APS360 Final Report
Lane Detection Using Deep Learning

Mustafa R. A. Khan
Student# 1006717037
mr.khan@mail.utoronto.ca

Phil Cuvin
Student# 1005130289
philippe.cuvin@mail.utoronto.ca

Kelvin Wei Hai Cui
Student# 1005337324
kelvin.cui@mail.utoronto.ca

1 Introduction

The development of a safe and efficient autonomous vehicle comprises one of the most significant and highly-researched engineering challenges of today. Self-driving cars promise the improvement of road safety through the significant reduction in traffic accidents [NHT (2021)], as more than 90% of car accidents in the US are caused by human error [NHT (2021)]. Reduced congestion on roads, emissions, pollution, and energy consumption, improved mobility, and greater convenience present additional reasons to motivate the development of autonomous road vehicles. [Nrm (2018)]. Autonomous driving systems consist of various modules such as lane keeping assistance, lane centering, lane departure warning, adaptive cruise control and traffic jam assist Zhang et al. (2021). The most critical task for these technologies is to detect the lane markings on the road to keep the vehicle in control relative to the road and evaluate the movement of other vehicles surrounding the car to prevent collision [Oğuz et al. (2022)]. The speed of the detection and accuracy of results required for a successful system, as well as the demonstrated increased detection and evaluation accuracy of deep learning lane detection systems versus classical approaches, make this problem uniquely suited for a deep learning solution [Hanuman & Kumar (2021)].

The goal of this project is to develop a deep learning model which can accurately detect road lane lines under various real world conditions. The central inspiration and motivation of the selection of this project comes from the interest of participating members in autonomous vehicles, specifically in the context of lane detection. As most members of the team are part of autonomous driving design teams, such as aUToronto and University of Toronto Formula Racing (UTFR), we concluded that this would be the best way for us to apply course concepts to an important problem that we are highly engaged with. Using the deep learning technique of an encoder-decoder CNN architecture 1, input images are first compressed to a bottleneck, important features are learned by the convolutional layers, and a semantic segmentation map is reconstructed from the input, detecting and classifying lane lines and non-lane line objects at a pixel-by-pixel level.

1.1 Illustration

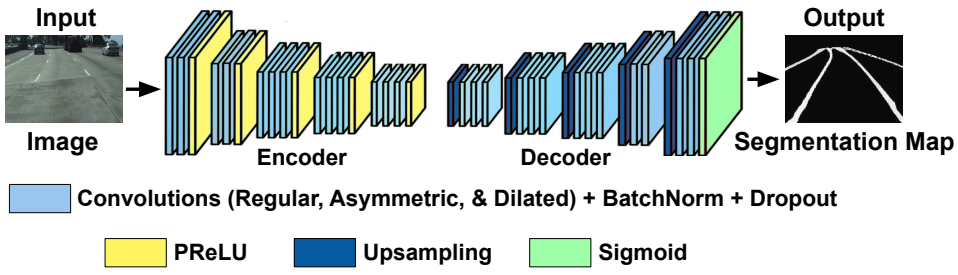

Figure 1: High-level overview of the lane detection pipeline. This illustration was made by modifying figures from Neven et al. (2018) and Mittal et al. (2018).

1.2 Background and Related Work

Advanced Driving Assistance Systems (ADAS) have been rising in popularity in consumer vehicles since the early 2000s, with lane detection becoming a staple around 2005 Research (2021). Such driver assistance systems mainly use camera sensors because vision is the primary modality humans use to view lane markings on the road. In the past, classical methods have been successful at finding lane lines using major assumptions about their shape and colour. While classical methods are able to deal with most lane detection problems, they fall short when it comes to robustness in different situations, failing to adequately deal with different illumination, such as shadows from trees or overpasses, according to Hanuman & Kumar (2021). Occlusion also poses an issue such as when lane lines are covered by another vehicle. Improvements in lane detection ability are therefore necessary as autonomous vehicles become prevalent in society as these systems won't just perform lane keeping but will also be used for localization and planning. Deep-learning based methods aim to increase performance in diverse situations, make the detectors more robust, and make the lane detection technology advanced enough to be used reliably in autonomous vehicles.

