampling during transfer learning. (default: 0.0)
  -d DATA_HOME, --data_home DATA_HOME
                        Path to directory containing all training and testing sample subdirectories. (default: KITTI/data_road)
  -tr TRAIN_DIR, --train_dir TRAIN_DIR
                        Path to the training images within the "--data_home" directory (default: training/um_road)
  -l LABEL_DIR, --label_dir LABEL_DIR
                        Path to ground-truth labels within "--data_home". This directory must include also the augmented images` labels. (default: enc_gt_image_1)
  -tt TEST_DIR, --test_dir TEST_DIR
                        Path to the testing samples within "--data_home". (default: training/umm_road)
  -ms MODEL_SAVE_DIR, --model_save_dir MODEL_SAVE_DIR
                        Name of the directory where the trained model an its weights will be saved. (default: models/)
  -cf CONFIG_FILENAME, --config_filename CONFIG_FILENAME
                        The name of the file that will be used to save the serialised (pickled) dictionary object. Do not add a file extension - a .pickle one will be added anyway. (default: current_config)
```

To train a model, we need a configuration object (Python dictionary) that contains parameters like batch size, number of epochs, data set paths and many others. Such a dictionary is created using the config.py Python script.

One dictionary is also provided with this distribution - default_config.pickle.
This file is a serialized (pickled) version of the configuration dictionary object that can be loaded into any script during runtime. It contains all default values of each parameter which are visible in the file.

Definitions for each of the optional arguments are also provided along with their default values.
It can be seen how the specified arguments, namely batch size, epochs, steps per epoch and configuration filename, are different from the default values. We generate a configuration dictionary that will train a U-net model using 128 × 128 images as input for the neural network. Additionally, the training process will continue for 2 epochs and during each of them, 2 batches containing 2 images will be presented to the model. The resulting model will be saved in the models/ directory within the current project folder.

optional arguments:

-h, --help show this help message and exit

-i IMSIZE, --imsize IMSIZE

This defines the size of the images inputted to the network. The data must be a tuple of two elements (Width, Height). NOTE: Model will not work if the given size is not compatible with its Input Layer. **(default: (128, 128))**

-b {2,4,8,16,32}, --bsize {2,4,8,16,32}

The number of samples to load in each batch (the available sizes are shown in curly the brackets - {2,4,...,32}). Depending on the capacity of the GPU or RAM, large "bsize" might terminate execution. **(default: 4)**

-c CLASSES, --classes CLASSES
Number of object classes the model needs to choose from for each pixel. **(default: 2)**

-e EPOCHS, --epochs EPOCHS
Number of training epochs. **(default: 20)**

-mv {unet,mob_net}, --model_version {unet,mob_net}
The model architecture that will be used for training. **(default: unet)**

-v [0.0, 1.0], --val_split [0.0, 1.0]
What fraction of the data to use for validation during training. **(default: 0.2)**

-s STEPS_PER_EPOCH, --steps_per_epoch STEPS_PER_EPOCH
Number of batches to iterate over in each epoch. **(default: None)**

-a AUGMENTED_DATA, --augmented_data AUGMENTED_DATA
Path to the augmented images. The corresponding augmented labels are expected to be in the main ground-truth folder. **(default: training/new_augs)**

-p [0.0, 1.0], --partial_sampling [0.0, 1.0]
What fraction of the data to use for partial sampling during transfer learning. **(default: 0.0)**

-d DATA_HOME, --data_home DATA_HOME
Path to directory containing all training and testing sample subdirectories. **(default: KITTI/data_road)**

-tr TRAIN_DIR, --train_dir TRAIN_DIR
Path to the training images within the "--data_home" directory **(default: training/um_road)**

-l LABEL_DIR, --label_dir LABEL_DIR
Path to ground-truth labels within "--data_home". This directory must also include the augmented images` labels. **(default: enc_gt_image_1)**

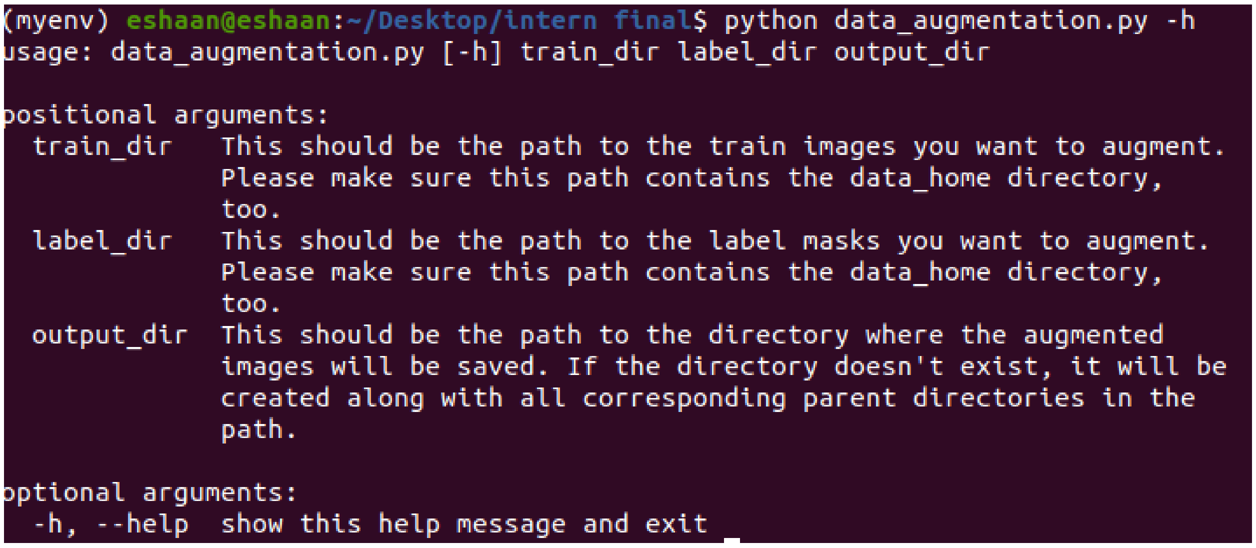
-tt TEST_DIR, --test_dir TEST_DIR
Path to the testing samples within "--data_home". **(default: training/umm_road)**

-ms MODEL_SAVE_DIR, --model_save_dir MODEL_SAVE_DIR
Name of the directory where the trained model an its weights will be saved. **(default: models/)**

-cf CONFIG_FILENAME, --config_filename CONFIG_FILENAME
The name of the file that will be used to save the serialised (pickled) dictionary object. Do not add a file extension - a .pickle one will be added anyway. **(default: current_config)**

- **datagen.py** - The DataGen class that implements the keras.utils.Sequence interface is defined here. DataGen is a subscriptable class to generate batches of training data in the form of Numpy arrays.

- **data_augmentation.py** - This script is responsible for data preprocessing



