Lane Detection Using Deep Learning

Intro to the problem:

Lane detection is a critical component of modern transportation systems, playing a significant role in enhancing road safety and enabling intelligent navigation. Accurate and robust lane detection systems provide essential information about lane boundaries on the road, contributing to various applications such as autonomous driving, advanced driver-assistance systems (ADAS), and computer vision tasks.

The primary motivation behind this project is to develop an effective lane detection system using deep learning techniques, specifically employing the U-Net model. Accurate lane detection is of utmost importance for enhancing road safety by preventing lane departure incidents, which are a leading cause of accidents. Lane detection systems can significantly reduce the risk of collisions and provide valuable assistance in maintaining safe driving behaviour.

Literature Review:

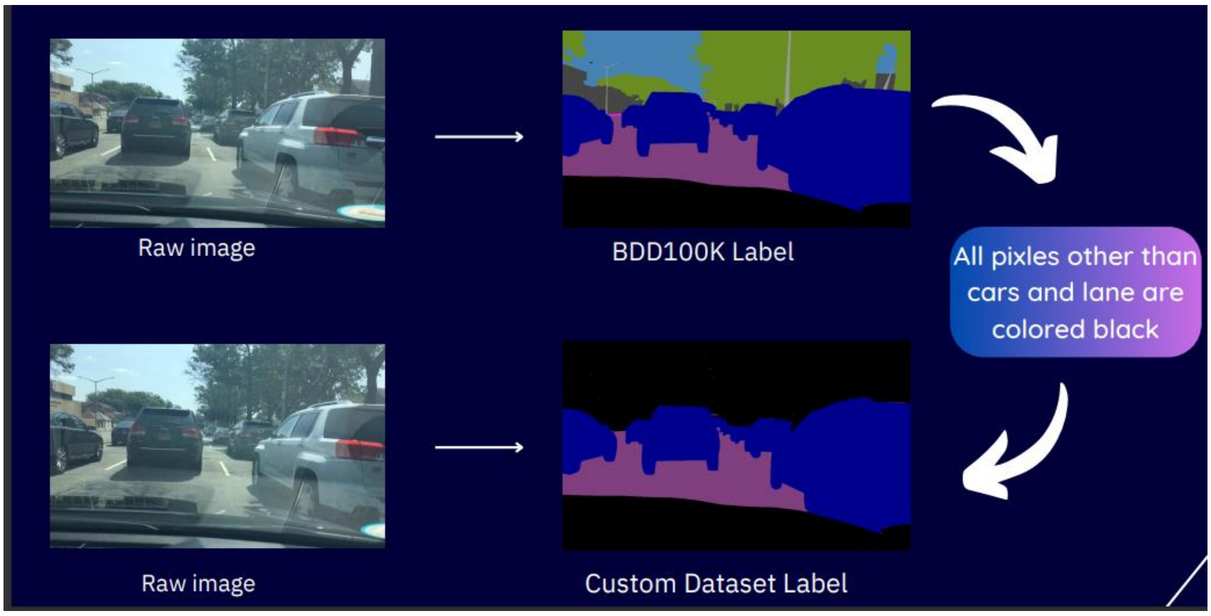
Traditional methods for lane detection have primarily relied on techniques such as Hough Transform, edge detection, or colour-based segmentation. These methods, although widely used, often struggle with robustness and accuracy in handling challenging scenarios, including varying lighting conditions, occlusions, and complex road geometries. Additionally, they heavily rely on handcrafted features, limiting their adaptability to different road environments.

With the advent of deep learning, there has been a shift towards data-driven approaches for lane detection, leveraging the capabilities of convolutional neural networks (CNNs) to automatically learn discriminative lane features. CNN-based models have demonstrated superior performance in various computer vision tasks, including semantic segmentation, which aligns well with the lane detection problem.