Recent developments in deep learning, particularly with the advent of convolutional neural networks (CNNs), have afforded leaps and bounds in the performance of lane detectors due to its capacity to capture spatial information. LaneNet [Wang et al. (2018)], a two stage detector, first generates lane proposals using an encoder-decoder architecture and then localizes the lane lines using LTSMs. However, this method is cumbersome as pixel-level segmentation of lane lines must be labeled in the incoming data. PINet [Ko et al. (2020)] performs instance segmentation by stacking hourglass networks. YOLOp [Wu et al. (2022)] performs segmentation using an encoder and three decoders to not only obtain lane lines but perform object detection and segmentation of the drivable area. However, as the goal of these systems is to run on fast moving vehicles, lane detection methods must be computationally efficient enough to run at real time. To address this, Qin et al. (2021) proposes Ultra-Fast Structure Aware Lane Detection, which showed promising results on the TUSimple and CULanes datasets. This system makes use of residual blocks that feed into convolutional layers, trained on segmentation maps of the lane lines. Once training has completed, the convolutional layers are removed, and a fully-connected classifier network predicts (x,y) pixel coordinates on each row in the image, rather than classifying each individual pixel. This approach tackles issues with occlusion by using context clues from other rows of the image to identify lanes. Nevertheless, while existing methods perform well on datasets, many are not adaptive to various real world environments which motivates the continued search for better architectures and improved data augmentation pipelines that can achieve robust lane detection.

2 Proposed Model

2.1 Data Processing

Our team decided to use the TUSimple Highway Scenes Dataset, which included US Highway images with ground truth annotations for the ego lanes, as well as any lanes to the left and right of the ego vehicle. The images were formatted as 1280x720 RGB images, with (x,y) coordinates marking out the key points of each lane. This dataset included 4,626 images, which was split roughly into 60% training, 20% verification and 20% test sets. The training set was used to tune the model, the verification set to tune the hyperparameters, while the test set was the "holdout data", which was only used to validate a fully trained model.

A key challenge we faced with this dataset was the format of the ground truth. We decided to change the nature of our problem to a semantic segmentation one, which required a segmentation map for training. In order to generate this segmentation map, we fitted a polynomial curve to each lane, and set all pixels within a certain thickness of the line to white, thus generating a new segmented ground truth.

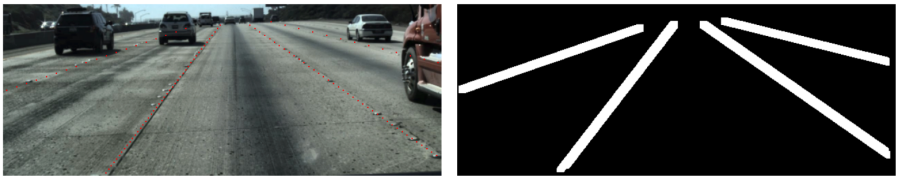

Figure 2: Comparison of TUSimple Ground Truth and Generated Segmentation Map

Furthermore, we recognize that this dataset has limitations when it comes to generalizing for all roads and conditions. Since it only features highway scenes, the trained model will not be able to handle urban driving well, as the occlusion from bumper to bumper traffic, shadows from neighbouring objects, and unseen lane markings such as crosswalks may all confuse the model. Furthermore, as the training dataset only contains daytime, clear weather samples, the model may not generalize well for nighttime, or adverse weather conditions such as rain, fog or snow.

In order to improve the efficiency of training, we created a custom data loader which was able to process image transformations and effectively batch each sample to reduce computational load. This dataloader allowed us to apply our pre-processing and augmentations pipeline through the pytorch transforms.

Our pre-processing included two main steps. The first step was reducing the size of the image. Cropping out the sky reduces the image size from 1280x720 to 1280x500. The image then gets resized to 640x250 - both of these steps serve to speed up training by reducing the size of our input, and only keeping the key segments required for our problem. The second step applies the Contrast Limited Adaptive Histogram Equalization (CLAHE) algorithm to increase the contrast of the image, making the lane lines more clear. These pre-procesing steps are applied to each sampled image.

Figure 3: Example image before and after CLAHE algorithm is applied.