```
(myenv) eshaan@eshaan:~/Desktop/intern final$ python data_augmentation.py -h
usage: data_augmentation.py [-h] train_dir label_dir output_dir

positional arguments:
  train_dir    This should be the path to the train images you want to augment.
               Please make sure this path contains the data_home directory,
               too.
  label_dir    This should be the path to the label masks you want to augment.
               Please make sure this path contains the data_home directory,
               too.
  output_dir   This should be the path to the directory where the augmented
               images will be saved. If the directory doesn't exist, it will be
               created along with all corresponding parent directories in the
               path.

optional arguments:
  -h, --help   show this help message and exit
```

- **ensemble_scheme.py** - Only one method is implemented in this Python file. The 'apply_voting_to_ensemble_predictions(predictions, savepath)' method takes a list of predictions as input and an optional output directory path. It merges the predictions of multiple models by applying a hard majority voting ensemble scheme.

Instructions of Using the code

Set-Up Working Environment -

Requirements: The requirements.txt file contains the names of the required third-party Python libraries and their versions which are needed to run this project.

1. Setting up the virtual env:

Run the following codes in your terminal to set up the virtual env. CODE -

```
conda create -n myenv python=3.9
source ~/miniconda3/etc/profile.d/conda.sh
conda activate myenv
conda deactivate   # deactivate env.
```

Install dependencies from requirements.txt file CODE -

