Dataset Description:

This dataset contains images for Drivable Area segmentation and Lane detection. All the images are generated using Stable diffusion in Google Colaboratory. This dataset is around 90 Megabytes. The project we are working on has two label outputs for each sample. And these outputs are overlayed on the original image.

Link to our project's repository. Click here

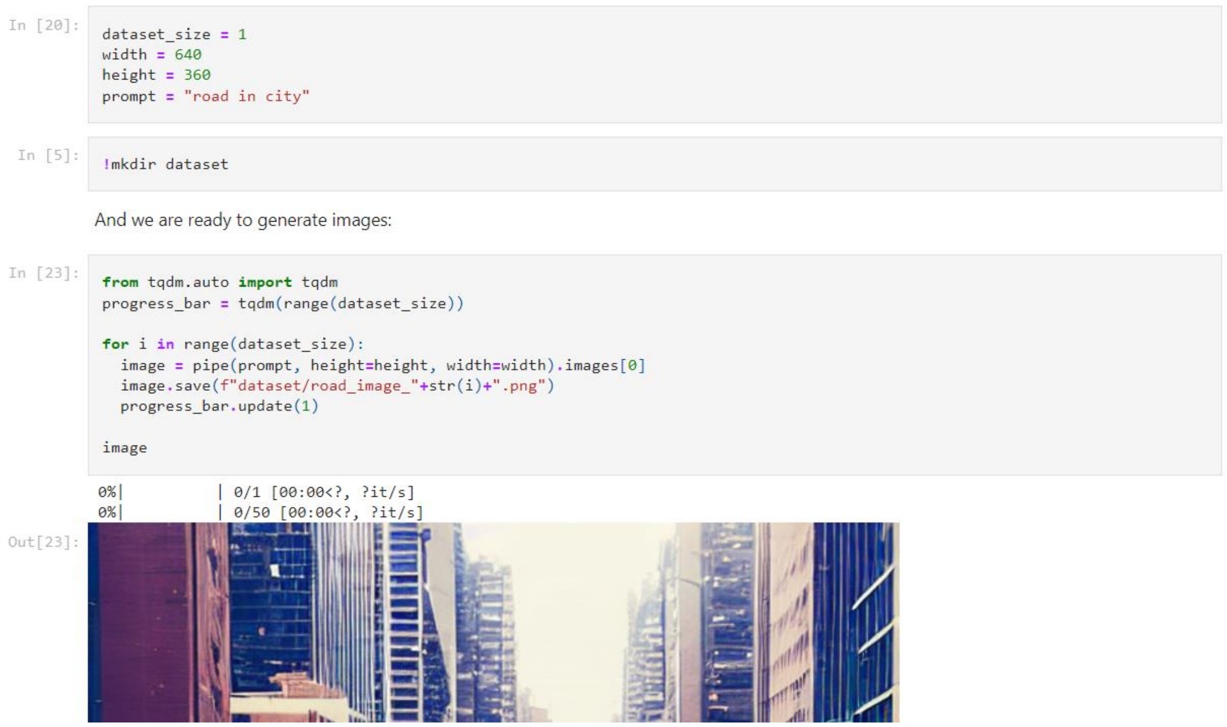
The dataset that we annotated is pushed into GitHub with Git LFS and OneDrive.

(GitHub) Link to the dataset. Click here

(OneDrive) Link to the dataset. Click here

Link to the Stable diffusion ipynb file. Click here

In the given below image. We changed dataset_size to 200 and height and width are provided to match the project specifications. And also the prompt is provided.



```
In [20]:
dataset_size = 1
width = 640
height = 360
prompt = "road in city"
```

```
In [5]:
!mkdir dataset
```

And we are ready to generate images:

```
In [23]:
from tqdm.auto import tqdm
progress_bar = tqdm(range(dataset_size))

for i in range(dataset_size):
  image = pipe(prompt, height=height, width=width).images[0]
  image.save(f"dataset/road_image_"+str(i)+".png")
  progress_bar.update(1)

image
```