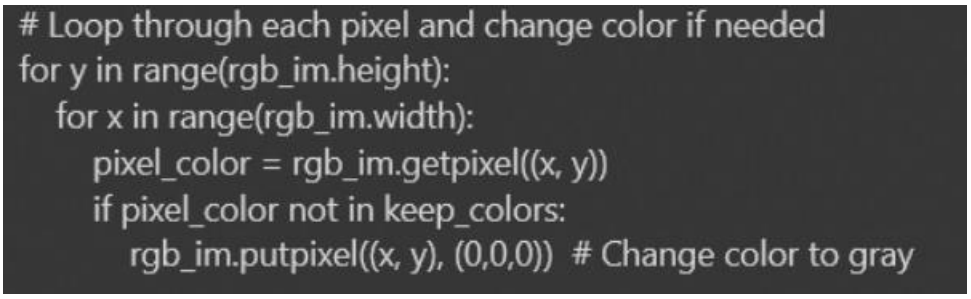
One prominent architecture that has gained attention for lane detection is the U-Net model. The U-Net model is characterized by its U-shaped architecture with skip connections, enabling it to capture both local details and global contextual information. It has shown remarkable success in semantic segmentation tasks by effectively delineating object boundaries.

Methodology:

Dataset Acquisition and Pre-processing:



We have created a custom labelled dataset using the BDD100K as our base dataset. The labels are modified in such way to make the pixels other than that of obstacles and lane as part of background rather than as separate instances.



```
# Loop through each pixel and change color if needed
for y in range(rgb_im.height):
    for x in range(rgb_im.width):
        pixel_color = rgb_im.getpixel((x, y))
        if pixel_color not in keep_colors:
            rgb_im.putpixel((x, y), (0,0,0))  # Change color to gray
```

We also ensured the dataset covers a diverse range of road conditions, lighting conditions, and lane geometries.

Pre-processing of the dataset by resizing the images to 128X128, normalizing pixel values, and augmenting the data to increase variability and improve model generalization.

Model Architecture:

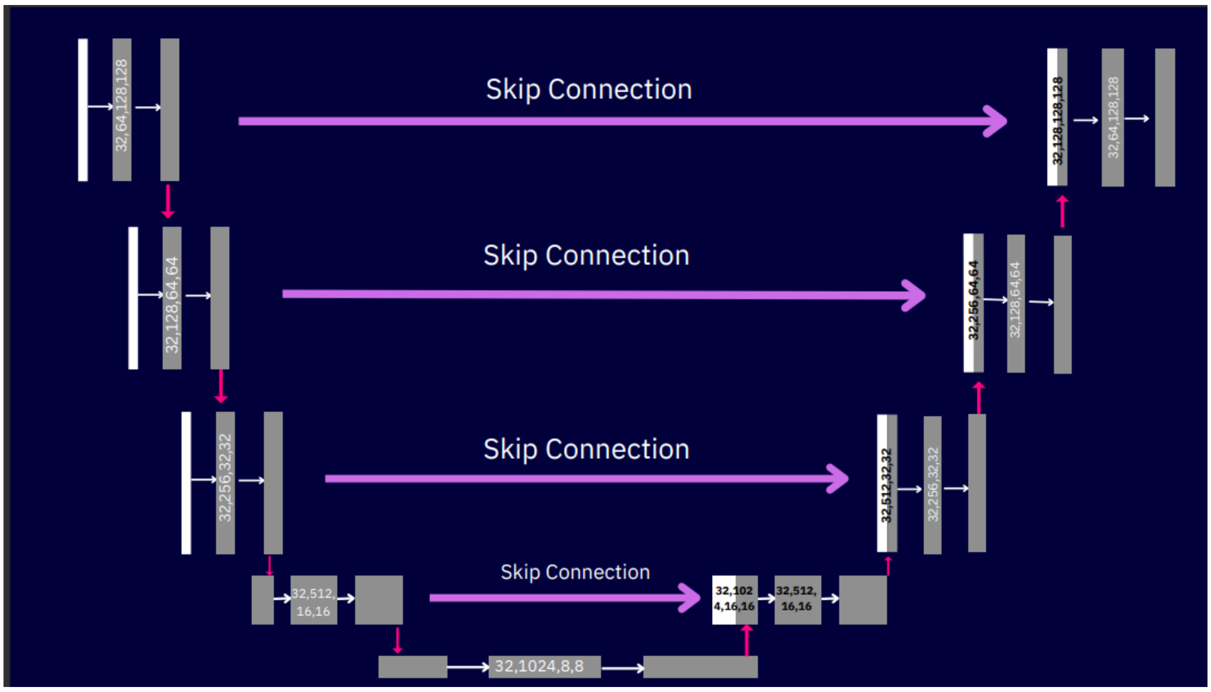
U-Net Model:

The U-Net model is a popular architecture in the field of deep learning, particularly for semantic segmentation tasks. It was introduced by Ranneberger et al. in 2015 and has since been widely adopted for various image segmentation applications, including lane detection.