```
pip install -r requirements.txt
```

2. Manual Masking: Done via common Paints software by Windows (Paints 3D).
(Note : this needs to be done only if ground_truth mask of training images are not already present)

3. Setup Training Dataset Directory -
Put all the ground truth mask images into this folder - KITTI/data_road/enc_gt_image_1/
Put all the training images into following folder - KITTI/data_road/training/um_road
Put all the testing images into following folder - KITTI/data_road/training/umm_road

4. Data Augmentation - Necessary step to increase size of dataset so that model could be trained better

CODE -

```
    python data_augmentation.py KITTI/data_road/training/um_road
KITTI/data_road/enc_gt_image_1 KITTI/data_road/training/new_augs
```

train_dir - KITTI/data_road/training/um_road/
label_dir - KITTI/data_road/enc_gt_image_1/
Output_dir - KITTI/data_road/training/new_augs/

- After this The augmented ground-truth masks are located in the enc_gt_image_1/ directory within data_road and have the word 'new' in their names. The new_augs/ folder contains all raw images' augmented versions within data_road/training/.

- Copy Images from new_augs/ to um_road folder.

5. Parameter Configuration -
(if you have done step 3 and 4 properly and do not want to change training parameters or path of dataset of the model then you may skip this step)

Otherwise - Check description about parameter configuration given above and use command `python config.py -h` for further help in setting parameters and training directory)

Example command -

```
python config.py -b 2 -e 2 -s 2 -cf test_config
```

(Following command will change required parameters and save it in test_config.pickle which will further be used for training and testing.

Default config pickle used is default_config.pickle

6. Model Training -
Run the following command if default pickle used

```
python training.py -c default_config.pickle
```

Else

```
python training.py -c --config_file_path
```

(--config_file_path is path of the latest config pickle file generates in step 5)

7 Evaluation on Training Dataset

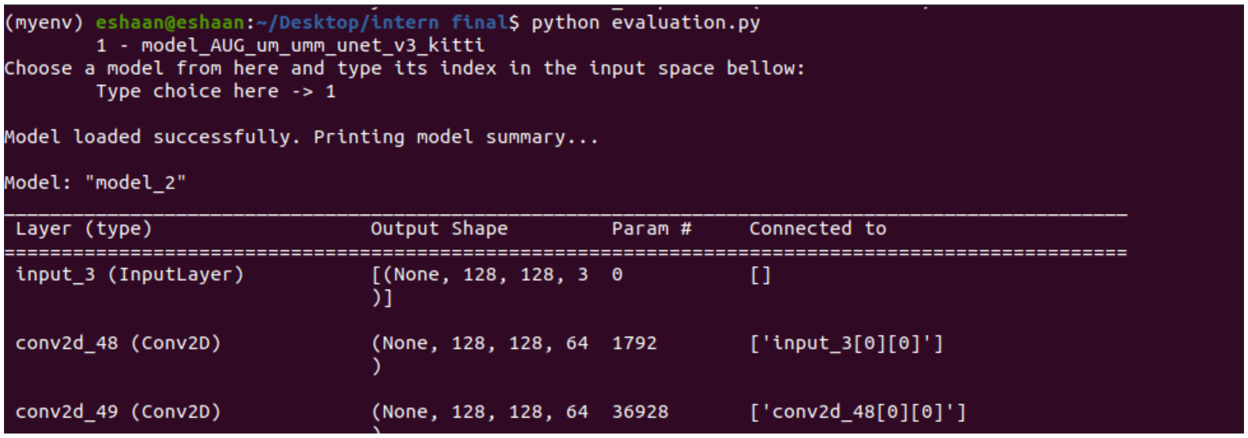
Run the following command