Additionally, we created data augmentations to increase the performance of our model. These augmentation transformations would each have a 25% chance of being applied to each sample and can be stacked together. Our augmentations included horizontal flipping, tilting within 15 degrees, and the addition of gaussian noise to each pixel. In particular, we wanted to highlight two augmentations specific to our task at hand. The first is occlusions, where a random patch of 80x80 pixels would be blacked out, simulating an occlusion from an object, such as another vehicle. The second one is shadows, where a random patch of 80x80 pixels would be significantly darkened, simulating luminosity changes, such as shadows from other objects. Both of these augmentations aim to address the drawbacks of our dataset.

Figure 4: Example random augmentations.

2.2 Model Architecture

Our primary architecture is constructed as an autoencoder, which was selected to allow for the model to classify every pixel in an image as a "lane line" or an "other" object, taking in an input image and outputting a binary segmentation map.

At a high level, the autoencoder consists of an encoder, a bottleneck, and a decoder. The encoder in our network is composed of layered 2D convolutions responsible for feature extraction and compressing the input image to the bottleneck feature space. The input then passes through the bottleneck, which is where the most condensed, feature-rich representation of the image exists. The decoder then subsequently upsamples this condensed representation of the image, using fewer layers than the encoder segment.

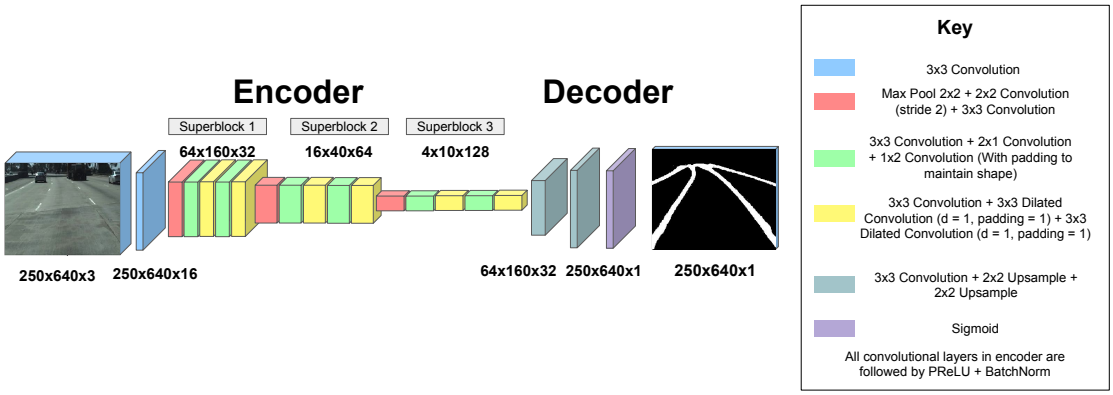

Figure 5: Final Model Architecture

The overall encoder architecture is inspired by similar successful model architectures (such as the encoder segment of UNet or the convolutional segments off ImageNet) that also have blocks of downsampling convolutions followed by blocks of non-downsampling convolutions. It was expected that integrating this type of design into the encoder would allow for several stages of feature learning at each intermediary feature space, resulting in a model that is able to robustly learn both low-level and high-level features from the input images and make better predictions than a purely downsampling model.

In the final model architecture, an initial convolutional layer with 3 input kernels and 16 output kernels passes its output into the main section of the encoder, which consists of downsampling, dilated, and asymmetric convolutional blocks. The model is then composed of a pattern of downsampling followed by four alternating asymmetric and dialated blocks, comprising a superblock, with each superblock being a different feature space for different levels of features to be learned. The encoder consists of an initial block and three superblocks, which contains 16 blocks and 43 convolutional layers in total. All convolutional layers in the encoder are followed by a PReLU activation function and batch normalization to allow for nonlinearity in the model weights and normalization of the weights, respectively.

A PReLU activation function was inserted after each convolutional layer in the encoder, instead of a ReLU activation function, as it was observed that using the PReLU activation function yielded better model accuracy for the same set of hyperparameters and the same initial seed. This is expected to have been caused by the PReLU activation function not setting negative weights to zero, as ReLU does, giving the network greater capacity to model nonlinearities and more complicated relationships that are present in the input data.