The key characteristic of the U-Net model is its U-shaped architecture, which consists of an encoder path and a decoder path. The encoder path gradually reduces the spatial resolution of the input image while capturing high-level features through a series of convolutional and pooling layers. This encoding process effectively extracts hierarchical representations from the input image.

The decoder path, on the other hand, is responsible for up sampling the low-resolution feature maps obtained from the encoder. It gradually recovers the spatial information lost during the encoding process through a series of up sampling and concatenation operations. By utilizing skip connections, which connect corresponding feature maps from the encoder to the decoder, the U-Net model enables the fusion of both local details and global context information. This unique characteristic makes the U-Net model particularly effective in capturing fine-grained details and preserving spatial relationships.

In the context of lane detection, the U-Net model has shown remarkable performance in accurately segmenting lane markings from road scenes. By training the model on annotated lane datasets, it learns to associate pixels belonging to the lane markings with the appropriate class or label. This allows the U-Net model to predict pixel-level segmentation masks, effectively delineating the lane boundaries in an image or video frame.



The Layers dimensions are explained below

We have Input as 128X128 Image with batch size as 32

Input: N x 3 x H x W where N is the batch size, H and W are the height and width of the input image respectively.

First DoubleConv: N x 64 x H x W

Downsample: N x 64 x H/2 x W/2

Second DoubleConv: N x 128 x H/2 x W/2

Downsample: N x 128 x H/4 x W/4

Third DoubleConv: N x 256 x H/4 x W/4

Downsample: N x 256 x H/8 x W/8

Fourth DoubleConv: N x 512 x H/8 x W/8

Downsample: N x 512 x H/16 x W/16

Bottleneck DoubleConv: N x 1024 x H/16 x W/16

Upsample: N x 512 x H/8 x W/8

Concatenate with skip-connection from fourth DoubleConv: N x 1024 x H/8 x W/8

Fifth DoubleConv: N x 512 x H/8 x W/8

Upsample: N x 256 x H/4 x W/4

Concatenate with skip-connection from third DoubleConv: N x 512 x H/4 x W/4

Sixth DoubleConv: N x 256 x H/4 x W/4

Upsample: N x 128 x H/2 x W/2

Concatenate with skip-connection from second DoubleConv: N x 256 x H/2 x W/2

Seventh DoubleConv: N x 128 x H/2 x W/2

Upsample: N x 64 x H x W

Concatenate with skip-connection from first DoubleConv: N x 128 x H x W

Eighth DoubleConv: N x 64 x H x W

Final Convolution: N x 3 x H x W

The size of the feature maps decreases by half after each down sample operation and increases by a factor of 2 after each up sample operation. The number of channels increases with each stage, starting from 64 and doubling at each stage until reaching 512 at the bottleneck layer.

Loss Functions:

Loss Function used to train the model is a combination of Mean Squared Error Loss and Cross Entropy Loss. By combining these two loss functions, we aim to leverage their respective benefits and improve the overall performance of our model.

The MSE loss function is commonly used for regression tasks and measures the average squared difference between the predicted and ground truth values. In the context of lane detection, MSE loss can be applied to the regression-like task of estimating the position or parameters of the lane markings, such as the coordinates of the lane boundary points or the curvature of the lanes. By minimizing the MSE loss, we encourage the model to accurately estimate the lane properties, leading to precise lane detection.

On the other hand, the Cross Entropy loss function is widely used for classification tasks and is particularly suitable for semantic segmentation. In lane detection, semantic segmentation involves assigning a class or label (e.g., lane or background) to each pixel in an image or video frame. By applying the Cross Entropy loss, we encourage the model to accurately classify each pixel into the correct lane or background category, promoting accurate segmentation of the lane markings.