```
python evaluation.py
```

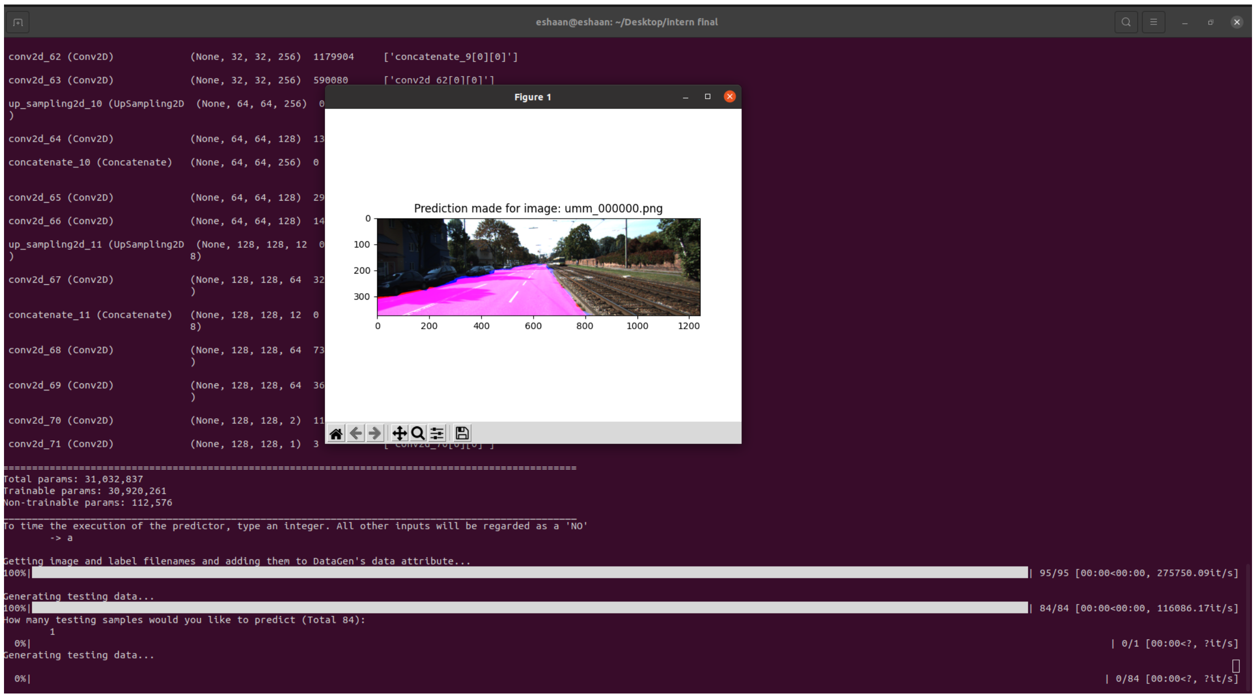
Or

```
python evaluation.py -c --config_file_path
```

(--config_file_path is path of the latest config pickle file generates in step 5)
(After running either of these commands a series of questions will be asked in the command line.
They would be about saving predicted images in particular folder and calculation time.
They can be easily answered by reading depending on the situation.)
Here are some screenshots of the process -

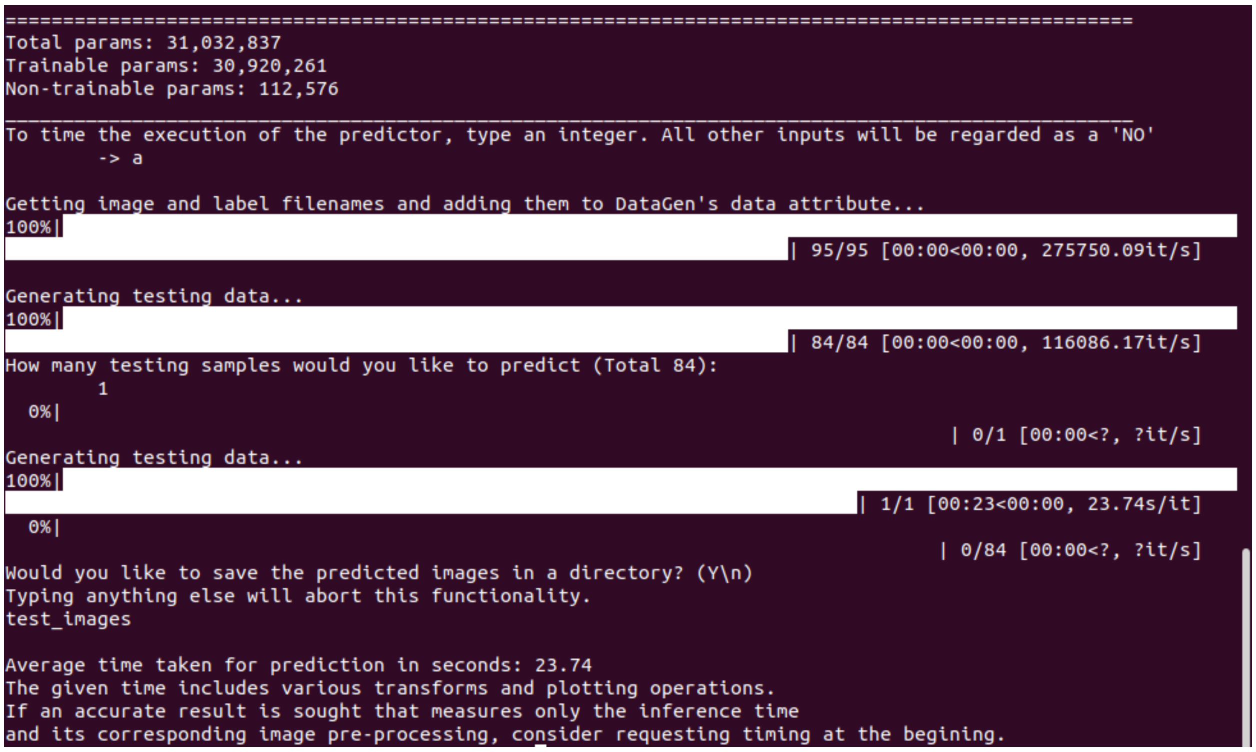


```
(myenv) eshaan@eshaan:~/Desktop/intern final$ python evaluation.py
        1 - model_AUG_um_umm_unet_v3_kitti
Choose a model from here and type its index in the input space bellow:
        Type choice here -> 1

Model loaded successfully. Printing model summary...

Model: "model_2"
___________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
===================================================================================================
 input_3 (InputLayer)           [(None, 128, 128, 3  0           []
                                )]

 conv2d_48 (Conv2D)             (None, 128, 128, 64  1792        ['input_3[0][0]']
                                )