The decoder consists of 4 conventional transposed convolutional layers in 2 blocks with upsampling to reverse the downsampling of the image in the encoder. The image tensor is finally passed through a sigmoid function for normalization, outputting an image of size (250, 640, 1) containing pixels with a value between 0 and 1, the value representing the model's predicted probability of the same pixel in the input image being a lane line.

The model output is evaluated pixel-by-pixel against the ground truth using binary cross-entropy, as this loss function was found in parallel testing to result in quicker learning and better generalization than dice loss. Accuracy is assessed through the IoU metric, which computes accuracy as the ratio of the common pixel values between the model and ground truth to the union of all pixel values in both model and ground truth. Pixel values in the output below 0.6 are thresholded to zero prior to calculation of accuracy to remove low-confidence lane line predictions from the model output.

2.3 Baseline Model

First, we convert images of the driving scene into the birds eye view using a homogrpahy matrix. Since we don't have the intrinsic and extrinsic matrix of the camera that collected the TuSimple data, an accurate homogrpahy matrix cannot be determined. Instead, we can approximate a homography matrix by relating a trapezoid in the image plane to a rectangle in the birds eye view plane as shown in Figure 6 using the OpenCV `findHomography` function. A perspective transformation can then be achieved using this matrix and the OpenCV `warpPerspective` function to arrive at the birds eye view.

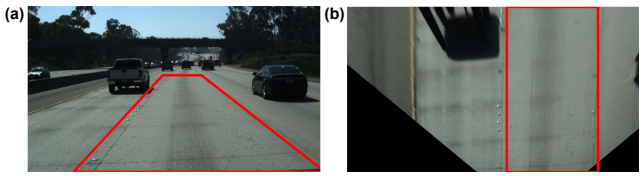

Figure 6: A trapezoid in the image of the driving scene in (a) is related to a rectangle in the birds eye view in image (b). A homography matrix is approximated that creates a mapping between these two perspectives.

Next the image is processed using the following image processing functions: histogram equalization, a sobel filter, colour thresholding, and binary thresholding. Histogram equalization improves contrast in the image making detections robust to lighting conditions. A sobel filter extracts edges by approximating gradients and suppresses noise in the image. The colour thresholding extracts lane marking pixels from both white and yellow lane lines. And binary thresholding eliminates dull colours on the road which improves detections.

This approach consists of a perspective transform that converts all images of the road into the bird's eye view using an approximated homography matrix. , edge pair detection, and line marking candidate extraction (detailed set of steps outlined in Figure 7).

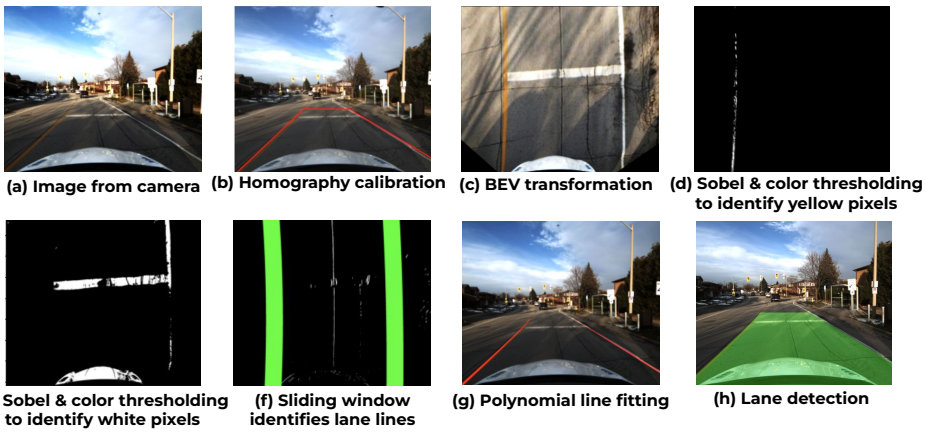

Figure 7: The baseline being used is a traditional, computer-vision approach to lane detection. It consists of a perspective transform described by (a), (b), (c), edge pair detection outlined by (d), (e), and line marking candidate extraction represented by (f),(g),(h).

Finally, a sliding window scanning the image aggregates potential lane line pixels in the driving scene. Window sizes are selected based on prior knowledge of the width of standardized lane lines and fine-tuning results via testing. By conducting a polynomial line fit to these candidate lane line markings, we achieve lane detection.