Combining the MSE and Cross Entropy losses allows us to address different aspects of the lane detection problem. The MSE loss focuses on precise estimation of lane parameters, while the Cross Entropy loss emphasizes accurate classification of pixels into lane and background categories.

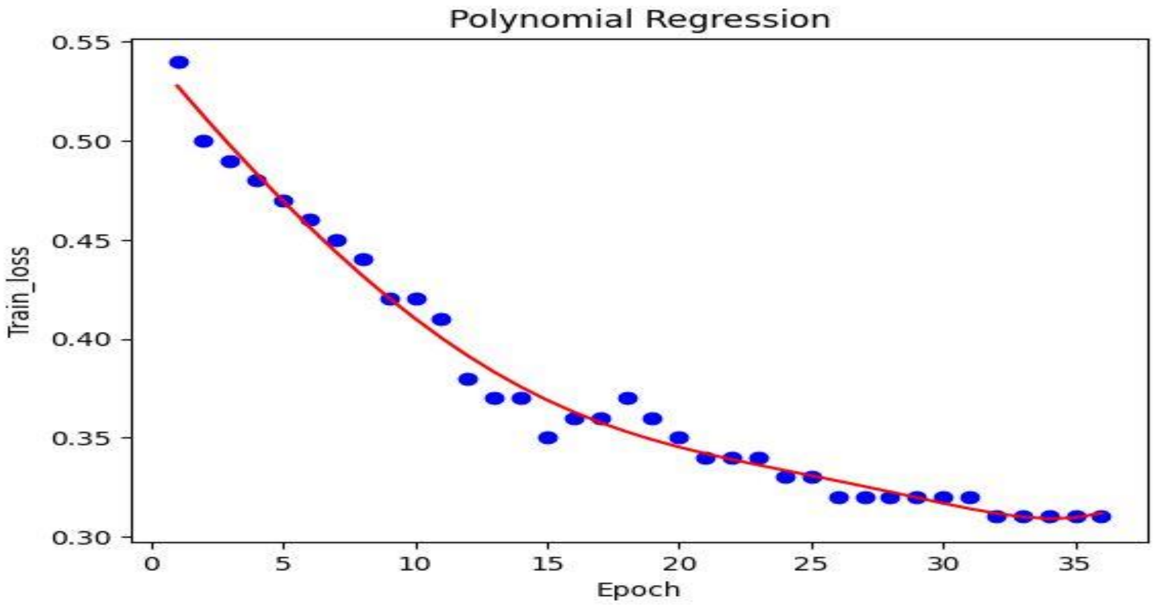
By employing the combined loss function, we aim to enhance the performance of our lane detection model. The MSE loss encourages accurate estimation of lane properties, while the Cross Entropy loss promotes accurate pixel-level classification.

The combined loss function is typically computed as a weighted sum of the individual loss functions, where the weights can be adjusted to balance the contribution of each loss component. Here the weights are 0.1 for MSE and 1 for Cross Entropy.