 conv2d_49 (Conv2D)             (None, 128, 128, 64  36928       ['conv2d_48[0][0]']
```



eshaan@eshaan: ~/Desktop/intern final

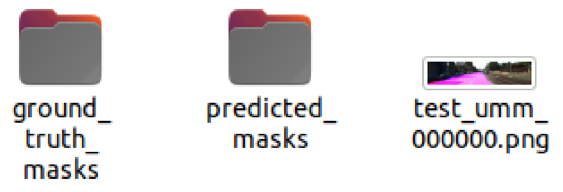
```
 conv2d_62 (Conv2D)             (None, 32, 32, 256)  1179904     ['concatenate_9[0][0]']

 conv2d_63 (Conv2D)             (None, 32, 32, 256)  590080      ['conv2d_62[0][0]']

 up_sampling2d_10 (UpSampling2D  (None, 64, 64, 256)  0
 )

 conv2d_64 (Conv2D)             (None, 64, 64, 128)  13

 concatenate_10 (Concatenate)   (None, 64, 64, 256)  0


 conv2d_65 (Conv2D)             (None, 64, 64, 128)  29

 conv2d_66 (Conv2D)             (None, 64, 64, 128)  14

 up_sampling2d_11 (UpSampling2D  (None, 128, 128, 12  0
 )                              8)

 conv2d_67 (Conv2D)             (None, 128, 128, 64  32
                                )

 concatenate_11 (Concatenate)   (None, 128, 128, 12  0
                                8)

 conv2d_68 (Conv2D)             (None, 128, 128, 64  73
                                )

 conv2d_69 (Conv2D)             (None, 128, 128, 64  36
                                )

 conv2d_70 (Conv2D)             (None, 128, 128, 2)  11

 conv2d_71 (Conv2D)             (None, 128, 128, 1)  3                ['conv2d_70[0][0]']