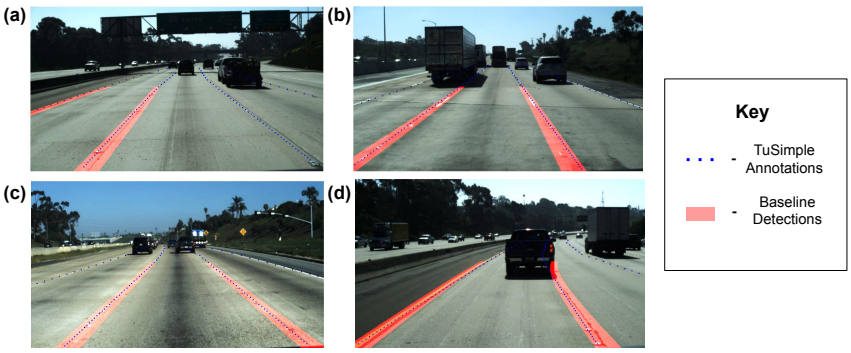

Figure 8: Baseline detections evaluated on TuSimple Dataset.

Quantiatively, we evaluated the performance of the baseline detector on 300 test images from the TuSimple dataset, and we achieved an average true positive rate of 37.65 % and an average false negative rate of 63.45 %. Qualitatively, we observe in Figure 8, that the baseline consistently makes detections up to a set distance but is unable to extend beyond that point and can only accurately detect one lane at a time. This is expected since the baseline algorithm can only detect one lane at a time, and for the majority of cases simply identifies the ego lane except when the lane markings on the road side are more prominent and the left lane is identified instead. Since the TuSimple dataset has an annotation for every lane, the accuracy of the baseline has therefore been reduced significantly despite detections of one lane being made consistently.

3 Results

3.1 Quantitative Results

Training was performed on the final model architecture with hyperparameters of 10 epochs, a batch size of 16, and a learning rate of 0.001, taking 622 minutes to complete.

Final training and validation loss were 2.0984 and 1.8460, respectively. Additionally, final training and validation accuracy were 66.24% and 60.83%, respectively, which indicates that the model was able to train successfully and develop acceptable accuracy, further evidenced by the fact that the training and validation accuracy plateau at 10 epochs. The lack of divergence between the training and validation accuracy indicates that the model is not overfit to training data, implying that hyperparameters have been optimized.

Hitrate serves as a means of assessing a model's performance on a set of data, calculated as the number of images within the set of data where the model predictions had an accuracy of 40% or greater.

Final test accuracy and hitrate were 47.53% and 75.30%, respectively, indicating that the model is able to consistently detect lanes in previously-unseen data, and suggesting that the model will likely accurately detect lane lines in previously unseen images of Toronto roads.

3.2 Qualitative Results

In contrast to the model presented in the progress report, the model performs roughly equally well across all images in the test set, regardless of how lane lines are marked in the images, indicating that the deeper model is able to more effectively detect lanes in previously-unseen data. However, images with occlusions and lane lines closer to the middle of the image remain difficult for the

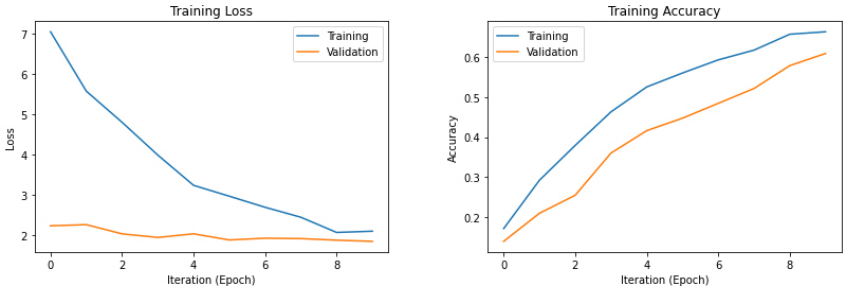

Figure 9: Training and validation loss and accuracy curves for the final model.