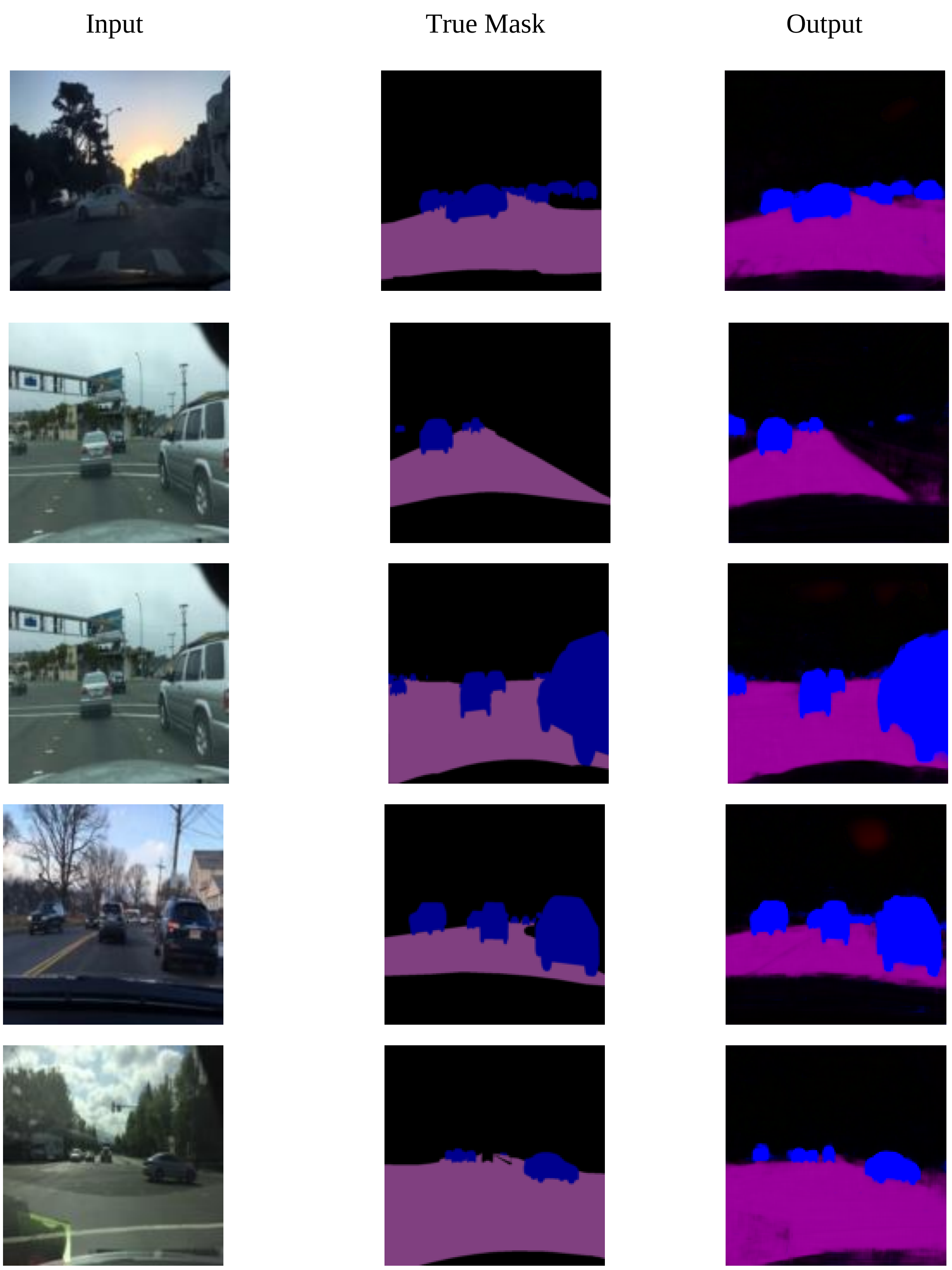
Training The Model:

We used a dataset of 7100 images of size 128x128 pixels for training our lane detection model. The dataset was split into 7000 training and 100 validation sets.

We used U-Net architecture with 4 encoding and decoding blocks with 64, 128, 256 and 512 filters, respectively. We trained the model for 100 epochs with a batch size of 32, using the MSE and cross-entropy loss function. We also applied random horizontal flips and rotations to the training images as data augmentation techniques.



Result:



The obtained result are very satisfactory as you can see. The Model could be improved if it can distinguish the pixels of sky, buildings, streetlights, grass etc. as different instances rather than all as part of background. But it will require more computation power.

Still the model performed very well give the less computation power we have.

References:

Blog on lane detection using U-Net
https://medium.com/@diazoangga/implementing-lane-detection-in-carla-using-lanenet-330d8fd8720c

Tutorial on U-Net
https://pyimagesearch.com/2022/02/21/u-net-image-segmentation-in-keras/

Road detection using convolutional neural networks-
(PDF) Road detection using convolutional neural networks (researchgate.net)

Real-time monocular image-based path detection
(PDF) Real-time monocular image-based path detection (researchgate.net)

Road Detection and Segmentation from Aerial Images using a CNN based System.

Road Detection and Segmentation from Aerial Images Using a CNN Based System | IEEE Conference Publication | IEEE Xplore

CNN based lane detection with instance segmentation in edge-cloud computing
CNN based lane detection with instance segmentation in edge-cloud computing | Journal of Cloud Computing | Full Text (springeropen.com)

Dataset: 1805.04687.pdf (arxiv.org)