Frameworks Used: PyTorch, Sci-kit learn, Segmentation Models.

Description:

To detect lanes on a given road. It's a preliminary step in any autonomous navigation system. So precise localisation of maps is a necessity to ensure the proper working of any navigation system.

Given are the set of images which has their own respective lane marking labels. There are only 501 images which we need to utilise in framing our solution.

Ideas:

- We can use any standard Computer Vision techniques which extracts patterns based on visual cues, but they are difficult to generalise in many kinds of situations where the images are not clear due to light exposure, weather changes etc..
- Also, standard CV methods can be complicated in detecting polygonal lanes which happen to be appearing at lane changes, crossings etc...
- Or we can use some Neural Network based approach where we can train our model to make more generalised predictions. Since we can augment our existing data by changing the filter, perspective, exposure etc.. we can expand the diversity of data to make our model more generalizable.
- Pretrained models such as LaneNet can be used but many API based solutions also exist for this use case. So let's build a custom U-Net based model which segments the road lanes in a given image.

Implemented Solution:

1. Since we've been given only 501 images, we need to use a model which is computationally efficient and light in memory.
2. We can use a U-Net based model to model this problem as a binary image segmentation problem. Also, **U-Net** based models are memory efficient and can deliver results ensembled from varied sampling blocks.
3. **EfficientNet-B3** is used as the encoder which has been pretrained on the imagenet DB and has parameters ~10M. It is light in memory and can be quite fast while using a streaming video which in turn gives more fps.
4. **BCEWithLogitsLoss** has been used as the loss function since we are trying to segment the lanes on the road, so only pixels with lanes need to be identified.
5. We can use **Pixel wise accuracy** as a metric.
6. Since the mask proportion over image is close to normal here we can use accuracy as the metric, when we have a skew in the mask proportion over images we have to use metrics which involves Precision and Recall which is not the case we are dealing with.
7. We can also use **Dice Loss** and **IoU** metric but the number of labels present in the data is unavailable.
8. Still, Pixel wise accuracy is much preferred since we can filter out precise lane markings without biases which commonly occur when we use IoU as a validation metric.
9. **Cross-Validation:** The validation split has been made based on the percentage of mask (lane) present in the images. In order to maintain an equivalent proportion of masked images, stratified split has been used. 15% of images have been reserved for validation purposes while the rest have been used for training purposes.

10. **Scaling and Data Augmentation:** Given images are having a resolution of 1280x720. In order to use a full scale image of this size we need more compute memory, but due to lack of compute I have to scale the images down to 512x512. Upon the resized images various data augmentations have been applied such as Flips, Rotations, RandomNoise, Perspectives so on.. Which almost adds 20 more images for a given image which adds up more amount of data to our model. So our augmentation adds more data close to 450*20 = **~9000** amount of samples.
11. When trained for 10 epochs the model has achieved a pixel-wise accuracy of **91.9%** which is a decent score. Given more data and compute we can get a better performing model than what we achieved now.

Further Development:

1. The predicted masks are in the form of probabilities, when we clip those values to 1 which are above 0.5 we can get precise predicted lane markings.
2. In order to refine the output more we can apply CV methods such as noise removal and line transformations as a post processing step.
3. Once we train a U-Net which performs significantly well we can distill the knowledge representation down to a smaller neural network on the basis of student-teacher contrastive learning based training. These kinds of distilled models are easy to deploy since they will be 10x lighter than their parent models and perform closer to the bigger model (teacher model).