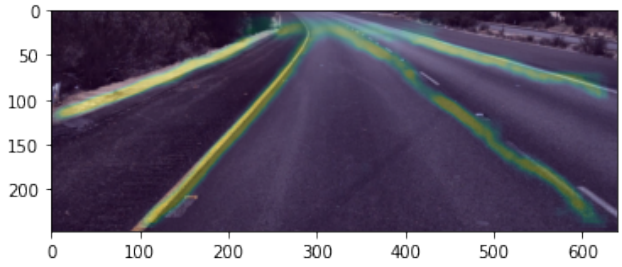

Figure 10: Segmentation map over test split example.

model to detect lanes within, although the model predictions in these cases still have an acceptable accuracy and hitrate, despite the increased challenges posed by the more complex inputs.

Figure 11 below shows a model prediction on an image in the test set (with thresholding to remove pixel values below 0.6), overlaid with the input image. Notably, despite the presence of curved lanes, the model is still able to reliably identify the lanes in the image, with the accuracy of the model on the displayed input being 42.60%.

The model outputs when using test data as inputs demonstrates that it is able to accurately detect lane lines in previously unseen data across a variety of road scenes and lane line occlusions, indicating that the model is able to generalize, and is ready for testing on entirely new data taken from Toronto roads.

3.3 Evaluating Model on New Data

To whether the model is able to more broadly generalize to unseen new data, the model was evaluated on images of the Gardiner expressway in Toronto taken by the team. A dataset was built from these images of the Gardiner expressway, and ground truths were created by manually annotating lane lines in the images. Across these images, the model accuracy was 33.2%, and the hitrate was 45.1%, which is less than on the TUSimple test dataset. This is likely due to the different appearance in the road and lane line markings in Toronto, as well as the increased level of occlusions. However, despite the model not performing as well as it did on the test dataset, the results indicate that the model is still able to accurately identify lane lines in new data and outside the original dataset.

4 Discussion

4.1 Is The Lane Detector Performing Well?

The quantitative performance of the deep learning based lane detection is far better than the baseline model. The classical approach was able to consistently identify lane lines in a single lane while the deep learning based approach identified lane lines in all the lanes on the road as was originally intended. Qualitatively, the deep learning approach performs well but the segmentation outputs are

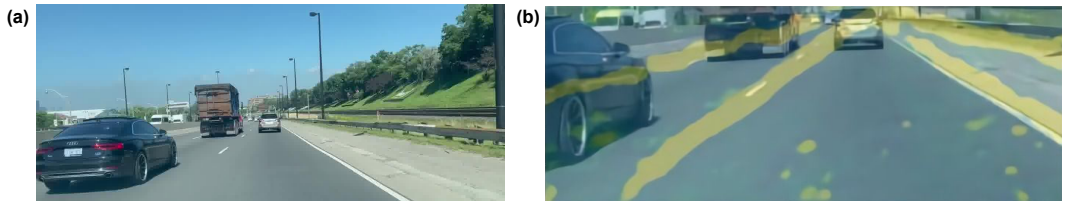

Figure 11: Lane detection output of our model on the Gardiner expressway in Toronto.

very wide. This is a symptom of the fact that the ground truth segmentation masks were generated by us and the ground truth lane lines were thickened during this process. When tested continuously on a video of a road scene, the lane detector worked best when driving straight on clear highways but detections flicker when changing lanes.

4.2 Interesting Lessons And Takeaways

We used Intersection over Union (IoU) as our base metric for accuracy and had to threshold spurious pixels to meaningfully compute a score for accuracy. IoU quantifies the percentage overlap between the ground truth mask and the model's semantic segmentation output. However, IoU is an imperfect metric because over-segmenting or under-segmenting small objects such as lane markings can cause scores to greatly fluctuate. Furthermore, the border of lane markings in an image has pixels that change colour as a gradient. This is in stark contrast to the black-and-white segmentation masks that the predictions are compared against. These reasons explain why it is essential to continue searching for better metrics to quantify accuracy and to holistically look at the performance of an architecture to see it is fit for deployment.

Model generalization to different domains proved to also be a big challenge. As mentioned in the data section, this is largely due to the homogeneity of scenes from the TUSimple Dataset. In order to address this in the future, additional datasets, such as CULanes, should be included for training the model. Datasets like CULanes offer more variety in scenes, especially with the challenging urban driving samples.