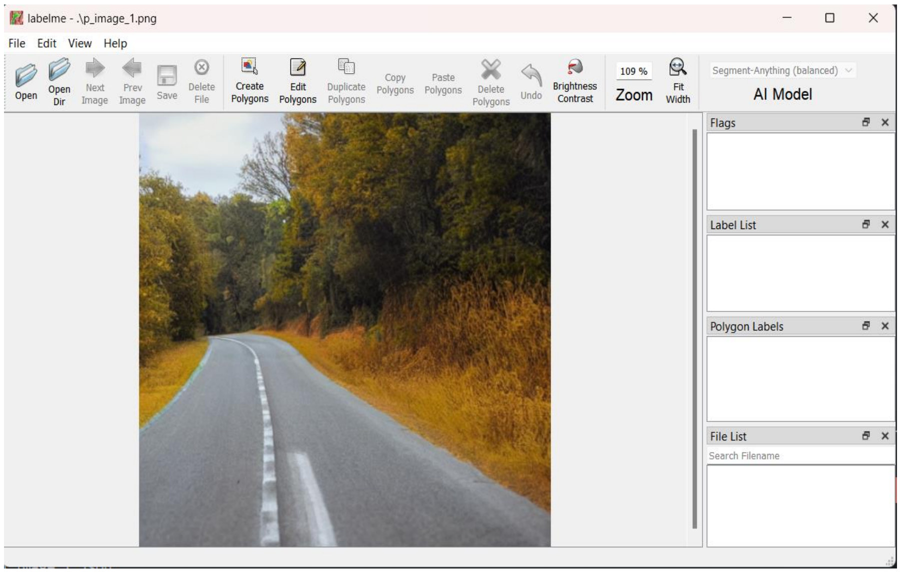
```
0%|          | 0/1 [00:00<?, ?it/s]
0%|          | 0/50 [00:00<?, ?it/s]
```

Out[23]:

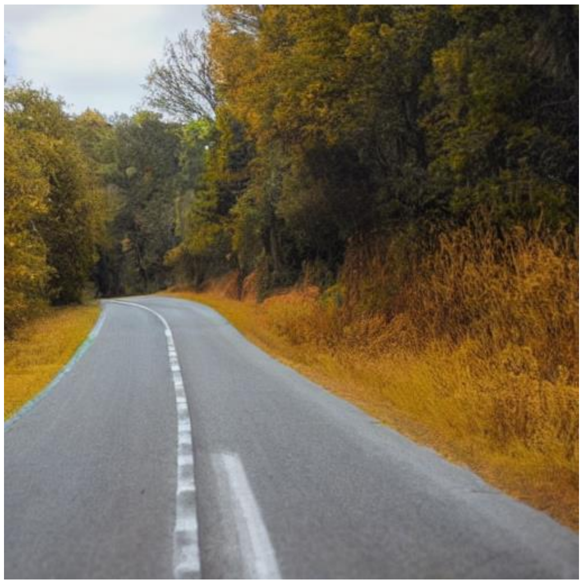
Annotation:

The images are annotated using labelme tool. Which is an opensource tool used to annotate image data. Each image is annotated twice one is for drivable area segmentation and another is for lane detection.

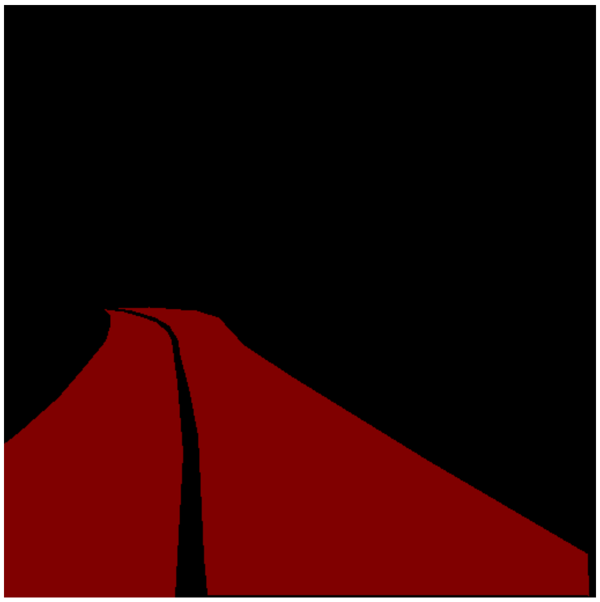
Label me tool:



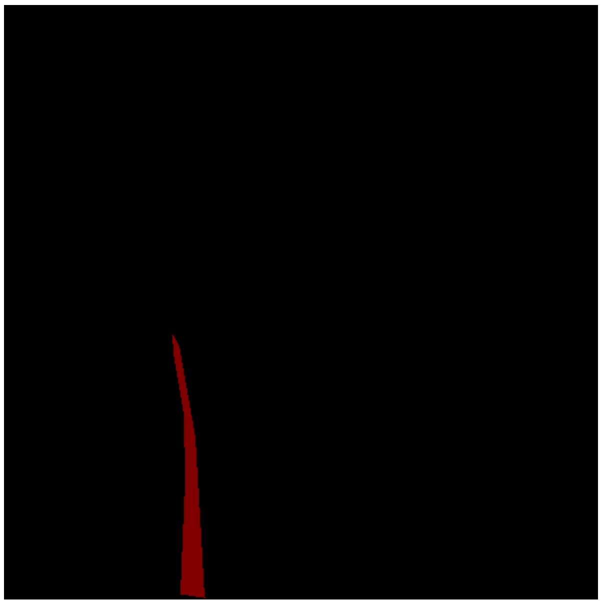
Original Image:



Annotation for Drivable area segmentation:



Annotation for Lane detection:



Partitioning:

The dataset is structured into three distinct partitions: Train, Test, and Validation. The Train split comprises 80% of the dataset, containing both the input images and their corresponding labels. Meanwhile, the Test and Validation splits each contain 10% of the data, with a similar structure, consisting of image data and label information.

Within each of these splits, there are three folders:

- Images: This folder contains the original images, serving as the raw input data for the task at hand.
- Segments: Here, you can access the labels specifically designed for Drivable Area Segmentation, crucial for understanding road structure and drivable areas.
- Lane: This folder contains labels dedicated to Lane Detection, assisting in identifying and marking lanes on the road.

Transformation:

1. Random Perspective Transformation:
   This transformation simulates changes in the camera's perspective, including rotation, scaling, shearing, and translation. It is applied with random parameters:
    - **degrees:** Random rotation in the range of -10 to 10 degrees.
    - **translate:** Random translation in the range of -0.1 to 0.1 times the image dimensions.
    - **scale:** Random scaling in the range of 0.9 to 1.1 times the original size.
    - **shear:** Random shearing in the range of -10 to 10 degrees.
    - **perspective:** A slight random perspective distortion.
2. HSV Color Augmentation:
    - This changes the hue, saturation, and value of the image.
    - Random gains for hue, saturation, and value are applied.
    - The hue is modified by rotating the color wheel.
    - The saturation and value are adjusted by multiplying with random factors.
    - This helps to make the model invariant to changes in lighting and color variations.
3. Image Resizing:
    - If the Images are not in the specified size, the images are resized to a fixed size (640x360) using cv2.resize.
4. Label Preprocessing:
    - The labels (segmentation masks) are thresholded to create binary masks. This means that pixel values are set to 0 or 255 based on a threshold (usually 1 in this case).
    - The binary masks are also inverted to create a binary mask for the background.

- These binary masks are converted to PyTorch tensors for use in training the semantic segmentation model.

Loss:

Two loss functions are used here one is focal loss and another is tversky loss.

Focal Loss:

$$Loss_{focal} = -\frac{1}{N}\sum_{c=0}^{C-1}\sum_{i=1}^{N} p_i(c)(1-\hat{p}_i(c))^{\gamma} log(\hat{p}_i(c))$$

Tversky Loss:

$$Loss_{tversky} = \sum_{c=0}^{C}(1 - \frac{TP(c)}{TP(c) - \alpha FN(c) - \beta FP(c)})$$

Total Loss:

$$Loss_{total} = Loss_{focal} + Loss_{tversky}$$

Optimization:

In this setup, an Adam optimizer with a dynamically decreasing learning rate is employed. This adaptive learning rate is regulated using a Polynomial Learning Rate Scheduler, which gradually reduces the learning rate as the training progresses.