===================================================================================================
Total params: 31,032,837
Trainable params: 30,920,261
Non-trainable params: 112,576

___________________________________________________________________________________________________
To time the execution of the predictor, type an integer. All other inputs will be regarded as a 'NO'
        -> a

Getting image and label filenames and adding them to DataGen's data attribute...
100%|██████████████████████████████████████████████████| 95/95 [00:00<00:00, 275750.09it/s]

Generating testing data...
100%|██████████████████████████████████████████████████| 84/84 [00:00<00:00, 116086.17it/s]
How many testing samples would you like to predict (Total 84):
        1
  0%|                                                  | 0/1 [00:00<?, ?it/s]
Generating testing data...

  0%|                                                  | 0/84 [00:00<?, ?it/s]
```



```
==================================================================================================
Total params: 31,032,837
Trainable params: 30,920,261
Non-trainable params: 112,576
__________________________________________________________________________________________________
To time the execution of the predictor, type an integer. All other inputs will be regarded as a 'NO'
        -> a

Getting image and label filenames and adding them to DataGen's data attribute...
100%|
                                                                   | 95/95 [00:00<00:00, 275750.09it/s]

Generating testing data...
100%|
                                                                   | 84/84 [00:00<00:00, 116086.17it/s]
How many testing samples would you like to predict (Total 84):
        1
  0%|
                                                                                 | 0/1 [00:00<?, ?it/s]
Generating testing data...
100%|
                                                                         | 1/1 [00:23<00:00, 23.74s/it]
  0%|
                                                                                | 0/84 [00:00<?, ?it/s]
Would you like to save the predicted images in a directory? (Y\n)
Typing anything else will abort this functionality.
test_images

Average time taken for prediction in seconds: 23.74
The given time includes various transforms and plotting operations.
If an accurate result is sought that measures only the inference time
and its corresponding image pre-processing, consider requesting timing at the begining.
```

Predicted Images : In the final part of the output, the user is prompted whether they would like to save all the overlays and the associated predicted and true masks to a directory. As per your input predicted road img and their corresponding predicted mask img, ground truth image are saved in the folder.
(name of your folder where you save images)

- ground_truth_masks - Contains ground truth mask of the image which is tested
- predicted_masks - Contains predicted mask of the test image by the model.
- All final predicted road images by the model



8 Testing our Model on random image :
Run the following command

python inference.py

Or

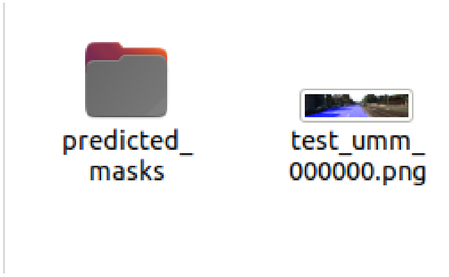
python inference.py -c *--config_file_path*

(--config_file_path is path of the latest config pickle file generates in step 5)

(After running either of these commands a series of questions will be asked in the command line. They would be about saving predicted images in particular folder and calculation time.
They can be easily answered by reading depending on the situation.Similar to step 7)

Predicted Images : In the final part of the output, the user is prompted whether they would like to save all the overlays and the associated predicted and true masks to a directory. As per your input predicted road img and their corresponding predicted mask img, ground truth image are saved in the folder.
(name of your folder where you save images)

- predicted_masks - Contains predicted mask of the test image by the model.
- All final predicted road images by the model



9.**Ensemble scheme - Optional - Only use if you have two or more model pretrained.**
Ensemble scheme is used so that models could be trained and several models are executed simultaneously and their outputs are fused together to derive the final prediction.
Such a design can reduce incorrect predictions of individual models and produce more precise segmentation masks.

This method merges the predictions of multiple models by applying majority hard voting.
For detail information - checkout ensemble_scheme.py

To use ensemble mode - just add ‘-e’ flag in Step 7 and Step 8 commands.
Example -

```
python inference.py -e
Or
python evaluation.py -e
```