4.3 Ethical Considerations

A lane detection system deployed in an autonomous vehicle has the potential to improve road safety but there are also risks that need to be dealt with. Particularly concerning is the ethical dilemma of how the vehicle reacts in a situation of system failure, and whether it prioritizes the safety of the vehicle's occupants or other users of the road. This dilemma has still not been resolved in the field of AI ethics, and remains a contentious point of debate for AI engineers, automotive and technology corporations, lawmakers, and drivers, which will only intensify as automated vehicles replace human-controlled vehicles on the road.

Another ethical concern emerges from the contexts in which this model is used. Given that the model is trained and validated on data from TuSimple (containing images of US highways),it will struggle to identify roads in low-income nations. This concern can be mitigated by training our system on road data obtained from a variety of road systems in nations across the spectrum of incomes and governmental infrastructural investment, with the goal of building a unified system that accurately detects lane lines independent of type of lane marking and road quality.

References

Driverless cars: The benefits and what it means for the future of mobility, 2018. URL https://www.mynrma.com.au/cars-and-driving/driver-training-and-licences/resources/driverless-cars-the-benefits-and-what-it-means-for-the-future-of-mobility.

Automated vehicles for safety, 2021.

A. Sai Hanuman and G. Prasanna Kumar. Survey Analysis of Robust and Real-Time Multi-Lane and Single Lane Detection in Indian Highway Scenarios. In *E3S Web of Conferences*, volume 309 of *E3S Web of Conferences*, pp. 01117, September 2021. doi: 10.1051/e3sconf/202130901117.

Dr. A. Sai Hanuman and G. Prasanna Kumar. Survey analysis of robust and real-time multi-lane and single lane detection in indian highway scenarios. *International Conference on Design and Manufacturing Aspects for Sustainable Energy*, 2021. URL https://www.e3s-conferences.org/articles/e3sconf/pdf/2021/85/e3sconf_icmed2021_01117.pdf.

Yeongmin Ko, Younkwan Lee, Shoaib Azam, Farzeen Munir, Moongu Jeon, and Witold Pedrycz. Key points estimation and point instance segmentation approach for lane detection. *Computer Vision and Pattern Recognition*, 2020. URL https://arxiv.org/pdf/2002.06604.pdf.

Ajay Mittal, Rahul Hooda, and Sanjeev Sofat. Lf-segnet: A fully convolutional encoder–decoder network for segmenting lung fields from chest radiographs - wireless personal communications, Apr 2018. URL https://link.springer.com/article/10.1007/s11277-018-5702-9#citeas.

Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 286–291, 2018.

Erkan Oğuz, Ayhan Küçükmanisa, Ramazan Duvar, and Oğuzhan Urhan. A deep learning based fast lane detection approach. *Chaos, Solitons Fractals*, 155:111722, 2022. ISSN 0960-0779. doi: https://doi.org/10.1016/j.chaos.2021.111722. URL https://www.sciencedirect.com/science/article/pii/S0960077921010766.

Zequn Qin, Huanyu Wang, and Xi Li. Ultra fast structure-aware deep lane detection. *Computer Vision and Pattern Recognition*, 2021. URL https://arxiv.org/pdf/2004.11757v4.pdf.

Hearst Autos Research. Adas: Everything you need to know, Nov 2021. URL https://www.caranddriver.com/research/a31880412/adas/.

Ze Wang, WeiQiang Ren, and Qiang Qui. Lanenet: Real-time lane detection networks for autonomous driving. *Computer Vision and Pattern Recognition*, 2018. URL https://arxiv.org/pdf/1807.01726v1.pdf.

Dong Wu, Manwen Liao, Weitian Zhang, Xinggang Wang, Xiang Bai, Wenqing Cheng, and Wenyu Liu. Yolop: You only look once for panoptic driving perception. *Computer Vision and Pattern Recognition*, 2022. URL https://arxiv.org/pdf/2108.11250v7.pdf.

Youcheng Zhang, lu Zongqing, Xuechen Zhang, Jing-Hao Xue, and Qingmin Liao. Deep learning in lane marking detection: A survey. *IEEE Transactions on Intelligent Transportation Systems*, PP:1–17, 04 2021. doi: 10.1109/TITS.2021.3070111.